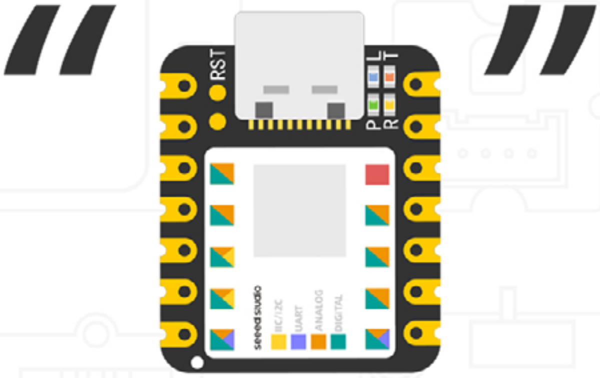
XIAO: Big Power, Small Board

Mastering Arduino and TinyML



Lei Feng, Marcelo Rovai

seeed studio
ARDUINO
EDGE IMPULSE

2024.01

Catalogue

Preface

From the expansive boards of the past, Arduino has come a long way and entered the Seeed Studio XIAO series: thumb-sized yet power-packed, opening a vast horizon for innovation. "XIAO: Big Power, Small Board" dives deep into these capabilities, guiding readers from the basics of Arduino to intricate miniaturized projects. Whether readers want to illuminate an LED or delve into Embedded Machine Learning (TinyML) with XIAO boards and Edge Impulse Studio, this book covers them. Need for prior knowledge? No worries! This book takes a hands-on, project-based approach, ensuring readers grasp the concepts while implementing them. By the end, they will be adept with XIAO and inspired by many user-created projects showcasing the endless possibilities this small board offers.

Acknowledgments

We want to express our sincere gratitude to Jiamou Yang, Yanming Wen, Mengdu Li, Chunchun Tian, Haixu Liu, Tianrui Wang, and Jianjing Huang for their invaluable technical support and manuscript revisions. This book would not have been possible without their contributions.

We extend our deepest gratitude to the entire TinyML4D Academic Network, comprised of distinguished professors, researchers, and professionals. Notable contributions from Marco Zennaro, Brian Plancher, José Alberto Ferreira, Jesus Lopez, Diego Mendez, Shawn Hymel, Dan Situnayake, Pete Warden, and Laurence Moroney have been instrumental in advancing our understanding of Embedded Machine Learning (TinyML).

Special commendation is reserved for Professor Vijay Janapa Reddi of Harvard University. His steadfast belief in the transformative potential of open-source communities, coupled with his invaluable guidance and teachings, has served as a beacon for our efforts from the very beginning.

Acknowledging these individuals, we pay tribute to the collective wisdom and dedication that have enriched this field and our work.

Illustrative images on the e-book and chapter's covers generated by OpenAI's DALL-E via ChatGPT

Introduction

From the expansive boards of the past, Arduino has come a long way and entered the Seeed Studio XIAO series: thumb-sized yet power-packed, opening a vast horizon for innovation. "XIAO: Big Power, Small Board" dives deep into these capabilities, guiding readers from the basics of Arduino to intricate miniaturized projects. Whether readers want to illuminate an LED or delve into Embedded Machine Learning (TinyML) with XIAO boards and Edge Impulse Studio, this book covers them. Need for prior knowledge? No worries! This book takes a hands-on, project-based approach, ensuring readers grasp the concepts while implementing them. By the end, they will be adept with XIAO and inspired by many user-created projects showcasing the endless possibilities this small board offers.

About this Book

Audience

The primary audience for “XIAO: Big Power, Small Board” encompasses hobbyists, students, educators, and professionals in electronics and machine learning who want to explore and maximize the potential of compact hardware platforms. Typically, these readers might hold positions as electronics enthusiasts, DIY project creators, electronics educators, or even junior embedded system developers. As they advance in their careers, they might be eyeing roles such as electronics design engineers, IoT developers, or machine learning hardware integrators.

Our audience possesses a basic understanding of electronics concepts but may have yet to delve deep into Arduino programming or compact hardware design. They likely have encountered standard beginner books on Arduino or general electronics but might have yet to venture into specialized hardware or TinyML. As for skills, they have some hands-on experience with basic electronics or programming but haven’t mastered the intricacies of TinyML or advanced microcontroller functionalities.

What readers will learn

By the end of this book, the reader will understand:

- The fundamentals of open-source hardware, focusing on the capabilities of the Seeed Studio XIAO series.
- How to transition from basic to advanced electronic projects, starting with simple LED controls and advancing to complex applications like telemetry and voice keyword detection.
- The concepts behind prototype design and its practical implications in product development.
- The intricacies of integrating various modules like the infrared receiver, ultrasonic distance sensor, and RTC clock with the XIAO platform.
- The significance and application of Tiny Machine Learning (TinyML), emphasizing its transformative power in hardware like the XIAO nRF52840 Sense and ESP32S3 Sense.
- Techniques to utilize advanced tools such as Edge Impulse Studio for real-world applications like anomaly and object detection and video or sound classification.

The reader will be able to:

- Set up, program, and troubleshoot projects across all XIAO series boards, advancing from basic hardware interactions to intricate project designs.
- Convert abstract ideas into tangible electronic product prototypes, leveraging the insights from the course.
- Design and implement intermediate-level projects such as a Smart Watch and Air Piano using specialized sensors and modules.
- Harness the power of Wi-Fi and MQTT protocols with XIAO ESP32C3 for cloud communications and data exchange.
- Deploy TinyML on different XIAO boards, executing tasks like image, motion, and sound classification besides anomaly and object detection.
- Innovate and extend project ideas, drawing inspiration from a curated collection of XIAO projects and adapting them for custom needs.

Software dependencies

Arduino IDE:

Major updates or changes to the Arduino IDE might affect content related to Arduino development and programming in the book.

Seeed Studio XIAO Libraries:

Updates to libraries specific to the XIAO series can influence the projects or example codes provided.

Edge Impulse Studio:

Significant updates or feature changes on this platform would necessitate adjustments in the TinyML chapters.

MQTT Libraries/Protocols:

Any changes related to MQTT libraries or the protocol itself could influence the content of telemetry and commands.

ESP32 Libraries:

Updates to libraries used by the XIAO ESP32C3 and ESP32S3 board may impact associated projects or examples.

Book outline

Chapter 1: Introduction to Hardware and Programming

In this chapter, readers start with basic programming on XIAO using Arduino IDE. Through simple example programs, they will learn to control LED lights, buttons, buzzers, and other electronic components, mastering core programming concepts like digital I/O, analog I/O, tone generation, and mapping values. By manually typing out code examples line-by-line, they will develop strong coding habits and grasp programming syntax.

Chapter 2: Project Practice for Beginners - Introduction to Prototype Design

In this chapter, readers will learn the basics of designing prototypes with XIAO through beginner-friendly projects. They will start from an idea and quickly create a verification prototype, focusing more on the practical application of code rather than line-by-line analysis. By leveraging Arduino libraries, community resources, and example programs, they will learn how to find and adapt code snippets to achieve desired effects efficiently. Furthermore, they will explore how to design the physical appearance of prototypes by creatively combining electronic hardware with everyday items. The key outcomes are grasping a project-based approach and developing skills to build simple interactive prototypes.

Chapter 3: Intermediate Project Practice—Complex Projects

In this chapter, readers will advance their prototyping skills by creating sophisticated IoT projects with XIAO. They will implement features like Wi-Fi connectivity, MQTT telemetry, and remote control commands using the XIAO ESP32C3. Through complex builds like an intelligent remote door, smartwatch, and air piano, you will hone programming techniques for wireless communication, cloud integration, and embedded control. Optional blueprints will be provided, but readers are encouraged to explore creative enclosure designs with alternative materials. The key outcomes are mastering intermediate IoT prototyping and preparing for advanced tinyML applications.

Chapter 4: Project Practice Advanced - tinyML Application

Among the XIAO series products, the Seeed Studio XIAO nRF52840 Sense has Bluetooth 5.0 wireless connectivity, low power consumption, and comes with onboard 6-axis IMU and PDM microphone sensors. The XIAO ESP32S3 Sense further integrates a camera, digital microphone, and SD card support. Those features make them powerful tools for TinyML (Embedded Machine Learning) projects. TinyML solves problems in a completely different way from traditional programming methods. This chapter will introduce readers to this cutting-edge field by walking through the entire machine-learning workflow from data collection, training, and testing to deployment and inference using the Edge Impulse Studio tool.

Chapter 5: Creative Experiments

Since its launch, the Seeed Studio XIAO series has been widely acclaimed for its compact size, powerful performance, and versatile product range. The maker community has produced a large number of projects created with XIAO. Due to space constraints, we have selected some outstanding projects made with XIAO by our makers. These projects fully demonstrate the powerful functions and wide applications of XIAO. Let us follow the makers' steps, stimulate creativity, and explore the endless possibilities of XIAO. Readers can draw inspiration from these projects, use imagination, and explore new territories with XIAO.

Copyright Statement

This book, "XIAO: Big Power, Small Board," is published under the GNU General Public License version 3.0 (GPL-3.0), ensuring it remains free and open for all to use, modify, and distribute. The GPL-3.0 License is a widely respected and used free software license, guaranteeing users the freedom to run, study, share, and modify the software. The authors, Lei Feng and Marcelo Rovai, along with all contributors, embrace the principles of open knowledge and education, ensuring that this valuable resource remains accessible to those who seek to advance their skills in Arduino, TinyML, and beyond. By choosing this license, we commit to fostering a community-driven approach to learning and innovation. For more details on the rights and limitations under this license, please refer to the official GPL-3.0 documentation.

Online Version and Community Feedback

To ensure that our readers always have access to the most current and enhanced version of "XIAO: Big Power, Small Board," we have established an online version of the book available at https://mjrovai.github.io/XIAO_Big_Power_Small_Board-ebook/. This platform not only serves as a repository for the latest updates but also as a space for our community to engage, offer feedback, and propose improvements. We highly encourage our readers to visit this site regularly to check for updates, additional resources, and to become part of our growing community of enthusiasts and professionals. Your insights and contributions are invaluable to us, and we look forward to your active participation in enriching this resource.

Chapter 1:
Introduction to Hardware and Programming

In this unit, we will enter the world of electronics and programming and explore how to control hardware through code. Starting with the example program, Blink, we will learn how to light up an LED, turn the light on and off through a button, control the sound of a passive buzzer, and so on. In each task, we will master commonly used programming languages, such as digital input/output, analog input/output, tone and map functions, etc., and learn the primary usage of libraries. The programs in this unit are relatively simple. During the learning process, write the program code for each task by hand, develop good habits, and avoid program upload failures due to errors in symbols or unfamiliar rules.

1.1
First Arduino program with Seeed Studio XIAO: Blink

Arduino is a globally popular open-source electronic prototyping platform, including various models of Arduino development boards and the Arduino IDE software platform. Because of its open, convenient, and easy-to-start characteristics, it has become the first choice for many software and hardware beginners.

With it, you can quickly complete project development and implement your ideas. To date, Arduino has introduced various models of controllers and numerous peripheral modules, such as sensors, actuators, expansion boards, etc. These modules can implement various exciting and practical projects when used with Arduino.

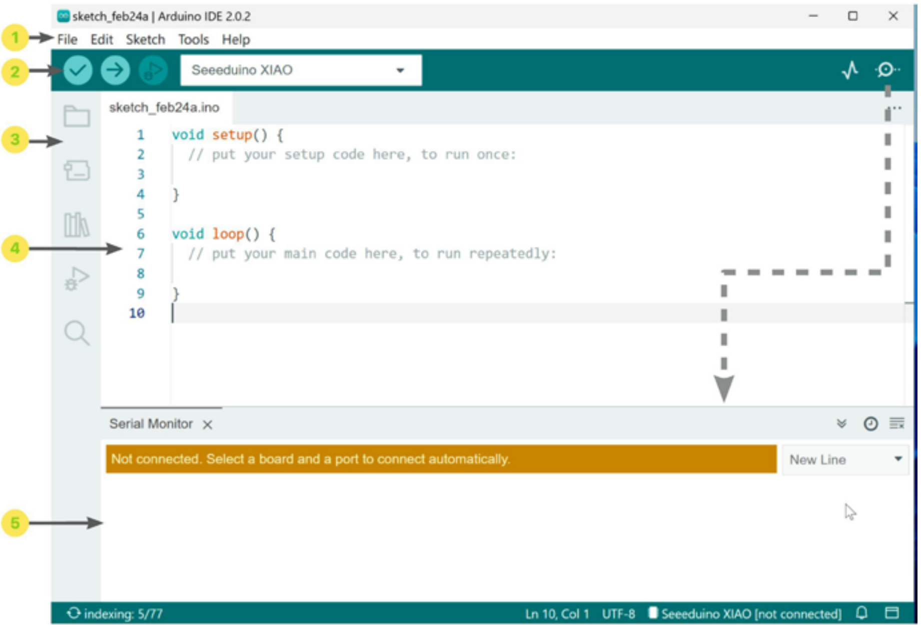
The Seeed Studio XIAO series products we are learning about today are development boards derived from Arduino. They belong to the Seeeduino series and are the smallest members of this series.

1.1.1 Arduino IDE Text Editor

We need to program the hardware through the Arduino IDE text editor. If you have not installed the Arduino IDE, go to the download page to install it: Software. The Arduino IDE (Integrated Development Environment) is a programming software designed explicitly for Arduino. Through it, we can write and upload different programs for Arduino hardware. When we open the Arduino IDE software, it will create a new file named Sketch, which we can rename.

For Windows Users

The interface of the Arduino IDE is spotless, and can be divided into four parts: menu bar, toolbar, editing area, and debug window.


sketch_feb24a | Arduino IDE 2.0.2
1
File Edit Sketch Tools Help
2
Seeeduino XIAO
3
sketch_feb24a.ino
4
Serial Monitor
New Line
5
indexing: 5/77
Ln 10, Col 1 UTF-8 Seeeduino XIAO [not connected]

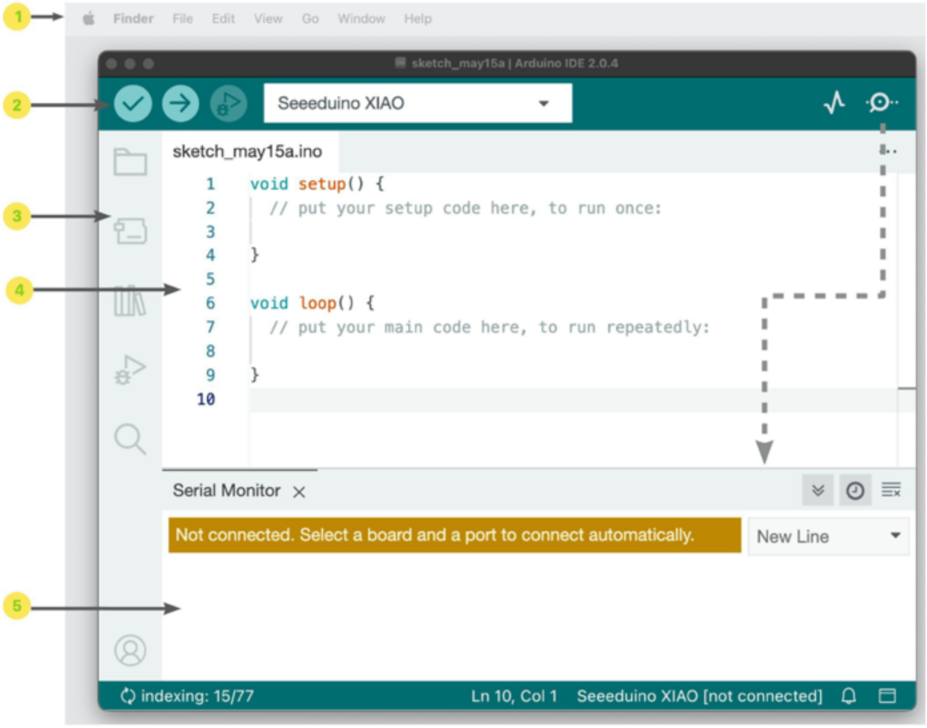
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Not connected. Select a board and a port to connect automatically.

1. Menu bar: Includes files, edit, sketch, tools, and help, such as new, save, example programs, select serial port, etc.
2. Horizontal toolbar: Contains several commonly used function buttons: verify, upload, debug, board selection, serial plotter, and serial monitor selection.
3. Vertical toolbar: Contains shortcuts to the project folder, board manager, library manager, debug, and search.
4. Code editing area: This is where you write program code, just as we usually type text in a Word window. Write the program code in this area.
5. Serial monitor, output window: On the right side of the horizontal toolbar, you can open or close the serial monitor window.

For MAC Users

Except for the location of the menu bar (at the top), which is slightly different from Windows users, all other tools and experiences are the same.



1
2
3
4
5
Finder File Edit View Go Window Help
sketch_may15a | Arduino IDE 2.0.4
sketch_may15a.ino
Serial Monitor
New Line

Seeeduino XIAO

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

```

Not connected. Select a board and a port to connect automatically.

indexing: 15/77   Ln 10, Col 1   Seeeduino XIAO [not connected]

1.1.2 Adding Seeed Studio XIAO to Arduino IDE

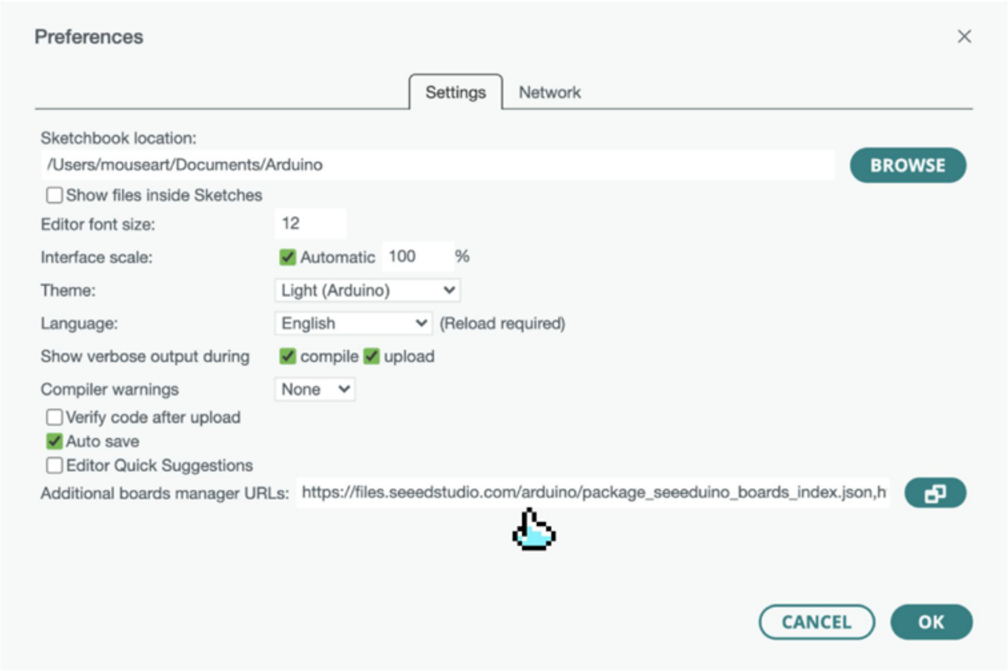
- Attention -

Due to space limitations, all parts of this course's program code and hardware connection are based on Seeed Studio XIAO SAMD21. Most of the code in the book can be applied to all products in the Seeed Studio XIAO series. If there are exceptions, they will be additionally marked or explained for applicable hardware. If not marked, they apply to multiple products.

We must add the Seeed Studio XIAO series products to the Arduino IDE to start our learning journey.

- For Windows users, first, open your Arduino IDE, click "File→Preferences" in the top menu bar, as shown in the figure, and copy the following URL into "Additional Boards Manager URLs."
- For Mac users, first, open your Arduino IDE, click "Arduino IDE→Preferences" in the top menu bar, as shown in the figure, and copy the following URL into "Additional Boards Manager URLs."

- For Seeed Studio XIAO SAMD21, XIAO nRF52840, and XIAO nRF52840 Sense, copy the link address below: https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json
- For Seeed Studio XIAO RP2040, copy the link address below: https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
- For Seeed Studio XIAO ESP32C3, XIAO ESP32S3, copy the link address below: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json



Preferences

Settings | Network

Sketchbook location: /Users/mouseart/Documents/Arduino

☐ Show files inside Sketches

Editor font size: 12

Interface scale: ☑ Automatic 100 %

Theme: Light (Arduino)

Language: English (Reload required)

Show verbose output during ☑ compile ☑ upload

Compiler warnings None

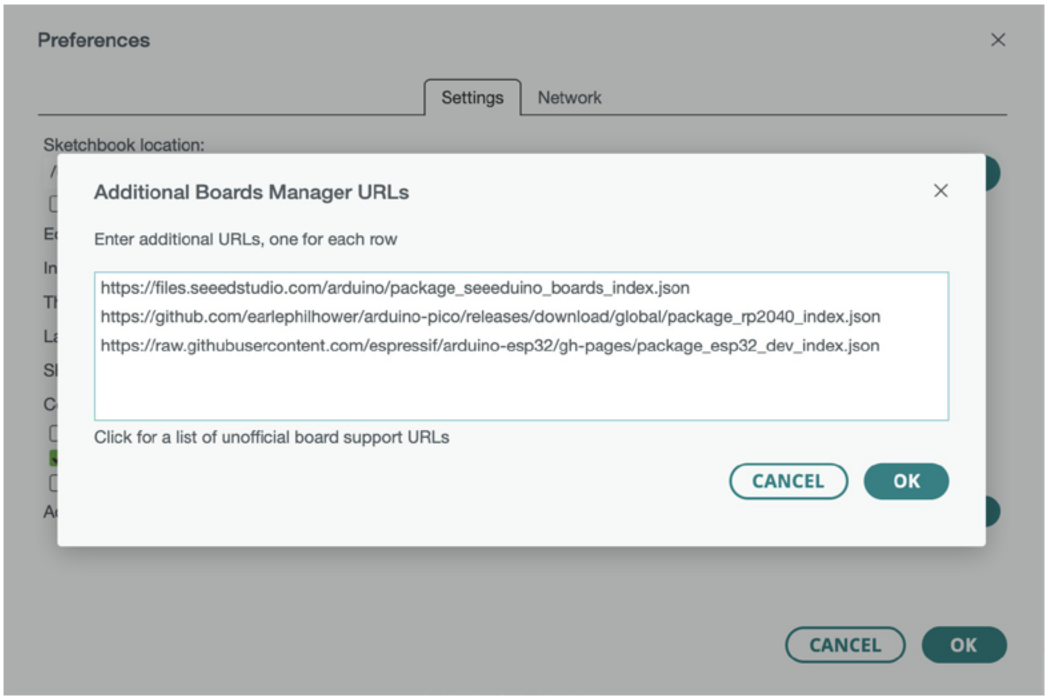
☐ Verify code after upload

☑ Auto save

☐ Editor Quick Suggestions

Additional boards manager URLs: https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json,h

If you frequently use multiple different models of XIAO at the same time, you can click on the ⧉ icon on the right side of the address bar and add all three addresses above to the board manager, as shown in the figure below.



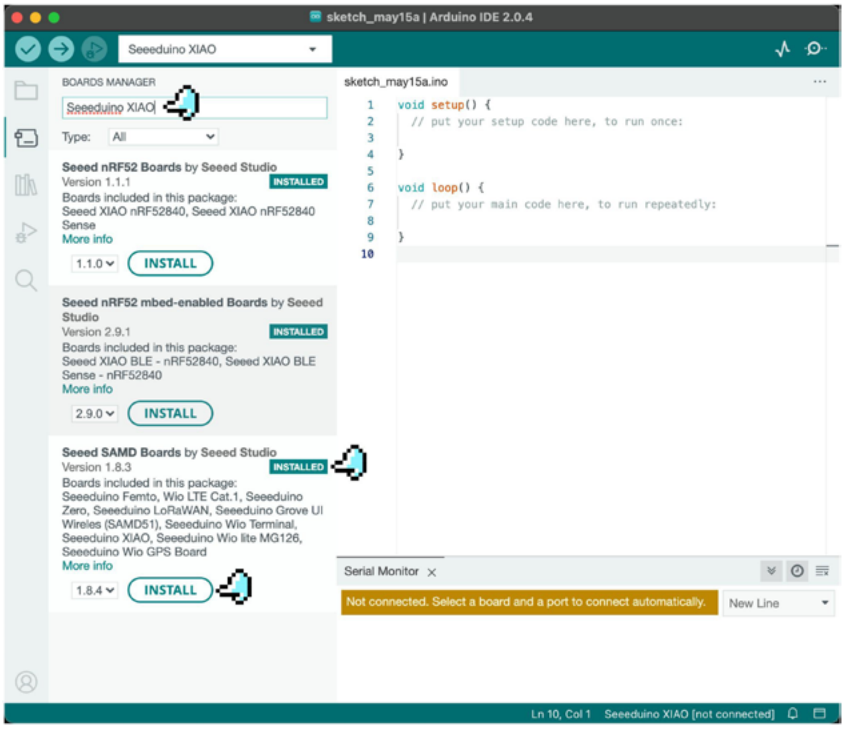
Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json
https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
```

Click for a list of unofficial board support URLs

Next, click on "Tools → Board → Board Manager", enter the keyword `Seeeduino XIAO` in the search bar, find `Seeed SAMD Boards` in the appeared entries, and click `INSTALL`.



sketch_may15a | Arduino IDE 2.0.4

Seeeduino XIAO

BOARDS MANAGER

Seeeduino XIAO

Type: All

Seeed nRF52 Boards by Seeed Studio
Version 1.1.1 INSTALLED
Boards included in this package:
Seeed XIAO nRF52840, Seeed XIAO nRF52840 Sense
More info
1.1.0 INSTALL

Seeed nRF52 mbed-enabled Boards by Seeed Studio
Version 2.9.1 INSTALLED
Boards included in this package:
Seeed XIAO BLE - nRF52840, Seeed XIAO BLE Sense - nRF52840
More info
2.9.0 INSTALL

Seeed SAMD Boards by Seeed Studio
Version 1.8.3 INSTALLED
Boards included in this package:
Seeeduino Femto, Wio LTE Cat.1, Seeeduino Zero, Seeeduino LoRaWAN, Seeeduino Grove UI Wireles (SAMD51), Seeeduino Wio Terminal, Seeeduino XIAO, Seeeduino Wio lite MG126, Seeeduino Wio GPS Board
More info
1.8.4 INSTALL

sketch_may15a.ino

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Serial Monitor

Not connected. Select a board and a port to connect automatically.

New Line

Ln 10, Col 1 Seeeduino XIAO [not connected]

When the installation starts, you will see an output pop-up window. After the installation is complete, an "INSTALLED" option will appear.

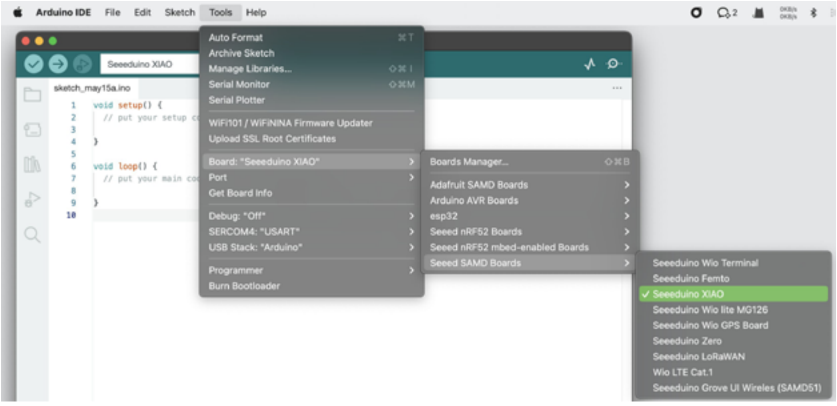
- Attention -

- *Enter "RP2040" in the search bar to find the installation package for Seeed XIAO RP2040.*
- *Enter "XIAO nrf52840" to find two installation packages: Seeed nRF52 Boards (for low-power projects) and Seeed nRF52 mbed-enabled Boards (for higher-power TinyML projects).*
- *Enter "ESP32" to find the installation package for ESP32 by Espressif Systems.*

Connecting Seeed Studio XIAO to Arduino IDE

Connect XIAO to the computer with a data cable, as shown in the figure below:

Next, click on "Tools → Board", find "Seeeduino XIAO" and select it, as shown in the figure below.



- Attention -

- *If your development board is XIAO nRF52840, please select Seeed XIAO nrf52840.*
- *If your development board is XIAO nRF52840 Sense, please select Seeed XIAO nrf52840 Sense.*
- *If your development board is XIAO RP2040, please select Seeed XIAO RP2040.*
- *If your development board is XIAO ESP32C3, please select XIAO_ESP32C3.*
- *If your development board is XIAO ESP32S3, please select XIAO_ESP32S3.*

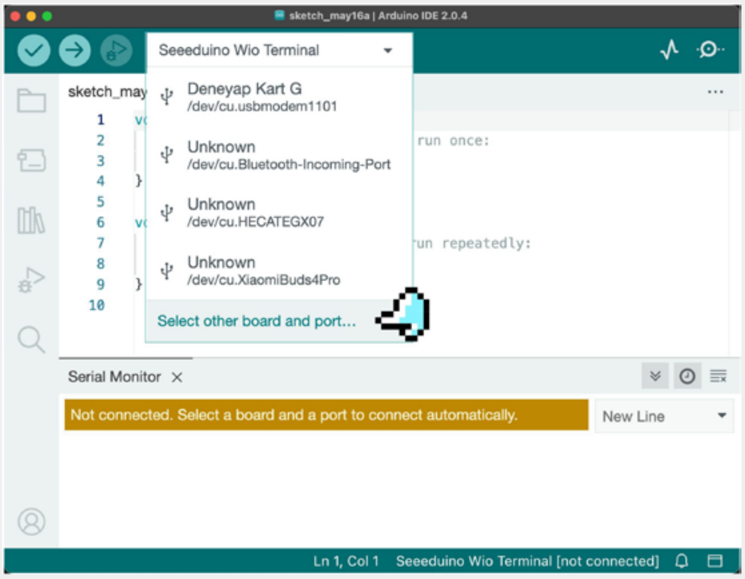
Check if the port connection is correct; if not, select it manually.

- The serial port on Windows systems is displayed as “COM+number,” as shown in the figure below.
- The serial port name on Mac or Linux systems is generally /dev/tty.usbmodem+number or /dev/cu.usbmodem+number, as shown in the figure below.

Now, we can start programming XIAO through the software.

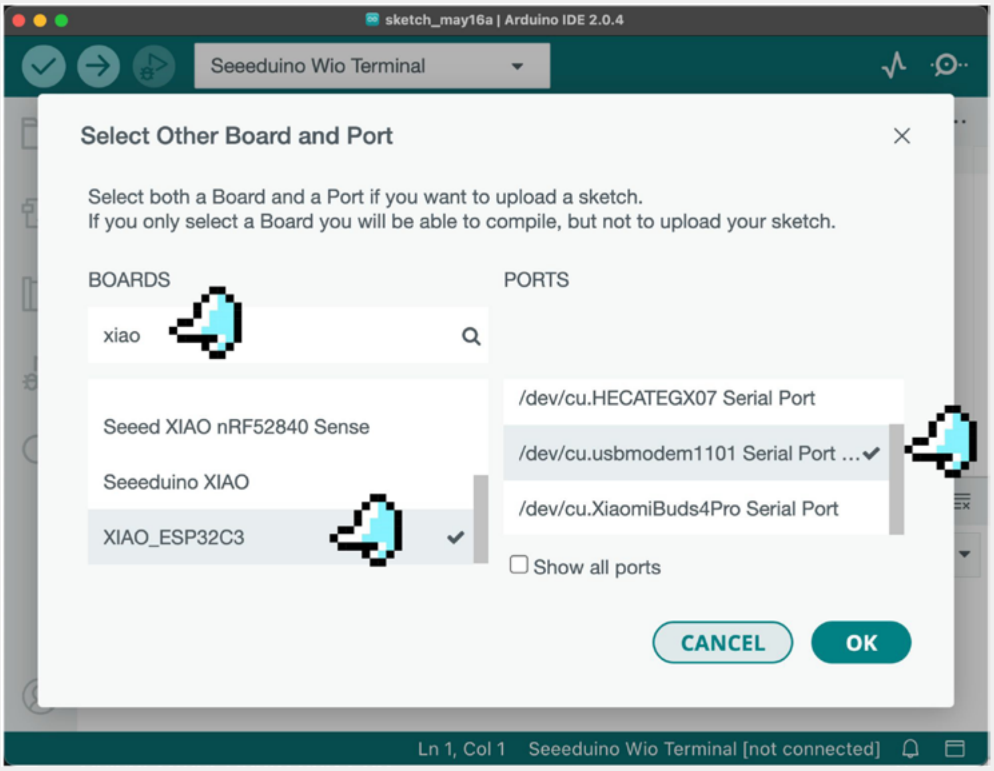
- Attention -

XIAO ESP32C3 may not be adequately recognized in Arduino IDE 2, and you need to specify the development board and port manually.



Not connected. Select a board and a port to connect automatically.

When ESP32C3 is plugged into a PC with Arduino IDE 2, it may not be able to match the correct development board automatically. As shown in the figure below, the display is not the XIAO ESP32 development board; you need to specify manually. Select " Other Board & Port..." from the Port drop-down menu. Enter "xiao" in the search bar of the development board, select the XIAO_ESP32C3 development board from the filtered list below, and confirm after selecting the port on the right.



sketch_may16a | Arduino IDE 2.0.4

Seeeduino Wio Terminal

Select Other Board and Port

Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

BOARDS

xiao

Seeed XIAO nRF52840 Sense

Seeeduino XIAO

XIAO_ESP32C3 ✓

PORTS

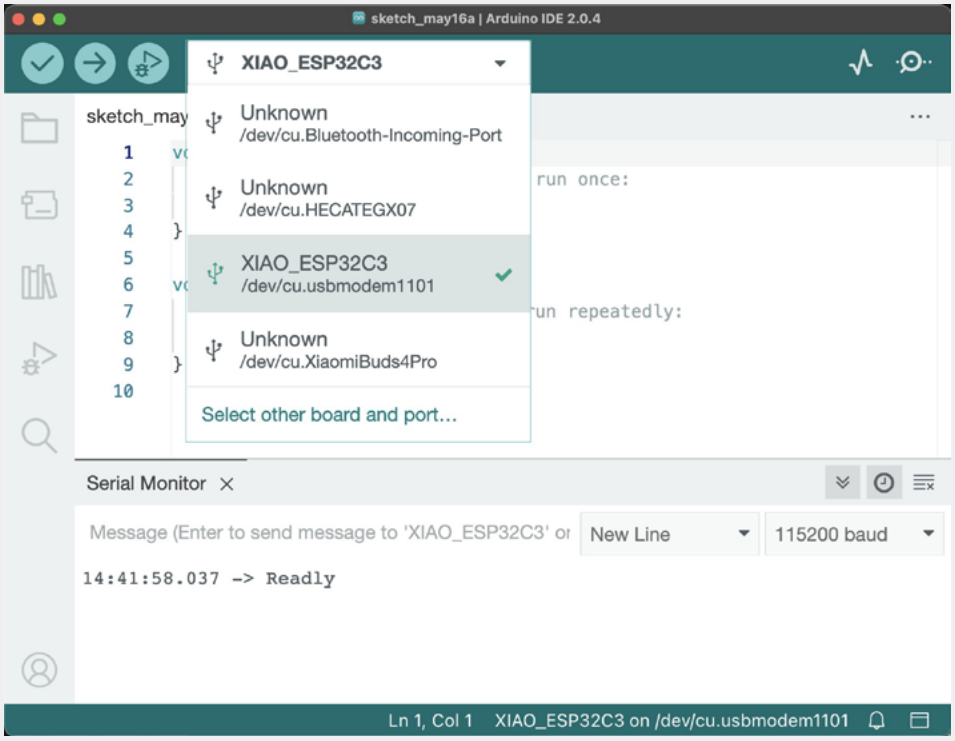
/dev/cu.HECATEGX07 Serial Port

/dev/cu.usbmodem1101 Serial Port ... ✓

/dev/cu.XiaomiBuds4Pro Serial Port

□ Show all ports

Now you can see that the development board and port are in the correct state.



sketch_may16a | Arduino IDE 2.0.4

XIAO_ESP32C3

Unknown
/dev/cu.Bluetooth-Incoming-Port

Unknown
/dev/cu.HECATEGX07

XIAO_ESP32C3 ✓
/dev/cu.usbmodem1101

Unknown
/dev/cu.XiaomiBuds4Pro

Select other board and port...

Message (Enter to send message to 'XIAO_ESP32C3' or

New Line

115200 baud

```
14:41:58.037 -> Readly
```

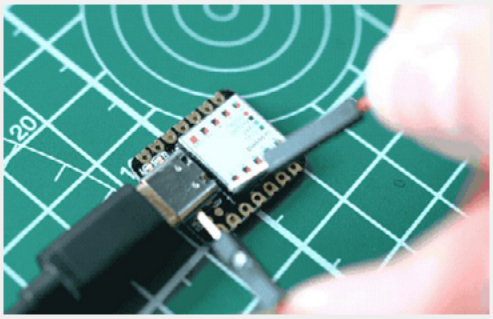
- Reset Seeed Studio XIAO -

Sometimes when the program upload fails, the Seeed Studio XIAO port may disappear, and we need to perform a reset operation. The reset method will be different for different models of XIAO.

Reset of Seeed Studio XIAO SAMD21

- *Connect XIAO SAMD21 to your computer.*
- *Open "Blink" in the Arduino IDE sample program and click upload.*
- *While uploading, short circuit the RST pin in the figure once with tweezers or a short wire.*
- *The reset is completed when the orange LED flashes and lights up.*

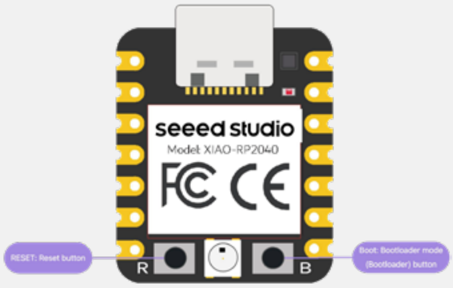
As shown in the figure below.



Reset of Seeed Studio XIAO PR2040

- *Connect Seeed Studio XIAO RP2040 to your computer.*
- *Press the reset button marked with "R" once, the position is shown in the figure below.*

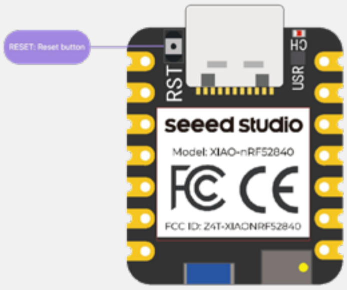
If this does not work, hold down the Boot button marked with "B", connect the board to your computer while holding down the BOOT button, and then release it to enter the bootloader mode.



Reset of Seeed Studio XIAO nRF52840 and Sense version

- *Connect Seeed Studio XIAO nRF52840 or Sense version to your computer.*
- *Press the reset button marked with "RST" once, the position is shown in the figure below.*

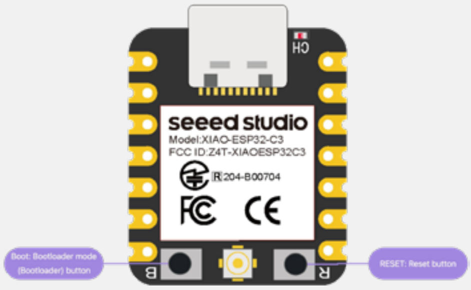
If this does not work, you can quickly click it twice to enter the bootloader mode.



Reset of Seeed Studio XIAO ESP32C3

- *Connect Seeed Studio XIAO ESP32C3 to your computer.*
- *Press the reset button marked with "R" once, the position is shown in the figure below.*

If this does not work, hold down the Boot button marked with "B", connect the board to your computer while holding down the BOOT button, and then release it to enter the bootloader mode.

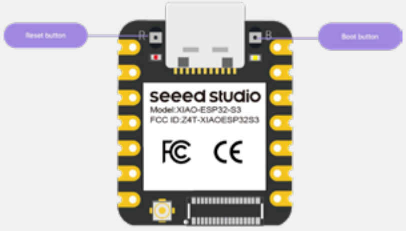


Reset of Seeed Studio XIAO ESP32S3

- *Connect Seeed Studio XIAO ESP32S3 to your computer.*

- *Press the reset button marked with "R" once, the position is shown in the figure below.*

If this does not work, hold down the Boot button marked with "B", connect the board to your computer while holding down the BOOT button, and then release it to enter the bootloader mode.



Structure of Arduino Programs

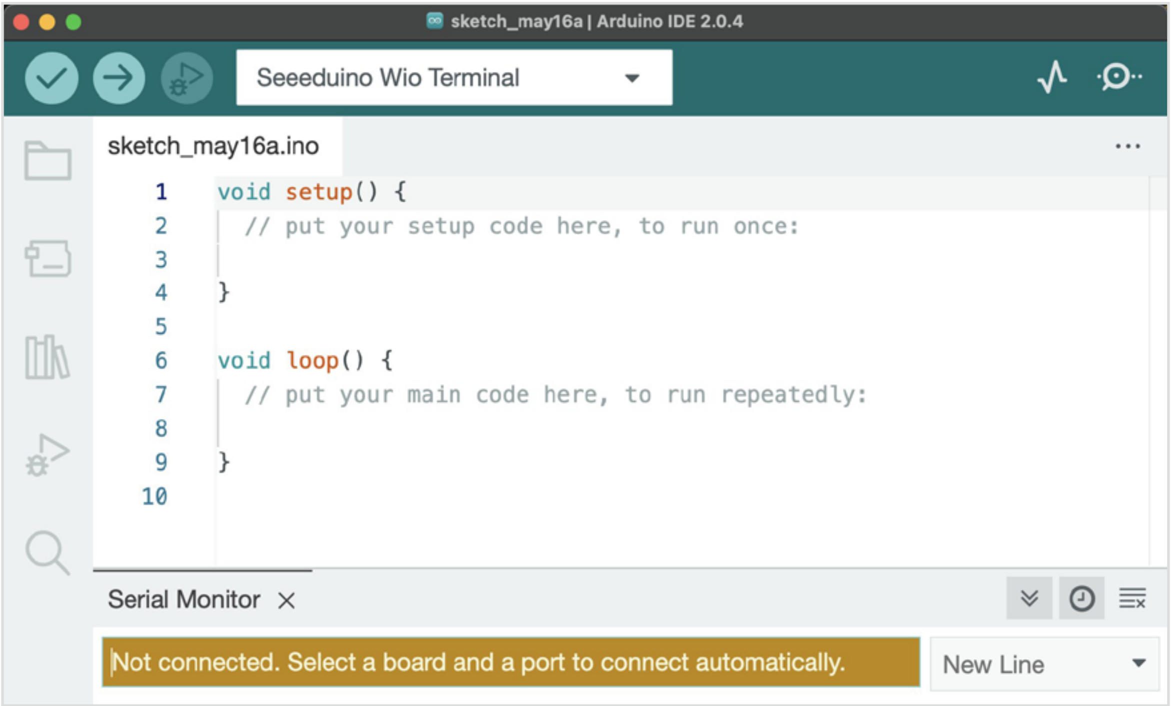
Now that we have the development board, how can we write programs into it to control its functions? That's when the Arduino IDE text editor comes in handy. We've already introduced the interface functions of Arduino IDE in the introduction, it's an important tool for writing and uploading programs. Arduino programs consist of two basic functions:

`setup()`

This function is called when the program begins. Use it to initialize variables, pin modes, start using libraries, etc. `setup()` runs only once each time the Arduino board is powered on or reset.

`loop()`

After the program in `setup()` is executed, the program in `loop()` begins to execute. The program in `loop()` runs repeatedly.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

```

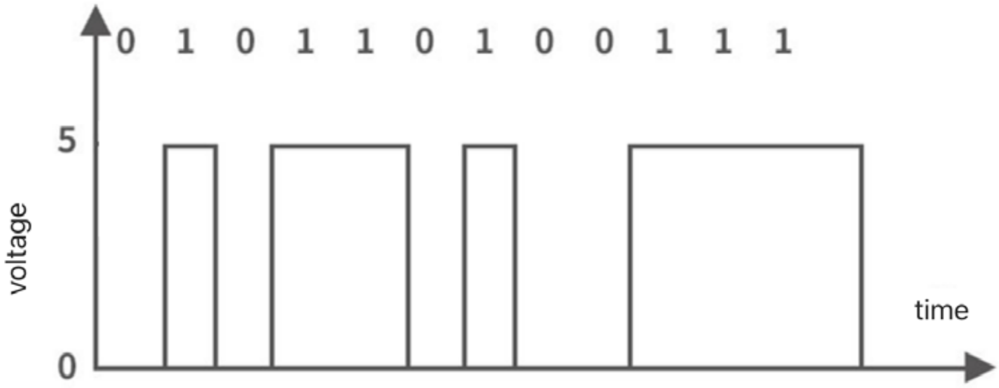
Not connected. Select a board and a port to connect automatically.

- Knowledge Window -

- *The contents after "/* */" and "//" are comments to help you understand and manage code, the comments will not affect the normal operation of the program;*
- *When writing programs, we need to use "{}" to wrap a set of codes;*
- *After each line of code, use ";" as an end symbol to tell the Arduino editor that this line of code instruction is over.*

Digital Signals and I/O Settings

Simply put, digital signals are signals represented in binary form of 0 and 1. In Arduino, digital signals are represented by high and low levels, high level means digital signal 1, and low level means digital signal 0. Seeed Studio XIAO has 11 digital pins, we can set these pins to perform the function of inputting or outputting digital signals.



In Arduino, you can use functions to set the status and function of pins. Here are the basic steps to set pins through functions:

1. First, determine the pin number of the pin you want to control.
2. In the Arduino code, use the `pinMode()` function to set the function of the pin, such as input or output. For example, to set the pin to output mode, you can use the following code:

```
int ledPin = 13; // The pin to be controlled void setup() {
pinMode(ledPin, OUTPUT); // Set the pin to output mode
}
```

3. Once you have set the pin to output mode, you can use the `digitalWrite()` function to set the status of the pin, such as setting it to high or low level. For example, to set the pin to high level, you can use the following code:

```
digitalWrite(ledPin, HIGH); // Set the pin to high level
```

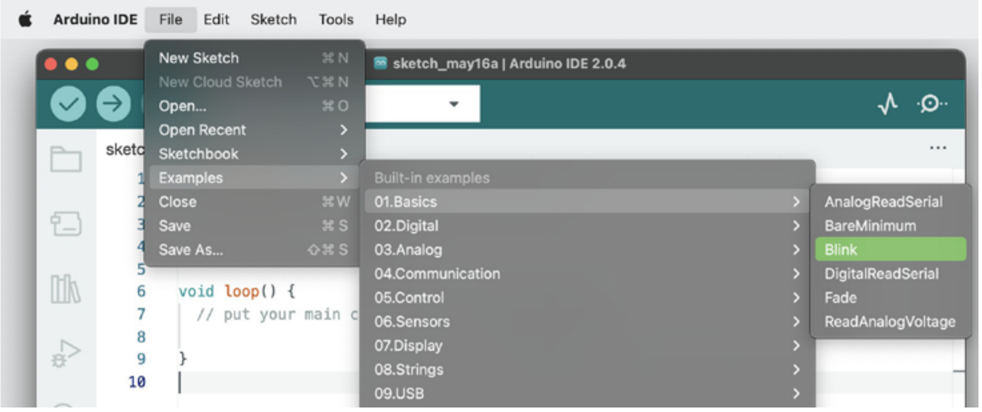
4. If you set the pin to input mode, you can use the `digitalRead()` function to read the status of the pin, such as detecting whether it is high or low level. For example, to read the status of the pin and save it to a variable, you can use the following code:

```
int buttonPin = 2; // The pin to read the status from
int buttonState = 0; // The variable to save the status
void setup() {
    pinMode(buttonPin, INPUT); // Set the pin to input mode
}
void loop() {
    buttonState = digitalRead(buttonPin); // Read the status of the
}
```

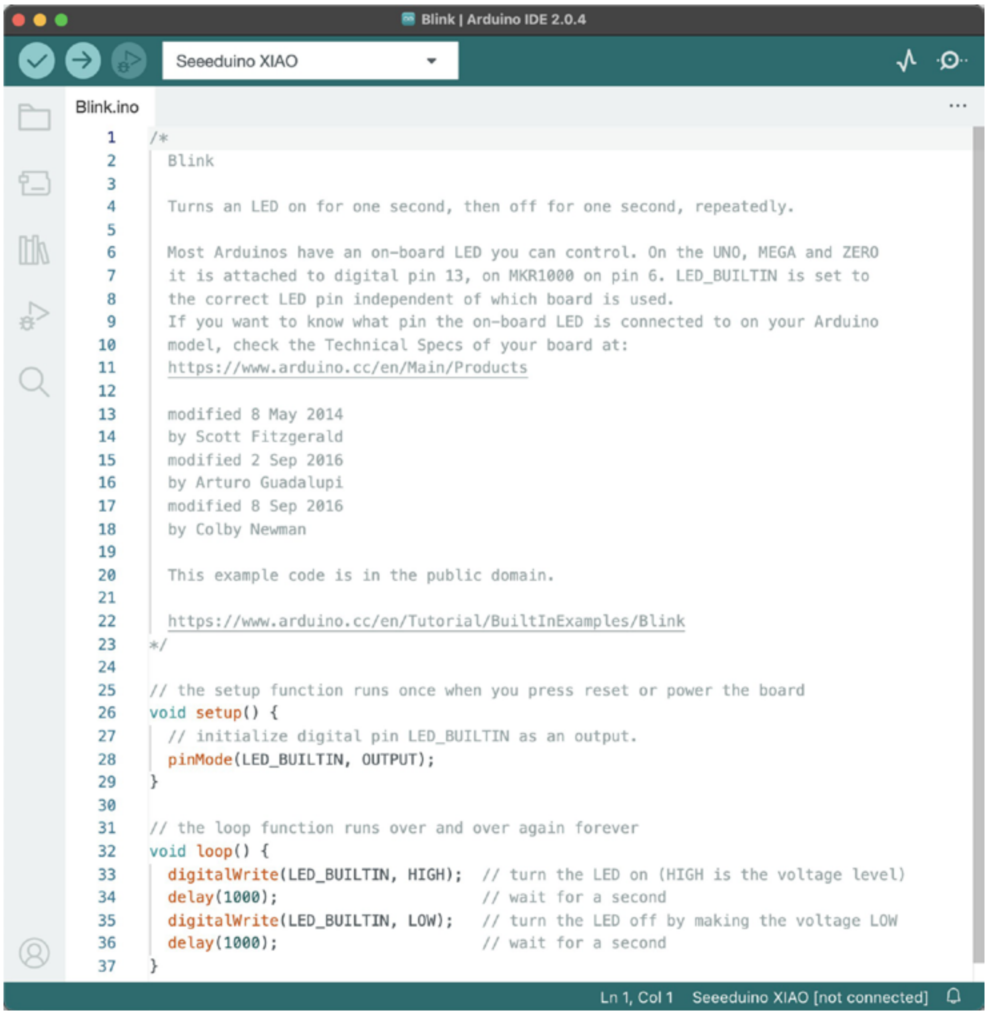
By using functions like `pinMode()`, `digitalWrite()`, and `digitalRead()`, you can easily set and control the status and function of pins in Arduino.

1.1.3 Task 1: Run Blink to Make XIAO's LED Flash

Just as “Hello World” is the first section in all programming languages, “Blink” is akin to “Hello World” in Arduino programming. It is the key to our journey in learning Arduino. Arduino provides many example codes to help us get started quickly, and Blink is one of them. We can select “File → Examples → 01.Basics → Blink” in the Arduino window to open the example program Blink.



After opening the example program, you can see the following code, which implements the effect of LED flashing. You can see that the code has orange and green color prompts, which proves that your input is correct. Pay attention to the difference between uppercase and lowercase.



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeated

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the b
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the
    delay(1000);                     // wait for a second
    digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making t
    delay(1000);                     // wait for a second
}
```

Code Analysis

`pinMode(LED_BUILTIN, OUTPUT);`

The first thing the code does is to initialize `LED_BUILTIN` as an output pin in the `setup()` function. Most Arduino series boards default the onboard LED to digital pin 13. The constant `LED_BUILTIN` connects the onboard LED to pin 13.

`digitalWrite(LED_BUILTIN, HIGH);`

In the `loop()` function, we set the `LED_BUILTIN` pin to the "on" state, outputting 5V or 3.3V voltage to this pin, which can be represented by `HIGH`. However, note that all I/O pins on XIAO are 3.3V. Do not input a voltage exceeding 3.3V, or it may damage the CPU.

`digitalWrite(LED_BUILTIN, LOW);`

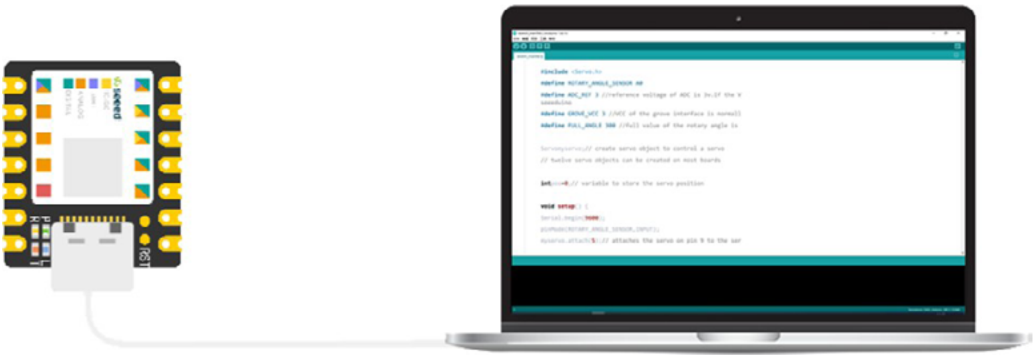
What comes on must turn off. This statement sets the `LED_BUILTIN` pin to the "off" state, outputting 0V voltage to this pin, which can be represented by `LOW`.

```
delay(1000);
```

This is a delay statement. It means that the LED can maintain the “on” or “off” state for 1 second, because the parameter in the function is in milliseconds, so 1000 milliseconds is 1 second. After controlling the “on” and “off” statements of the LED, a delay must be added, and the waiting time should be the same to ensure that the LED flashes evenly.

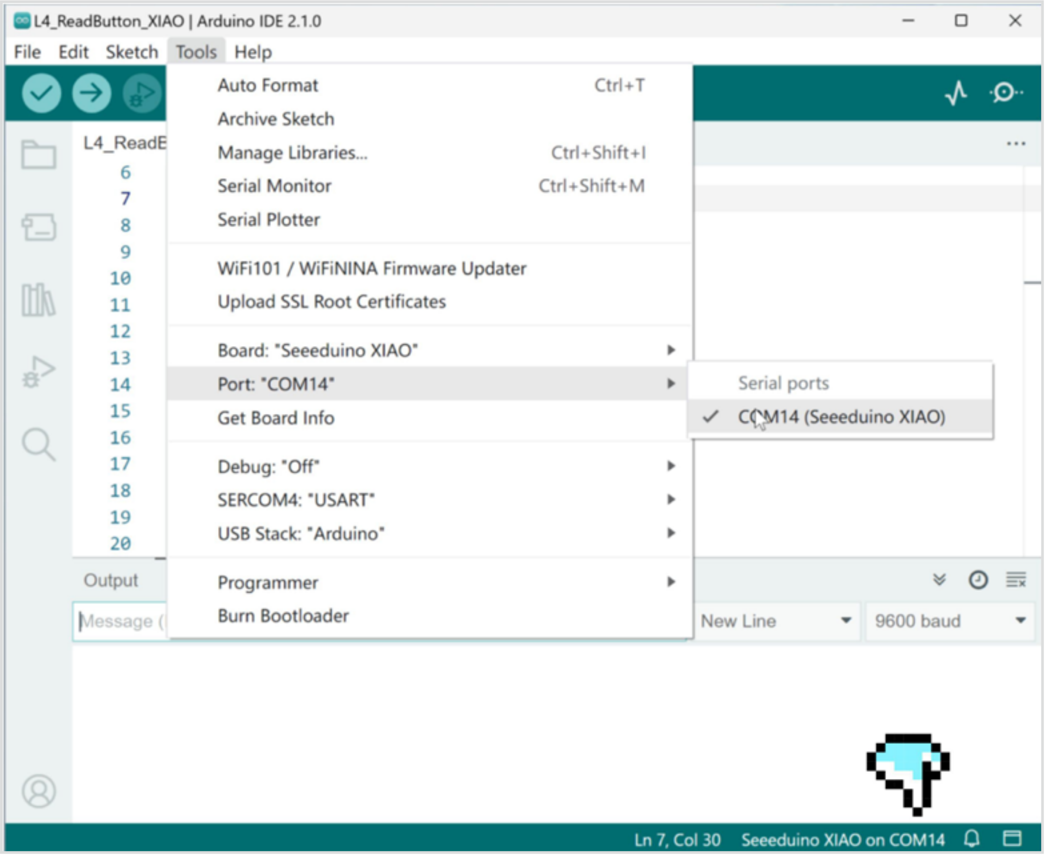
Upload the Program

Next, we will learn how to upload the program. Use the data cable in the kit to connect XIAO to the computer, as shown in the figure.



Choose the serial port of the development board from the “Tools” bar. For Windows users, it is generally COM3 or a larger number. Select it as shown in the figure below.

If several ports are displayed for selection, unplug the data cable, reopen the “Tools” bar, and the port that disappears is the XIAO port. Reconnect the circuit board and then select this serial port. After selecting the board and the serial port, you can see the controller model and corresponding serial port that have been set up in the lower right corner of the IDE interface.



L4_ReadButton_XIAO | Arduino IDE 2.1.0

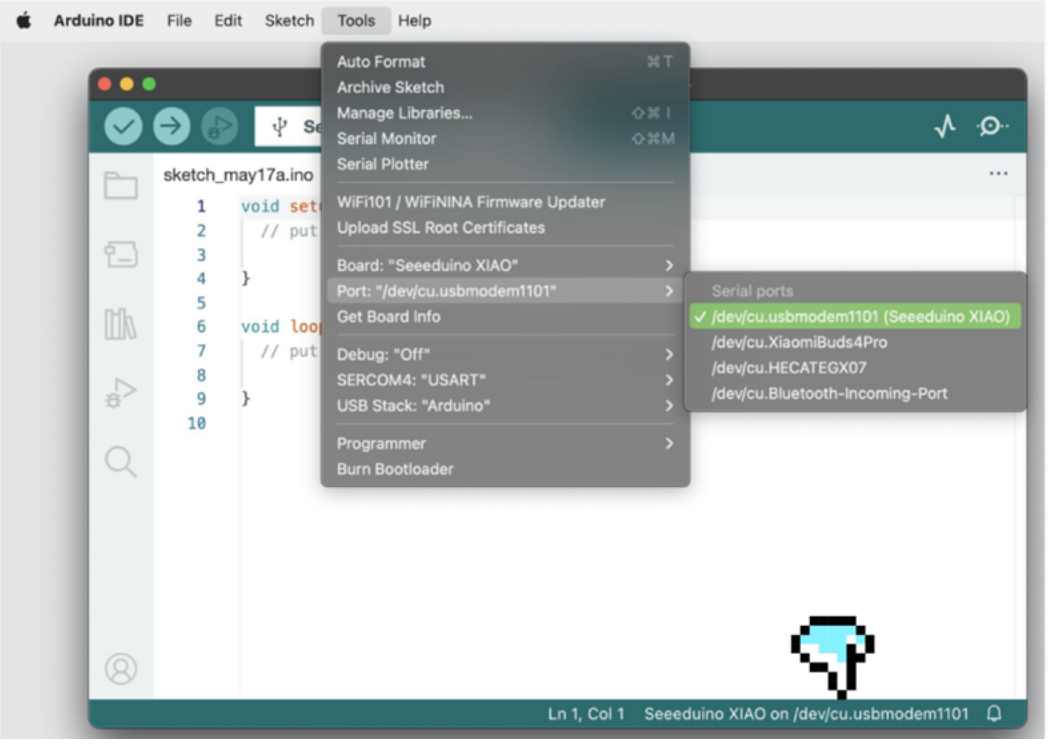
Board: "Seeeduino XIAO"

Port: "COM14"

✓ COM14 (Seeeduino XIAO)

Ln 7, Col 30 Seeeduino XIAO on COM14

In Mac or Linux systems, the serial port name is generally `/dev/tty.usbmodem+number` or `/dev/cu.usbmodem+number`, as shown in the figure below.



```
void set
  // put
}

void loo
  // put
}
```

Board: "Seeeduino XIAO"

Port: "/dev/cu.usbmodem1101"

✓ /dev/cu.usbmodem1101 (Seeeduino XIAO)

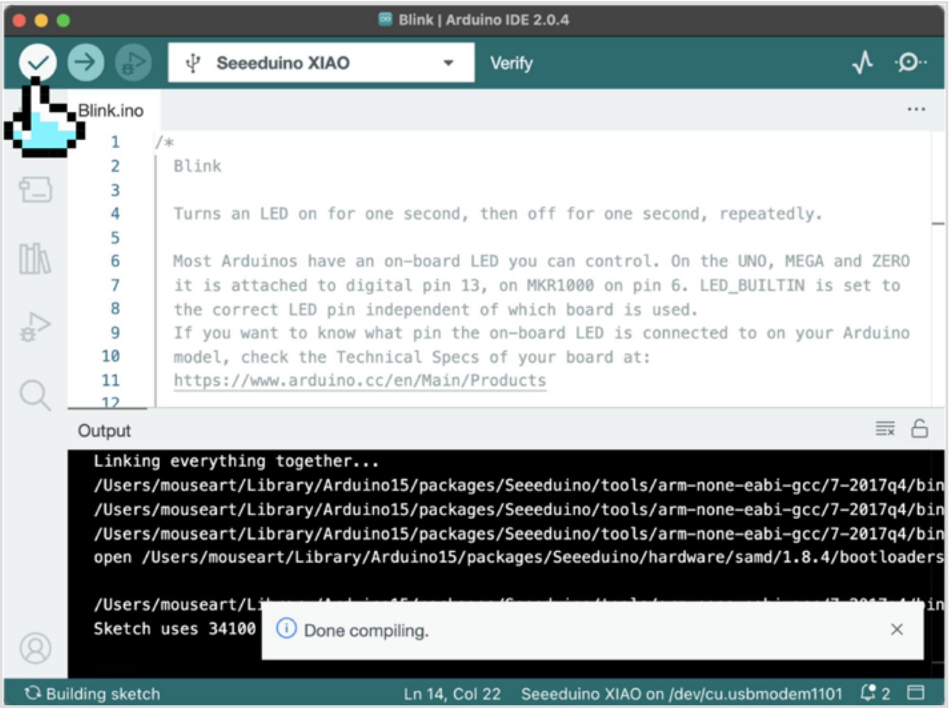
Debug: "Off"

SERCOM4: "USART"

USB Stack: "Arduino"

Ln 1, Col 1 Seeeduino XIAO on /dev/cu.usbmodem1101

Next, we can upload the program. Before uploading, we can click the ✓ verify-button.png(verify button) to verify whether the program is correct. If "Compilation Completed" is displayed, the program is correct.



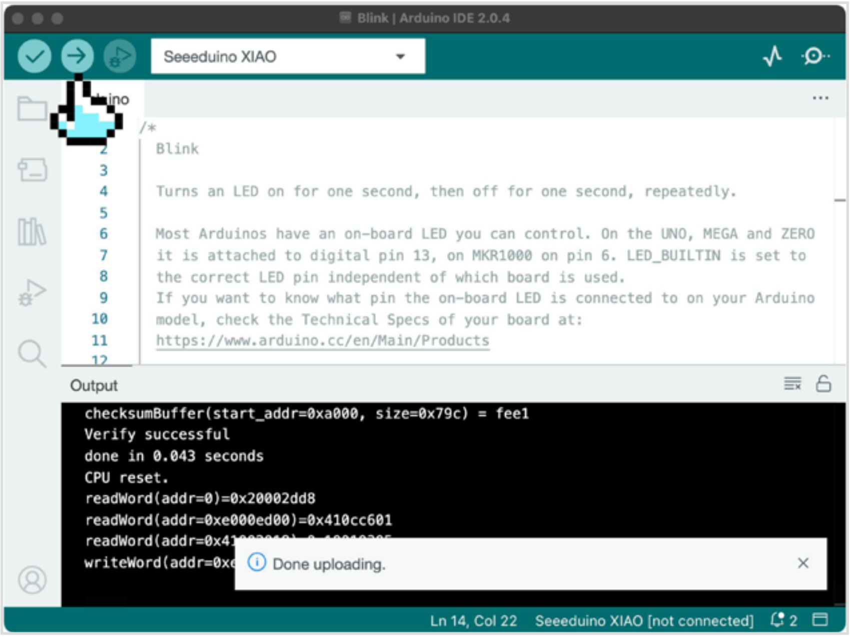
```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products
```

```
Linking everything together...
/Users/mouseart/Library/Arduino15/packages/Seeeduino/tools/arm-none-eabi-gcc/7-2017q4/bin
/Users/mouseart/Library/Arduino15/packages/Seeeduino/tools/arm-none-eabi-gcc/7-2017q4/bin
/Users/mouseart/Library/Arduino15/packages/Seeeduino/tools/arm-none-eabi-gcc/7-2017q4/bin
open /Users/mouseart/Library/Arduino15/packages/Seeeduino/hardware/samd/1.8.4/bootloaders

/Users/mouseart/Li
Sketch uses 34100
```

ⓘ Done compiling.

Building sketch   Ln 14, Col 22   Seeeduino XIAO on /dev/cu.usbmodem1101

Click the → upload-button.png (upload button), the debug window will display "Compiling Project→Upload". When "Upload Completed" is displayed, you can see the effect of the program running on XIAO, as shown in the upload successful prompt window displayed on a Mac computer.



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products
```

```
checksumBuffer(start_addr=0xa000, size=0x79c) = fee1
Verify successful
done in 0.043 seconds
CPU reset.
readWord(addr=0)=0x20002dd8
readWord(addr=0xe000ed00)=0x410cc601
readWord(addr=0x4
writeWord(addr=0xe
```

Done uploading.

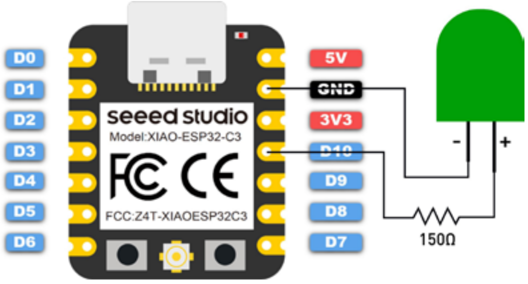
Ln 14, Col 22   Seeeduino XIAO [not connected]   2

- Attention -

When you start writing code, you will often forget the rules of uppercase and lowercase, punctuation, and make mistakes. Therefore, try to write code by yourself instead of copying and pasting. After the example program is successfully uploaded, try to create a new Sketch and start manually inputting the code.

1.1.4 Task 2: Complete the Blink example by connecting an external LED to Seeed XIAO ESP32C3 without LED

If the XIAO you have on hand is Seeed XIAO ESP32C3, since it does not have an onboard LED available for users, in order to run the Blink program, you need to first connect an LED to the D10 pin of the board, as shown below:



- Attention -

You must connect a resistor (about 150Ω) in series with the LED to limit the current flowing through the LED to prevent the strong current from burning the LED.

Then copy the following program to the Arduino IDE:

```
// Define the LED pin according to the pin diagram
int led = D10;

void setup() {
    // Initialize the digital pin 'led' as output
    pinMode(led, OUTPUT);
}

void loop() {
    digitalWrite(led, HIGH);  // Turn the LED on
    delay(1000);              // Wait for a second
    digitalWrite(led, LOW);    // Turn the LED off
    delay(1000);               // Wait for a second
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/code/L1_Blinks_XIAO_ESP32C3/L1_Blinks_XIAO_ESP32C3.ino

Code Analysis

`int led = D10;`

Seeed XIAO ESP32C3 does not have an onboard LED, so we did not preset an LED corresponding pin in the Arduino core. Just now, we connected the LED to the `D10` pin, so we need to declare it in the program.

`pinMode(led, OUTPUT);`

We defined `led` as `D10`, and this step is to initialize led(D10) as an output pin.

1.1.5 Extended Exercise

Rewrite the Blink program: In the example program, the LED is on and off for 1 second each time, so it seems to blink evenly. Try adjusting the waiting time to give the LED different blinking effects.

Hint:

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/code/L1_ll_Blinks_1_en/L1_ll_Blinks_1_en.ino

For XIAO ESP32C3, we also need to modify the pin definition part of the program:

```
int led = D10;
void setup() {
    pinMode(led, OUTPUT);
}
void loop() {
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/code/L1_ll_blinks_2_en/L1_ll_blinks_2_en.ino

1.2
Using the Button Switch on the XIAO Expansion Board to Control an LED Light

In the previous section, we learned how to control an LED light to blink using only the Seeed Studio XIAO and the onboard LED light. However, there was no interaction with the external environment, such as controlling the LED light through light or sound. In this section, we will introduce a simple sensor - the button switch, to form an automatic control system of sensor-controller-actuator. Before starting the task, we need to learn some basic knowledge, like what variables are and the common program structures, so that we can better understand and run the program.

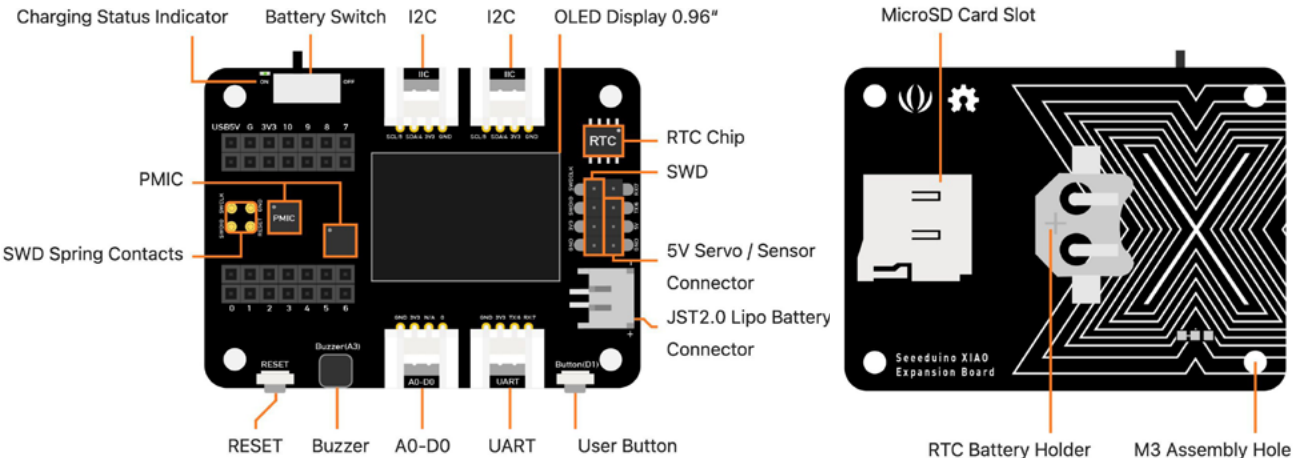
1.2.1 Background Knowledge

In the last section, we only used the onboard LED light of the Seeed Studio XIAO without connecting other modules. It could take quite some effort for beginners to use Dupont wires to connect external sensors to a board the size of a thumb and also involve a breadboard. Is there a simpler method?

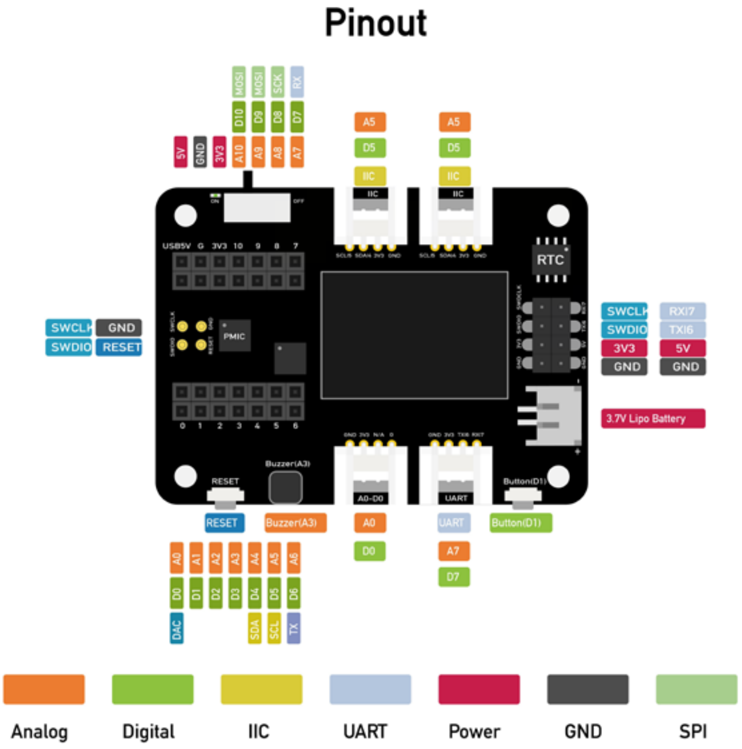
Seeed Studio XIAO Expansion Board

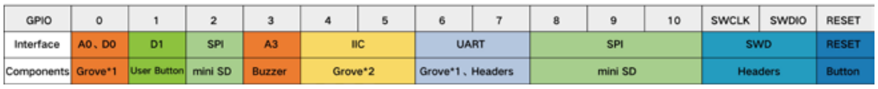
The Seeed Studio XIAO Expansion Board, only half the size of Raspberry Pi 4, is powerful and can quickly and easily build prototypes and projects. The board has a variety of peripherals such as OLED, RTC, expandable memory, passive buzzer, RESET/User button, 5V servo/sensor connector, various data interfaces... You can explore the infinite possibilities of Seeed Studio XIAO. The board also supports CircuitPython.

All models in the Seeed Studio XIAO series have uniform specifications and support the Seeed Studio XIAO Grove Shield and Seeed Studio XIAO Expansion Board. The series includes XIAO SAMD21, XIAO RP2040, XIAO nRF52840, XIAO nRF52840 Sense, XIAO ESP32C3 and XIAO ESP32S3. The front and back function interfaces of the XIAO expansion board are shown in the following figure:



To make it easier and quicker to build projects with Seeed Studio XIAO, we equipped it with a powerful expansion board. This board has a wealth of onboard peripherals and can quickly connect to more electronic modules to implement various functions. The expansion board brings out all the pins of XIAO, as shown in the pin diagram below:

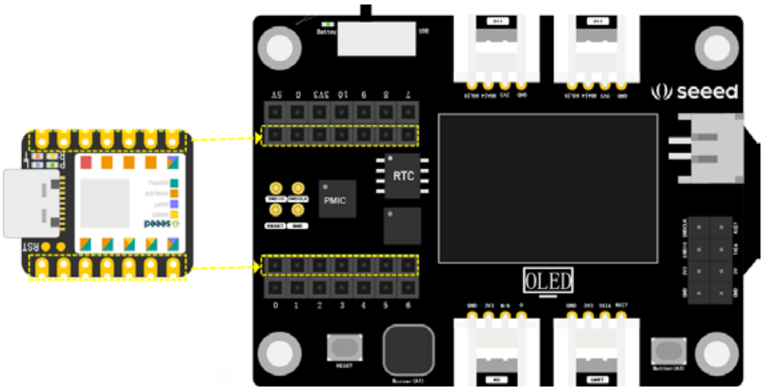




GPIO	0	1	2	3	4	5	6	7	8	9	10	SWCLK	SWDIO	RESET
Interface	A0、D0	D1	SPI	A3	IIC		UART		SPI			SWD		RESET
Components	Grove*1	User Button	mini SD	Buzzer	Grove*2		Grove*1、Headers		mini SD			Headers		Button

In most cases, the XIAO expansion board is suitable for all Seeed Studio XIAO series products.

When we need to use the XIAO expansion board, we need to connect the XIAO development board to the corresponding position on the expansion board, as shown in the figure below. Connect the pin headers on the XIAO main board to the position circled in yellow on the expansion board. Be sure to align it before pressing down to avoid damaging the pins. After that, we can start working on projects in combination with the expansion board.

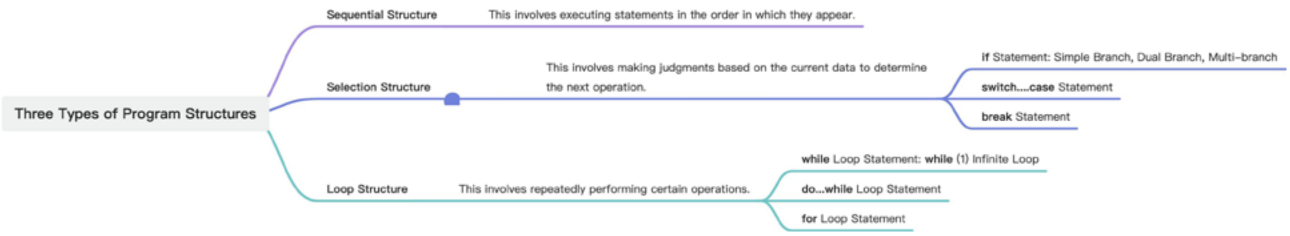


- Attention -

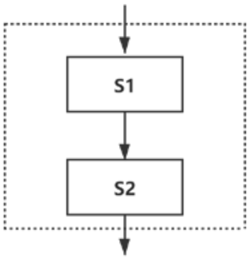
Please first plug the Seeed Studio XIAO into the two female headers on the expansion board, and then plug in the Type-C power supply, otherwise it will damage the Seeed Studio XIAO and the expansion board.

Three Basic Structures of Programs

The three basic structures of programs are sequential structure, selection structure, and loop structure.


Three Types of Program Structures
Sequential Structure
This involves executing statements in the order in which they appear.
Selection Structure
This involves making judgments based on the current data to determine the next operation.
if Statement: Simple Branch, Dual Branch, Multi-branch
switch....case Statement
break Statement
Loop Structure
This involves repeatedly performing certain operations.
while Loop Statement: while (1) Infinite Loop
do...while Loop Statement
for Loop Statement

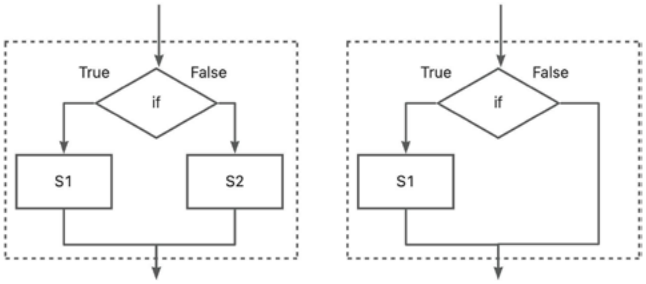
Sequential Structure



As the name suggests, the program in a sequential structure is executed in the order of the statements. It is the most basic and simple program structure. As shown in the figure below, the program will first execute the operation in the S1 box, then the operation in the S2 box, and so on.

Selection Structure

In a program,sometimes we need to make judgments based on the situation to decide the next step. For instance, the program might need to judge the light value in the current environment. If the light value is high, indicating a bright environment, there's no need to light up the light. If the light value is low, indicating a dim environment, then it's necessary to turn on the light. In such cases, we use a selection structure.



As shown in the following figures, the selection structure will judge whether the condition is fulfilled. If "True", it executes S1; if "False", it executes S2; or if "True", it executes S1, if "False", it exits the selection structure.

The if Statement

The if statement is the most common selection structure, which executes the following statement when the given expression is true. The if statement has three structural forms as shown in the following example. Simple branch structure: Execute when the condition is fulfilled.

```
if (expression) {
  statement;
}
```

Dual branch structure: Execute statement1 when the condition is fulfilled, otherwise execute statement2.

```
if (expression) {
 statement1;
}
else {
  statement2;
}
```

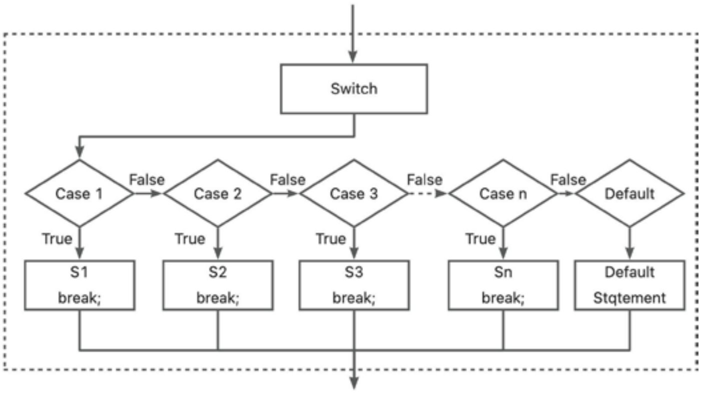
Multi-branch structure: Use nested if statements to judge different situations.

```
if (expression1) {
  statement1;
}
else if (expression2) {
  statement2;
}
else if (expression3) {
  statement3;
}
```

`switch……case` Statement

When dealing with multiple selection branches, using an “if……else” structure to write a program can be quite lengthy. In this case, it’s much more convenient to use a `switch` statement. The `switch` structure compares the expression in parentheses with the constants after `case`. If they match, it executes the corresponding statement and exits the structure via a `break` statement. If none match, it runs the statement after `default`. It's important to note that the expression in parentheses after `switch` must be of integer or character type.

```
switch (expression) {
  case constant_expression1:
    statement1;
    break;
  case constant_expression2:
    statement2;
    break;
    ……
  default:
    statementn;
    break;
}
```



`break` Statement

The `break` statement can only be used in a `switch` multi-branch selection structure and loop structures. It is used to terminate the current program structure, allowing the program to jump to subsequent statements for execution.

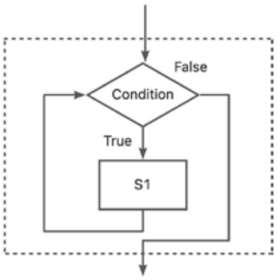
Loop Structure

A loop structure is used when a part of the program needs to be executed repeatedly, based on given judgment conditions to determine whether to continue executing a certain operation or exit the loop. There are three common types of loop statements:

`while` Loop

The while `loop` is a type of “when” loop that executes the statements in the loop body when a certain condition is met.

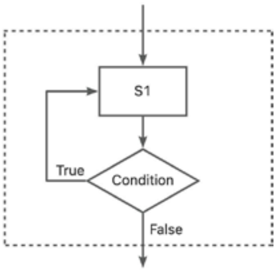
```
while (expression) {
  statement;
}
```



`do……while` Loop

This is a type of "until" loop. The statement in the loop body is executed once before the expression is evaluated. If the expression is true, the loop continues.

```
do {
  statement;
} while (expression);
```

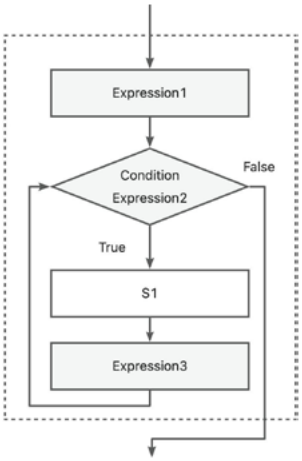


`for` Loop

This includes three expressions: Expression1 for initialization, Expression2 for judgment, and Expression3 for increment.

```
for (Expression1; Expression2; Expression3) {
  statement;
}
```

In addition to the above loop statements, there are control statements, `break` and `continue`, in the loop structure used to prematurely end the loop or exit the loop. In this lesson, we just need to understand these program structures. In later courses, we will gradually master them through project examples.



1.2.2 Task 1: Control the LED on the XIAO using the button on the XIAO expansion board

Analysis

The effect we want to achieve is that when the button is pressed, the LED lights up; when the button is released, the LED goes off. The program is written in three steps:

- Define pins and create variables.
- Initialize and set pin status.
- Read the button status, implement condition judgment. If the button is pressed, the light is on, otherwise, the light is off.

- Variable -

In a program, a value that can change is called a variable. For example, defining an integer variable `i` *as* `int i;`*. We can assign a value to the variable at the same time as we define it, such as* `int i =0;`*. Furthermore, depending on the data type, different statements are used to define variables, such as defining a floating point number,* `float x = 1.9;`*, and so on. For more details, refer to the Arduino data types and constants documentation https://www.arduino.cc/reference/en/#variables.*

Writing the Program

Step 1: Define pins and create variables. The on-board button switch on the XIAO expansion board is D1, so we define it as pin 1 and set a variable for the button status. Note that LED_BUILTIN will set the LED to the correct pin, so we don't need to manually define it:

```
const int buttonPin = 1;  // The on-board button switch on the XIAO expansion board
is D1, which we define as pin 1
// If you are using XIAO RP2040, please change 1 to D1
int buttonState = 0;  // buttonState is a variable to store the button status
```

Step 2: Set pin status. Set the LED pin to output status and the button pin to input pull-up status. Use INPUT_PULLUP to enable internal pull-up resistors. When the button is not pressed, it returns 1 or HIGH (high level). When the button is pressed, it returns 0 or LOW (low level).

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT);// Set the LED pin to output status
    pinMode(buttonPin, INPUT_PULLUP);// Set the button pin to input status
}
```

Step 3: Continuously read the button status. If the button is pressed, the light is on, otherwise, the light is off. Because the on-board LED of the XIAO is negative logic, when the button is pressed and returns 0, the LED is on; when it returns 1, the LED is off.

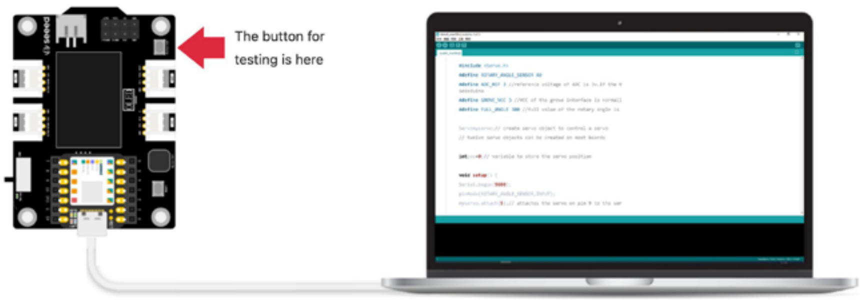
```
void loop() {
    // Read the button status and store it in the buttonState variable
    buttonState = digitalRead(buttonPin);
    // Check whether the button is pressed, if the button is pressed
    if (buttonState == HIGH) {
        // Turn on the LED:
        digitalWrite(LED_BUILTIN, HIGH);
    }
    else {
        // Turn off the LED:
        digitalWrite(LED_BUILTIN, LOW);
    }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/code/L2_Button_XIAO_en/L2_Button_XIAO_en.ino

Uploading the Program

We upload the program we wrote to the hardware. First, use the data cable in the kit to connect the XIAO to the computer.

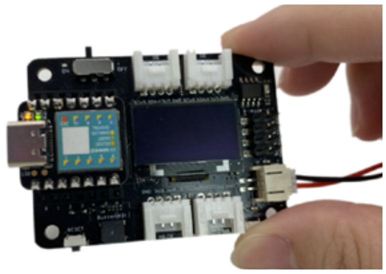
Note the position of the buttons on the XIAO extensions used for testing in the figure.



Then click the verify button ✓ to verify the program. If it is correct, click the upload button → to upload the program to the hardware. When the debugging area displays “Done uploading.”, we can press the button to see if the LED lights up.

- Attention -

There are two identical buttons on the expansion board. One is the RESET button near the Type-C interface, and the other is the user-defined button near the lithium battery interface. Test with the one near the lithium battery interface.

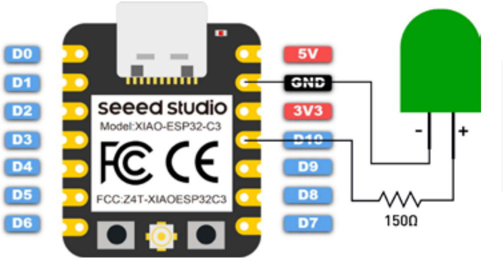


1.2.3 Task 2: Use the button on the XIAO expansion board to control the external LED on the XIAO ESP32C3

For the Seeed XIAO ESP32C3, it doesn't have an on-board LED for users to use. To run the Blink program, you need to first connect an LED to the `D10` pin of the board as shown:

- Attention -

Be sure to add a resistor (about 150Ω) in series with the LED to limit the current flowing through the LED to prevent overcurrent from burning out the LED.



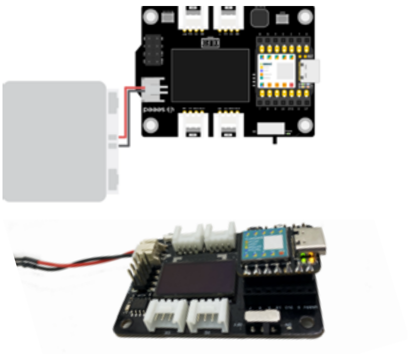
Then copy the following program into the Arduino IDE:

~~~
/*
 * Button controlling external LED of XIA

Apologies for the confusion. It seems that there was an issue with quoting text from
the document. Let's continue:

#### Task 2: Use the button on the XIAO expansion board to control the external LED
on the XIAO ESP32C3
For the Seeed XIAO ESP32C3, it doesn't have an on-board LED for users to use. To ex-
ecute the Blink program, you first need to connect an LED to the board's `D10` pin as
shown.

> ⚠ Note: Make sure to add a resistor (about 150Ω) in series with the LED to limit
the current flowing through the LED and prevent overcurrent from burning out the LED.

Then, copy the following program into the Arduino IDE:
```cpp
/*
 * Button controlling external LED of XIAO ESP32C3
 */

const int buttonPin = 1;     // The pin number of the button
int buttonState = 0;     // Variable for reading the button status
int led = D10;  // Pin number of the LED

void setup() {
  // Initialize the LED pin as an output:
  pinMode(led, OUTPUT);
  // Initialize the button pin as an input:
  pinMode(buttonPin, INPUT_PULLUP);
}
~~~

```
  void loop() {
    // Read the state of the button:
    buttonState = digitalRead(buttonPin);
    // Check if the button is pressed. If it is, the button state is HIGH
    if (buttonState == HIGH) {
      // Turn the LED on:
      digitalWrite(led, HIGH);
    }
    else {
      // Turn the LED off:
      digitalWrite(led, LOW);
    }
  }
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L2_Button_XIAO_ESP32C3_en

Powering XIAO with an external battery

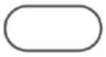
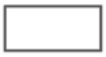
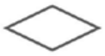
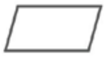
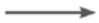


When demonstrating the effect, in addition to using a data cable to power the computer, you can also use an external lithium battery. This makes it convenient to move and do projects, as shown in the picture.

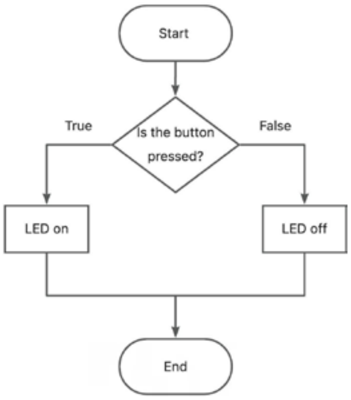
Expanded Exercise

Flow Chart

Before writing the program, you can first draw a flow chart of the program to help organize your thoughts. The common flow chart symbols are as follows:

	Start/End box: Indicates the start or end of a program.
	Process box: Indicates the execution or handling of certain tasks.
	Decision box: Represents the judgment of a certain condition.
	Input/Output box: Represents the input of data or the output of results.
	Connection line: Represents the direction of the process flow.

The button-controlled LED program we implemented in this section is represented by the following flow chart. You can try drawing it yourself.



1.3 Transforming XIAO and its Expansion Board into a Morse Code Transmitter

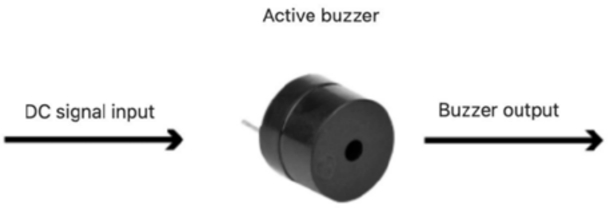
Everyone knows that "SOS" is an internationally recognized emergency signal, a form of Morse code. Today, we will transform Seeed Studio's XIAO into a Morse code transmitter. We will try to make the onboard buzzer of the expansion board send signals automatically. In addition, we will learn how to control the buzzer manually with a button.

1.3.1 Background Knowledge

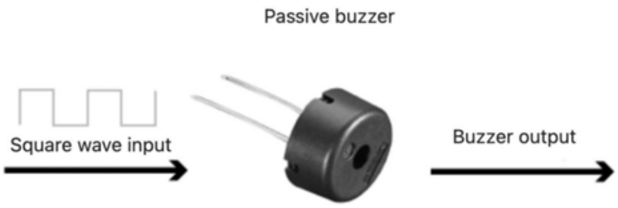
Buzzer

A buzzer is an integrated electronic sound device that generates sound based on the input of an electrical signal. Buzzers are often installed on electronic products for sound generation. There are two types of buzzers: active (source buzzers) and passive (sourceless buzzers).

Active Buzzers: These buzzers have a simple oscillation circuit inside. When connected to a DC power supply, the buzzer can convert a constant DC into a certain frequency pulse signal, thereby driving the internal aluminum sheet to vibrate and make a sound. Active buzzers can usually only emit some fixed-pitch (frequency) sounds and are widely used in the sound devices of computers, printers, copiers, alarms, electronic toys, car electronics, phones, timers, and other electronic products.

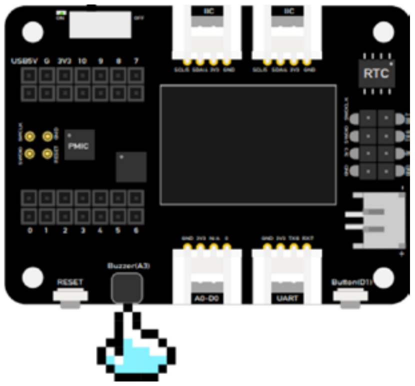


Passive Buzzers: These buzzers work similarly to loudspeakers. They don't have an internal oscillator and need to be connected to a changing current signal to work. They usually use different frequency square wave signals for driving. The sound generated by passive buzzers will change according to the change in input signal, and they can output a variety of sounds like speakers, not just emitting a fixed single tone (frequency).



The standalone buzzer module is shown in the figure below.

In the Seeed Studio XIAO expansion board, there is an onboard passive buzzer connected to pin A3. We can output PWM pulse signals to this pin to control the buzzer.



`tone()` and `noTone()` Functions

`tone()` Function

The `tone()` function can generate a fixed frequency PWM signal to drive a passive buzzer to make a sound, and it can define the frequency and duration of the sound.

Syntax: `tone(pin, frequency);` `tone(pin, frequency, duration);`

Parameters:

`pin:` The pin to which the buzzer is connected (in the Seeed Studio XIAO expansion board, it's A3).

`frequency:` The frequency of the sound (unit: Hz), the type allowed is unsigned integer.

`duration:` The duration of the sound (unit: milliseconds, this parameter is optional), the type allowed is unsigned long.

`noTone()` Function

This function is used to stop the sound of the buzzer controlled by the `tone()` function. If there is no sound generated, the function is invalid.

Syntax: `noTone(pin);`

Parameters:

`pin:` The pin to stop the sound.

Common Operators

In previous studies, we have used some operators. Next, we will learn about common types of operators and their usage methods.

Arithmetic Operators:

Operator	Explanation
=	Assignment operator
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Modulus operator

Comparison Operators:

Operator	Explanation
!=	Not equal to
<	Less than
<=	Less than or equal to
==	Equal to
>	Greater than
>=	Greater than or equal to

Boolean Operators:

Operator	Explanation
&&	Logical "and"
!	Logical "not"
||	Logical "or"

Compound Operators:

Operator	Explanation
++	Self-increment
+=	Compound addition
--	Self-decrement
-=	Compound subtraction

For detailed explanations, see: https://www.arduino.cc/reference/en/

Morse Code

Source of the picture:

https://en.wikipedia.org/wiki/Samuel_Morse

Morse code is a method of expressing information in telecommunication, named after the inventor of the telegraph, Samuel Morse.

The international Morse code encodes the 26 English letters A to Z, some non-English letters, Arabic numbers, and a small number of punctuation marks and prosigns. There is no distinction between upper and lower case letters. Each Morse code symbol consists of a series of dots (·) and dashes (—). The duration of a dot is the basic unit of time measurement in Morse code transmission. The duration of a dash is three times the duration of a dot. After each dot or dash in a character, there is a time when the signal is absent, called a space, equal to the duration of a dot. For example, the standard emergency distress signal SOS is expressed in Morse code as shown in the figure below.

● ● ● ▬ ▬ ▬ ● ● ●

If it's expressed in sound, it sounds like this.

https://files.seeedstudio.com/wiki/XIAO_Big_Power-Board-ebook-photo/chapter_1-3/sos.wav

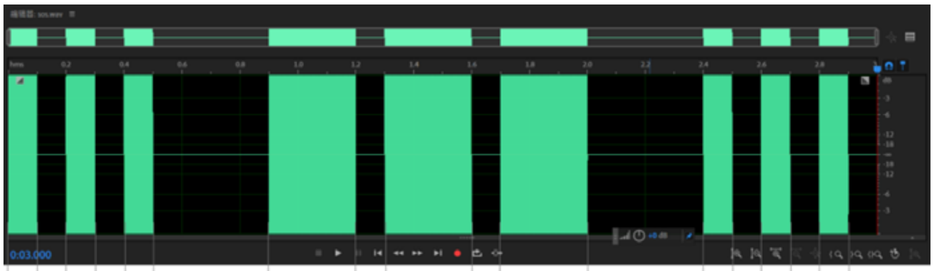
1.3.2 Task 1: Automatic Broadcasting of "SOS"

Analysis

Automatic broadcasting means that when the control board is started, the onboard buzzer automatically emits the Morse code of “SOS”. The program is written in three steps:

- Define the buzzer pin
- Initialization, setting the state of the buzzer pin
- Loop the buzzer to play the Morse code of "SOS"

Let’s first look at how to reflect the Morse code of “SOS” through the program. If you import the audio file of Morse code into the audio editing software, you can see the waveform of the sound and the duration of each syllable, which is generally divided into long and short sounds. To facilitate understanding and programming, we use a binary way to mark the switch of the buzzer, 1 indicates the buzzer is on, 0 indicates the buzzer is off, and the gray number represents how long the current status needs to last. After a Morse code ends, because it needs to be looped, you need to leave time between the two Morse codes, here it is set to 0.8 seconds.



1 0.1		1 0.1		1 0.1		1 0.3		1 0.3		1 0.3		1 0.1		1 0.1		1 0.1	
	0 0.1		0 0.1		0 0.4		0 0.1		0 0.1		0 0.4		0 0.1		0 0.1		0 0.8

To emit a short sound, which corresponds to a dot in Morse code, from the buzzer, you can use the following code in Arduino:

```
tone(pinBuzzer, 200);
delay(100);
noTone(pinBuzzer);
delay(100);
```

In this code snippet:

- `tone(pinBuzzer, 200)` generates a sound at a frequency of 200 Hz on the buzzer connected to the `pinBuzzer` pin.
- `delay(100)` waits for 100 milliseconds. This is how long the sound lasts.
- `noTone(pinBuzzer)` stops the sound on the buzzer.
- The final `delay(100)` ensures there's a pause before the next sound is generated, representing the space between the signals.

The code you provided is a complete Arduino program for emitting the SOS Morse code signal using a buzzer. Here is the English explanation:

Writing the Program

Step 1: Define pins and create variables

```
int pinBuzzer = 3; // Define the buzzer on pin 3, if you're using XIAO RP2040/XIAO
ESP32, change 3 to A3
```

Step 2: Set pin state

```
void setup() {
    pinMode(pinBuzzer, OUTPUT); // Set the buzzer pin to output state
}
```

Step 3: Loop to play "SOS" Morse code

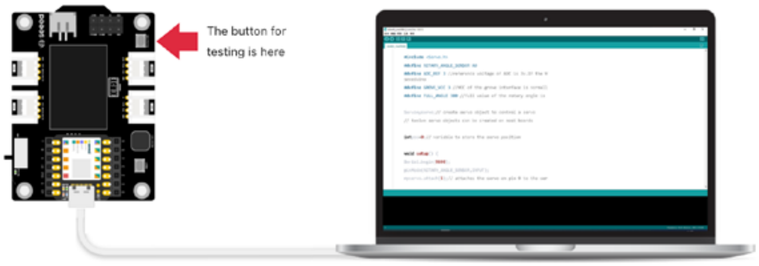
```
void loop() {
    // Emit three short signals:
    for(int i = 0; i < 3; i++){
        tone(pinBuzzer, 200);
        delay(100);
        noTone(pinBuzzer);
        delay(100);
    }
    delay(200);

    // Emit three long signals:
    for(int i = 0; i < 3; i++){
        tone(pinBuzzer, 200);
        delay(300);
        noTone(pinBuzzer);
        delay(100);
    }
    delay(200);

    // Emit three short signals again:
    for(int i = 0; i < 3; i++){
        tone(pinBuzzer, 200);
        delay(100);
        noTone(pinBuzzer);
        delay(100);
    }
    delay(800); // Wait before repeating
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L3_SOS_XIAO_en

Uploading the Program

To upload the program to your hardware, connect your XIAO to your computer using the data cable included in the kit. After this, click on the verify button ✓ to check your program. If it passes verification, click on the upload button → to upload the program to your hardware. When the debug area shows "Done uploading.", you can listen to the Morse code sound. Is it the rhythm you expected?

Note the position of the buttons on the XIAO extensions used for testing in the figure.



1.3.3 Task 2: Control the buzzer with a button

Controlling the buzzer with a button to emit Morse code can be done manually. The code logic is simple: use an if statement to determine if the button is pressed. If it is, then the buzzer emits a sound.

Analysis

The program is also written in three steps:

- Define the buzzer and button pins.
- Initialize by setting the state of the buzzer and button pins.
- Determine whether the button is pressed; if pressed, emit a sound.

Write the program

Step 1: Define the buzzer and button pins

```
const int buttonPin = 1; // The button is on pin 1, if you are using XIAO RP2040/
XIAO ESP32, please change 1 to D1
int pinBuzzer = 3;// The buzzer is on pin 3, if you are using XIAO RP2040/XIAO
ESP32, please change 3 to A3
```

Step 2: Set the button and buzzer pin states

```
void setup() {
    // Set the buzzer pin as output:
    pinMode(pinBuzzer, OUTPUT);
    // Set the button pin as input:
```

```
    pinMode(buttonPin, INPUT_PULLUP);
}
```

Step 3: Check the button state, if the button is pressed, the buzzer sounds. Here, the `tone()` function is used to control the passive buzzer to make a sound.

```
void loop() {
    // buttonState is the button variable, read the button state and store it in the
variable:
    int buttonState = digitalRead(buttonPin);

    // Check if the button is pressed, if the button is pressed:
    if (buttonState == LOW) {
         // The buzzer sounds with a frequency of 200, for a duration of 200 milli-
seconds
        tone(pinBuzzer, 200, 200);
    }
}
```

- Attention -

There are two identical buttons on the expansion board, one is the RESET button, which is closer to the Type-C interface, and the other is the user-defined button, which is closer to the lithium battery interface. When testing, press the one closer to the lithium battery interface.

The complete program is as follows:

```
/*
 * Button-SOS
 */
const int buttonPin = 1; // The button is on pin 1, if you are using XIAO RP2040/
XIAO ESP32, please change 1 to D1!
int pinBuzzer = 3; // The buzzer is on pin 3, if you are using XIAO RP2040/XIAO
ESP32, please change 3 to A3!
void setup() {
  // Set the buzzer pin as output:
  pinMode(pinBuzzer, OUTPUT);
  // Set the button pin as input:
  pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  // buttonState is the button variable, read the button state and store it in the
variable:
  int buttonState = digitalRead(buttonPin);

  // Check if the button is pressed, if the button is pressed:
  if (buttonState == LOW) {
    // The buzzer sounds with a frequency of 200, for a duration of 200 milliseconds
    tone(pinBuzzer, 200, 200);
  }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L3_ButtonSOS_XIAO_en

Uploading the program

We will upload the written program to the hardware. First, connect the XIAO to your computer using the data cable from the kit.

Next, click ✓ (verify button) to validate the program. If there are no errors, click → (upload button) to upload the program to the hardware. When the debug area shows "Done uploading.", we can press the button on the XIAO expansion board and test whether the buzzer will sound.

1.3.4 Extended Exercise

The passive buzzer can emit different pitches to form a simple melody. Research how to make Arduino play notes through a search engine. You can open the extended exercise code to experience the effect of playing "Happy Birthday" with the buzzer.

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L3_HappyBirthday_en

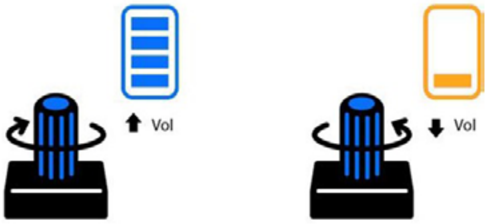
1.4
Monitor Knob Value Changes with Serial Monitor

When we write a few lines of code to control the board to light up the LED, or to use a button switch to control the buzzer, we can intuitively see the working state of these external hardware. If it achieves our expected results, it is very fortunate. What if it doesn't? The program compiles without error, where is the mistake? It would be nice if they could speak up. In this section, we will learn how to communicate with the computer through the serial monitor and check the running status and information of the program and hardware.

1.4.1 Background Knowledge

Rotary Potentiometer

The rotary potentiometer, although it doesn't seem common, has a very wide range of uses in household appliances and industrial equipment. For example, the volume knob on the sound system.



The rotary potentiometer can produce an analog output value between 0 and VCC (the voltage of the connected circuit) on its connected pins. By rotating the knob, you can change the output voltage value. The range of the knob's angle is 300°, and the output value is 0-1023. We can use the rotary potentiometer to control the LED light to show brightness changes, or control the servo to rotate at different angles, etc.

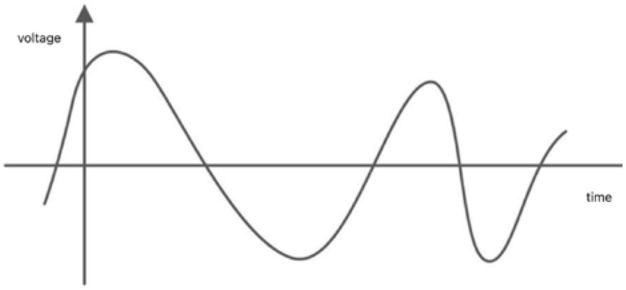
Analog I/O

In the Arduino series of development boards, the pins with "A" in front of the pin number are analog input pins. We can read the analog value on these pins to achieve the effect we want.

Analog Signal

In life, analog signals are everywhere, such as the change in sound, light, temperature, etc., the frequency, amplitude, etc. of the signal can change continuously with time.

So how do we read the analog value of the pin through the development board? The analog input pin has an ADC (analog-to-digital converter), which can convert the external input analog signal into a digital signal that the development board can recognize, thereby achieving the function of reading in analog values, i.e., it can convert a 0-5V voltage signal into an integer value of 0-1023.



- `analogRead();`

Read the value from the specified analog pin.

Syntax: `analogRead(pin);`

Parameters:

`pin:` The name of the analog input pin to be read.

- `analogWrite();`

Corresponding to analog input is analog output. We use the analogWrite() function to achieve this function. It should be noted that when using this function, it is only through a special way to output different voltages to achieve the effect of approximate analog values. This method is called PWM pulse width modulation, so we are writing PWM square waves to the specified pin, not the true analog value.

Syntax: `analogWrite(pin, value);`

Parameters:

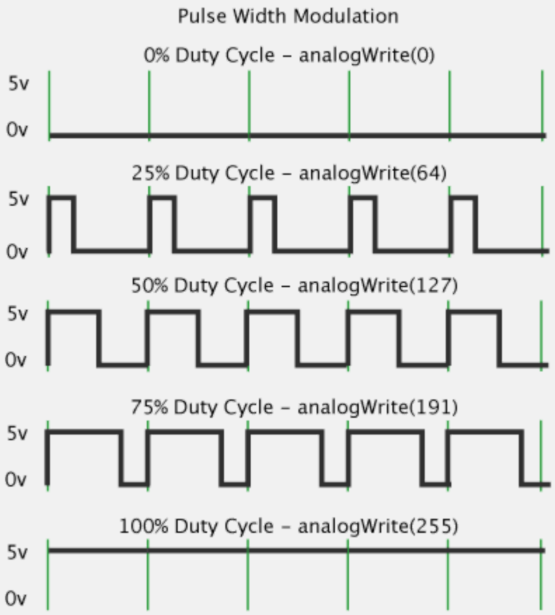
`pin:` The pin to output PWM, allowed data type: int.

`value:` Duty cycle, between 0-255, allowed data type: int.

- PWM Pulse Width Modulation -

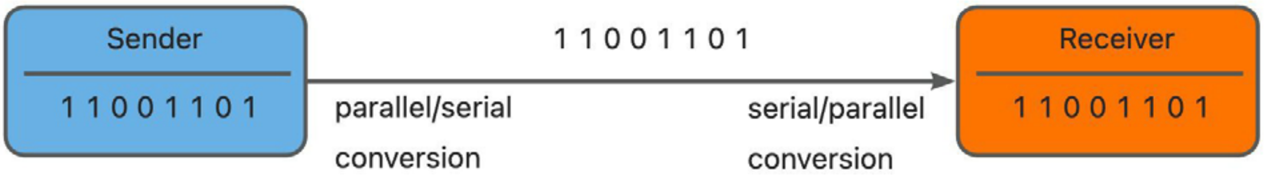
Pulse width modulation (PWM) is a way to achieve analog results through digital output. Simply put, you can control the charging current by adjusting the period of PWM and the duty cycle of PWM. As shown in the figure, the voltage is switched back and forth between 0V (low level) and 5V (high level). A switchback is a period. In this period, if the time of high voltage is 25% and the time of low voltage is 75%, the duty cycle is 25%, and the output voltage is 5V.

When we write a few lines of code to control the lighting of LEDs on the development board, or use button switches to control the buzzer, we can directly observe the working status of these external hardware. If it achieves our expected results, we are lucky. But what if it doesn't? The program compiles without errors, so where is the problem? It would be nice if they could talk. In this section, we will learn how to communicate with the computer, monitor the running status and information of the program and hardware through the serial monitor.



Serial Communication

When we want to communicate with other devices using XIAO, the most common method is serial communication. All Arduino series development boards have this functionality. As we know, computers understand binary data (like 1010). Therefore, among electronic devices, serial communication achieves its function by sending and receiving such data. The key component to implement this function is the USART (Universal Synchronous/Asynchronous Receiver Transmitter). In the Arduino IDE, we can observe the sent and received data through the Serial Monitor, and we need related serial communication functions to implement this feature.



- `**Serial.begin();**`

This function is used to open the serial port and set the data transmission rate.

Syntax: `Serial.begin(speed);`

Parameters:

`Serial`: Serial port object.

`speed`: Baud rate, commonly set to values like 9600, 115200, etc.

- `**Serial.println();**`

Syntax: `Serial.println(val);`

Parameters:

`Serial`: Serial port object.

`val`: The value to be printed, which can be of any data type.

For example, to print “hello world!!!” to the Serial Monitor, we need to initialize the serial port in the `setup()` function and output “hello world!!!” through the serial port in the `loop()` function:

```
void setup() {
    Serial.begin(9600); // Initialize the serial port and set the data transmission
rate to 9600
}
void loop() {
    Serial.println("hello world!!!"); // Output "hello world!!!" through the serial
port
}
```

Returning to the question at the beginning of this section: when we have written the code and verified it to be correct, but the effect of running the code exceeds expectations or the hardware doesn’t respond at all, where is the problem? At this time, we can use the Serial Monitor to observe the data sent or received by the hardware to make a judgment. For instance, we can control the on-off state of an LED with a button, and we can use the Serial Monitor to check the returned value when the button is pressed to determine whether the button is working properly. Next, we will learn how to use the Serial Monitor to make the hardware “speak”.

1.4.2 Task 1: Use the Serial Monitor to Check if the Button is Pressed

Analysis

Remember controlling the on-off state of an LED with a button? Some of the code can be reused. We only need to read the button on-off setting and button on-off state code, and then add the initialization of the serial port and the data sent to the serial port. The program writing still follows three steps:

- Define button pins and variables.
- Initialize the serial port, set the serial port baud rate, and set the status of the button on-off pin.
- Read the button state and send it to the serial port.

Write the program

Step 1: Define the button pin and variable.

```
const int buttonPin = 1; // Define the button switch as pin 1. If you are using XIAO
RP2040/XIAO ESP32, please change 1 to D1
int buttonState = 0; // Define buttonState as a variable to store the button status
```

Step 2: Initialize the serial port, set the baud rate of the serial port, and set the button switch pin status.

```
void setup() {
    pinMode(buttonPin, INPUT_PULLUP);  // Set the button pin as input
    Serial.begin(9600); // Initialize the serial port
}
```

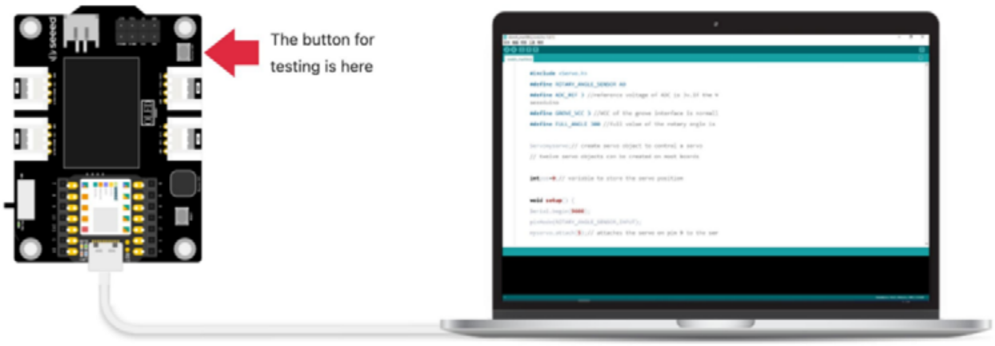
Step 3: Read the button status and send it to the serial port

```
void loop() {
    buttonState = digitalRead(buttonPin);  // Read the button status and store it in
the buttonState variable
    Serial.println(buttonState); // Send the button status data to the serial port
    delay(500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L4_ReadButton_XIAO_en

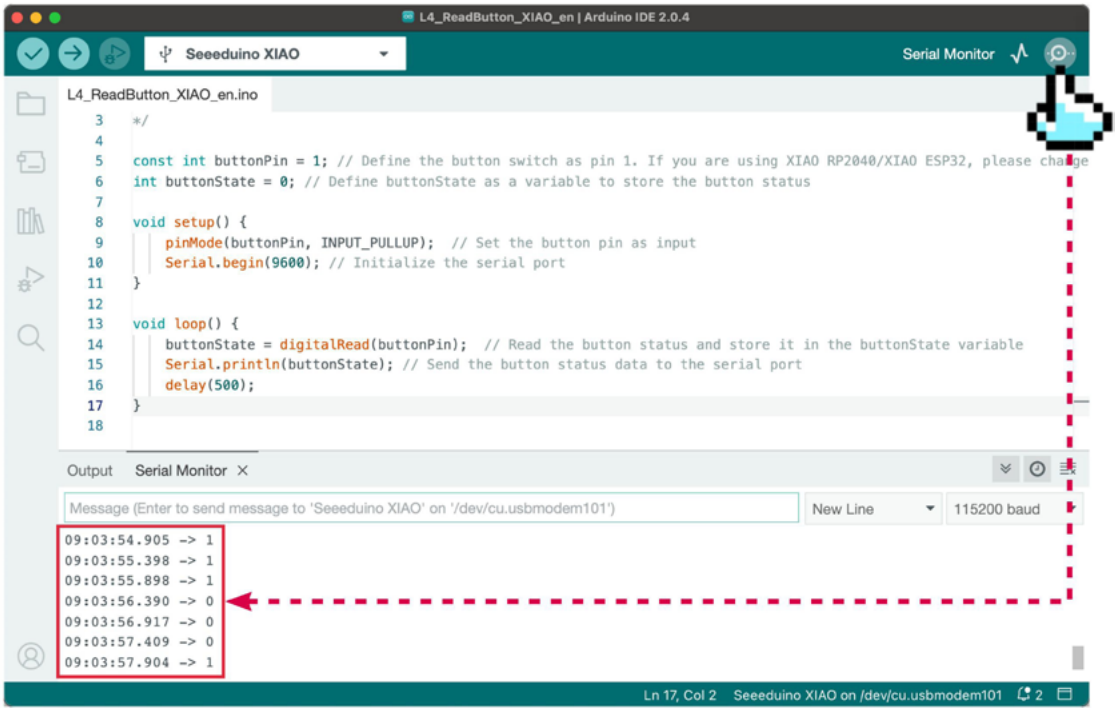
Upload the program

We will upload the written program to the hardware. First, connect the XIAO to the computer with the data cable from the kit.



Note the position of the buttons on the XIAO extensions used for testing in the figure.

Click ✓ (Verify Button) in the Arduino IDE to verify the program. If the verification is correct, click → (Upload Button) to upload the program to XIAO. When the debug area shows “Done uploading.”, open the serial monitor and observe the value changes printed by the serial monitor when the button is pressed and released. What did you find?


L4_ReadButton_XIAO_en | Arduino IDE 2.0.4
Seeeduino XIAO
Serial Monitor
L4_ReadButton_XIAO_en.ino
Output
Serial Monitor
Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 17, Col 2
Seeeduino XIAO on /dev/cu.usbmodem101
2

```
*/

const int buttonPin = 1; // Define the button switch as pin 1. If you are using XIAO RP2040/XIAO ESP32, please change
int buttonState = 0; // Define buttonState as a variable to store the button status

void setup() {
    pinMode(buttonPin, INPUT_PULLUP);  // Set the button pin as input
    Serial.begin(9600); // Initialize the serial port
}

void loop() {
    buttonState = digitalRead(buttonPin);  // Read the button status and store it in the buttonState variable
    Serial.println(buttonState); // Send the button status data to the serial port
    delay(500);
}
```

```
09:03:54.905 -> 1
09:03:55.398 -> 1
09:03:55.898 -> 1
09:03:56.390 -> 0
09:03:56.917 -> 0
09:03:57.409 -> 0
09:03:57.904 -> 1
```

When we press the button on the XIAO expansion board, the serial monitor shows `0`, and when we release the button, the serial monitor shows `1`.

1.4.3 Task 2: Using the Serial Monitor to View Knob Value Changes

Analysis

In Task 1, the button switch is a digital input that sends out digital signals 0 and 1, while the knob potentiometer returns an analog signal. We need to read the rotation angle value of the knob potentiometer on pin `A0` and send it to the serial port. The program also consists of three steps:

- Define the knob potentiometer pin and variables.
- Initialize the serial port and set the status of the knob potentiometer pin.
- Read and calculate the rotation angle value of the knob potentiometer and send it to the serial port.

Write the program

Step 1: Define the knob potentiometer pin and variables. Here we need to define the voltage value of the ADC (Analog-to-Digital Converter) and the reference voltage of the Grove module interface, because we will calculate the voltage changes in the circuit where the knob switch is connected through these voltage values.

```
#define ROTARY_ANGLE_SENSOR A0  // Define the rotary potentiometer interface A0
#define ADC_REF 3 // ADC reference voltage is 3V
#define GROVE_VCC 3 // Grove interface reference voltage is 3V
#define FULL_ANGLE 300 // The maximum rotation angle of the knob potentiometer is
300°
```

Step 2: Initialize the serial port, set the baud rate of the serial port, and set the status of the knob potentiometer pin.

```
void setup()
{
    Serial.begin(9600);//Initialize the serial port
     pinMode(ROTARY_ANGLE_SENSOR, INPUT);//Set the rotary potentiometer pin to input
state
}
```

Step 3: Read and calculate the rotational angle value of the rotary potentiometer and send it to the serial port. Here, we first need to set the data type of the voltage variable, set the analog value variable of the rotary potentiometer pin, and then calculate the real-time voltage. After calculating the real-time voltage, calculate the rotational angle value of the rotary potentiometer.

```
void loop()
{
    float voltage;    //Variable voltage is of floating-point type
    int sensorValue = analogRead(ROTARY_ANGLE_SENSOR);    //Read the analog value at
the rotary potentiometer pin
    voltage = (float)sensorValue*ADC_REF/1023;    //Calculate real-time voltage
    float degrees = (voltage*FULL_ANGLE)/GROVE_VCC;    //Calculate the rotation angle
of the knob
     Serial.println("The angle between the mark and the starting position:");     //
Print characters at the serial port
     Serial.println(degrees);     //Print the rotation angle value of the rotary po-
tentiometer at the serial port
    delay(100);
}
```

- #define Macro Definition -

`#define` is a pre-processing command used for macro definitions. In Arduino, we can use `#define` to name constants. During the compilation of the program, all occurrences of the "macro name" will be replaced with the string in the macro definition, such as `#define ledPin 5`. During compilation, 5 will replace all uses of `ledPin`. Syntax: `#define constant name constant value`. The "#" symbol is mandatory, and there is no need to use the ";" symbol at the end of the sentence.

The complete code is as follows:

```
/*
 * Use the serial monitor to view the knob potentiometer
 */
#define ROTARY_ANGLE_SENSOR A0//Define the rotary potentiometer interface A0
#define ADC_REF 3 //ADC reference voltage 3V
```

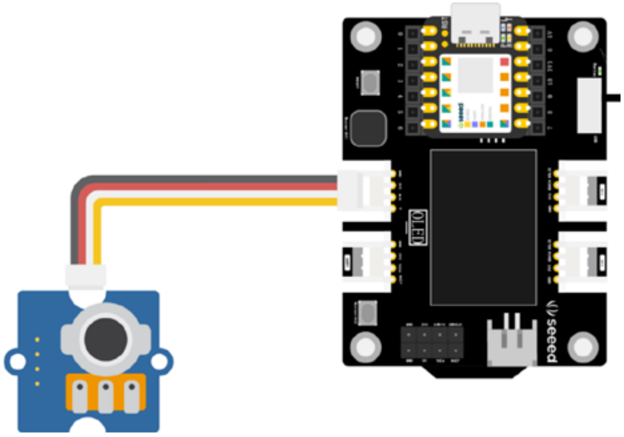
```
#define GROVE_VCC 3 //Reference voltage 3V
#define FULL_ANGLE 300 //The maximum rotation angle of the rotary potentiometer is
300°

void setup()
{
    Serial.begin(9600);//Initialize the serial port
    pinMode(ROTARY_ANGLE_SENSOR, INPUT);//Set the rotary potentiometer pin as an in-
put
}

void loop()
{
    float voltage;//Variable voltage is of floating-point type
    int sensorValue = analogRead(ROTARY_ANGLE_SENSOR);//Read the analog value at the
rotary potentiometer pin
    voltage = (float)sensorValue*ADC_REF/1023;//Calculate real-time voltage
    float degrees = (voltage*FULL_ANGLE)/GROVE_VCC;//Calculate the rotation angle of
the knob
    Serial.println("The angle between the mark and the starting position:");//Print
characters at the serial port
    Serial.println(degrees);//Print the rotation angle value of the rotary potenti-
ometer at the serial port
    delay(100);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L4_ReadRotary_XIAO_en

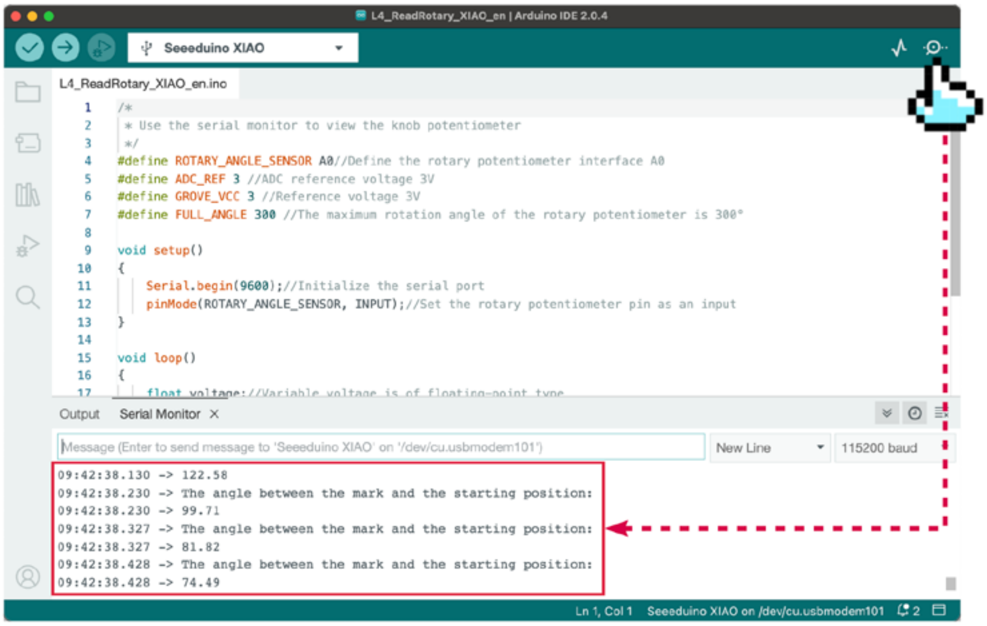
Upload the program

After writing the program, since external sensors are used, connect the knob module to the `A0` interface using the four-color Grove cable as shown in the image below:

After connecting, connect the XIAO main control board to your computer using a data cable.



In the Arduino IDE, click on the verification button ✓ to verify the program. If it verifies correctly, click the upload button to upload the program → to the hardware. When the debugging area shows "Done uploading.", you can proceed. Open the serial monitor and rotate the knob potentiometer to observe the data changes displayed in the serial monitor. These changes represent the angle value of the knob.

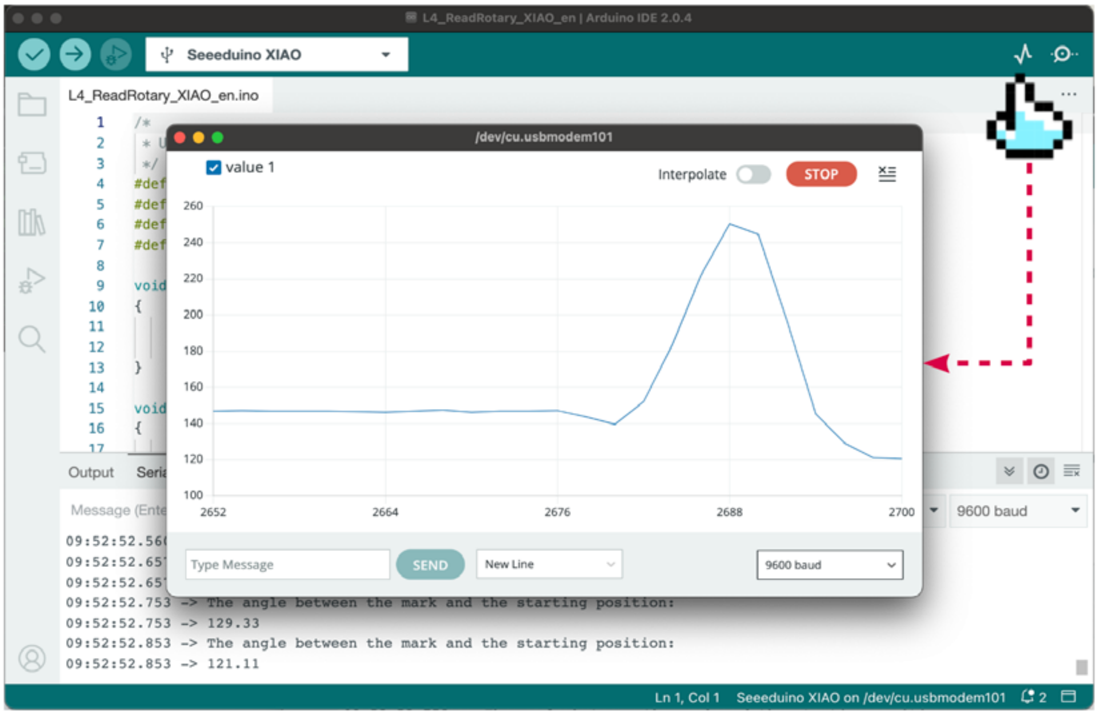

L4_ReadRotary_XIAO_en | Arduino IDE 2.0.4
Seeeduino XIAO
L4_ReadRotary_XIAO_en.ino
Output
Serial Monitor
Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 1, Col 1
Seeeduino XIAO on /dev/cu.usbmodem101
2

```
/*
 * Use the serial monitor to view the knob potentiometer
 */
#define ROTARY_ANGLE_SENSOR A0//Define the rotary potentiometer interface A0
#define ADC_REF 3 //ADC reference voltage 3V
#define GROVE_VCC 3 //Reference voltage 3V
#define FULL_ANGLE 300 //The maximum rotation angle of the rotary potentiometer is 300°

void setup()
{
    Serial.begin(9600);//Initialize the serial port
    pinMode(ROTARY_ANGLE_SENSOR, INPUT);//Set the rotary potentiometer pin as an input
}

void loop()
{
    float voltage;//Variable voltage is of floating-point type
```

```
09:42:38.130 -> 122.58
09:42:38.230 -> The angle between the mark and the starting position:
09:42:38.230 -> 99.71
09:42:38.327 -> The angle between the mark and the starting position:
09:42:38.327 -> 81.82
09:42:38.428 -> The angle between the mark and the starting position:
09:42:38.428 -> 74.49
```

1.4.4 Extended Exercise

While observing the angle value of the knob potentiometer in the serial monitor, we find that the value is constantly jumping and changing. Observing through the numbers alone is not very intuitive. At this time, we can use the serial plotter. With it, we can plot the data that is printed to the Arduino's serial port in real time. Based on the second task, close the serial monitor and open the "**Tools → Serial Plotter**" as shown in the image below:



```
09:52:52.56
09:52:52.65
09:52:52.65
09:52:52.753 -> The angle between the mark and the starting position:
09:52:52.753 -> 129.33
09:52:52.853 -> The angle between the mark and the starting position:
09:52:52.853 -> 121.11
```

The serial plotter draws the data obtained from the serial port into an XY axis curve chart, where the X-axis represents the change in time and the Y-axis represents the data obtained from the serial port. Through the chart, you can more intuitively see the change in data. Please give it a try.

1.5 Controlling LED and Servo with a Knob

In the last section, we learned how to use the serial monitor and observed the differences between digital input and analog input through it. In this section, we will further explore the use of analog values by combining them with a rotary potentiometer!

1.5.1 Background Knowledge

Servo and Servo Library

Servo

A servo, also known as a servo motor, is a DC motor with gears and a feedback system. We can control the servo to rotate to a specific angular position by sending signals to the circuit. This makes it suitable for electronic devices or robots that require precise position control.

Servo Library servo.h

When we want to control a servo using XIAO or other Arduino development boards, we can use the servo.h library file. It's one of the Arduino standard libraries, which is convenient to use and also avoids the problem of limited PWM pin quantity. Here are the relevant functions of the servo library:

- Declare the library file

  `#include <Servo.h>`
- Create the `myservo` object to control the servo

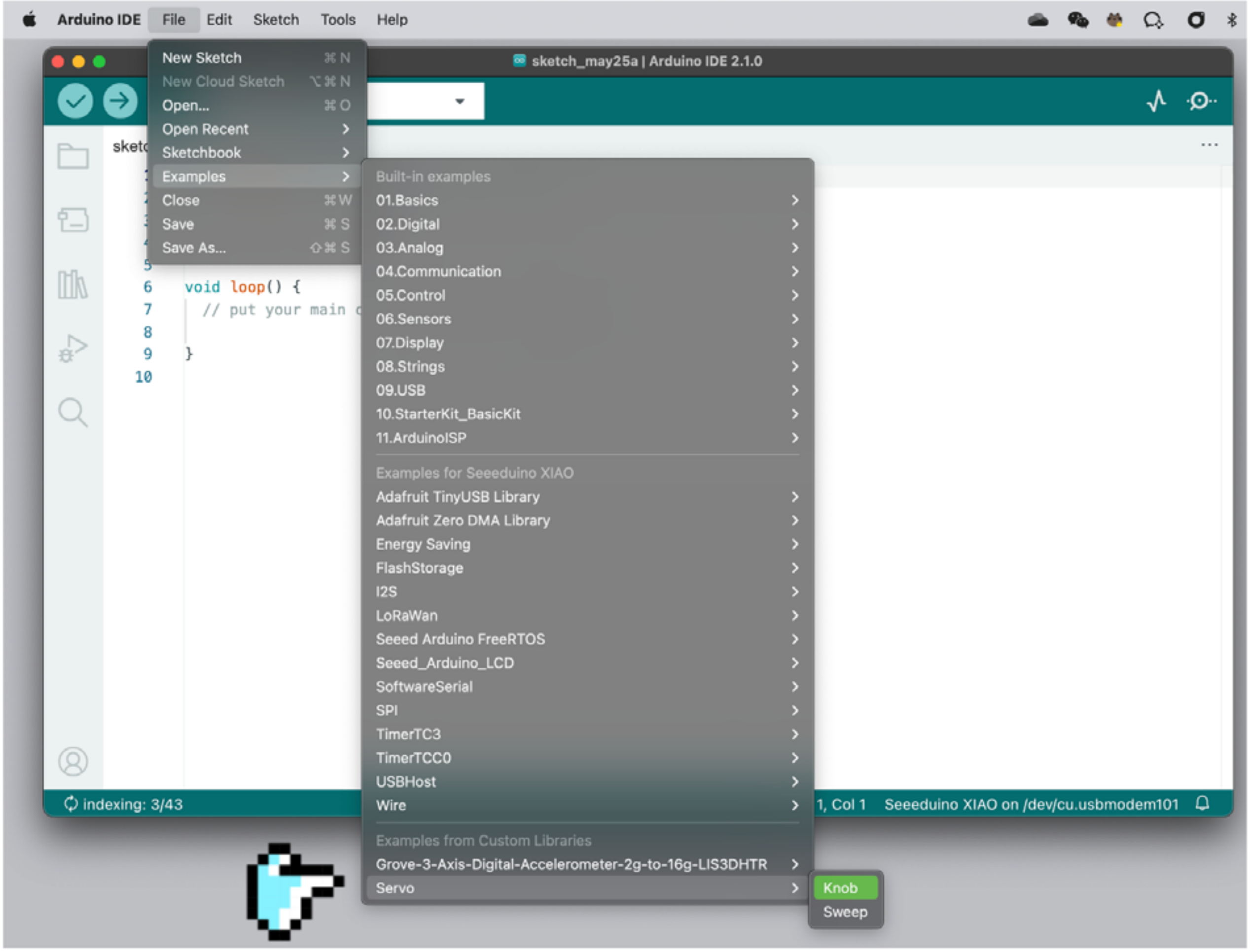
  `Servo myservo;`
- Use the `attach()` function to call the signal pin

  `myservo.attach();`
- Use the `write()` function to write the angle to the servo, setting the rotation angle of the shaft

  `myservo.write();`

The servo library does not need to be manually installed. You can open the example program "**File → Examples → Servo**" and check the two example programs "Knob" and "Sweep" to familiarize yourself with the use of the servo library.

If you can't find Servo under Examples, you can visit https://github.com/arduino-libraries/Servo and add the Servo example by installing the library.



```
void loop() {
  // put your main c
}
```

`map()` Function

The `map()` function is used to map a number from one range to another. That is, `fromLow` gets mapped to `toLow`, and `fromHigh` gets mapped to `toHigh`. It's the simplest form of linear mapping.

Syntax: `map(value, fromLow, fromHigh, toLow, toHigh)`

Parameters:

`value`: The number to be mapped.

`fromLow`: The lower limit of the current range of the value.

`fromHigh`: The upper limit of the current range of the value.

`toLow`: The lower limit of the target range of the value.

`toHigh`: The upper limit of the target range of the value.

Example: Map `val` from the range 0-1023 to 0-255.

```
void setup() {}
void loop() {
    int val = analogRead(0); // read the value from analog pin A0
    val = map(val, 0, 1023, 0, 255); // map val to the range 0-255
    analogWrite(9, val); // output the analog value to pin 9
}
```

1.5.2 Task 1: Using a knob potentiometer to control the brightness of the onboard LED on the XIAO board

Analysis:

When using a knob potentiometer to control the LED, we need to use the map() function, because the analog value directly output by the knob potentiometer is 0-1023, this value is not the angle value of the knob rotation, we need to calculate the angle value of the knob potentiometer rotation first, then map this value to the brightness range of the LED 0-255 with the map() function. The steps to write the program are as follows:

- Define the knob potentiometer, LED pin.
- Initialize the serial port, set the status of the knob potentiometer and LED pin.
- Read and calculate the rotation angle value of the knob potentiometer, and send it to the serial port.
- Map the angle value of the knob potentiometer to the LED brightness value and store it in the brightness variable, and the LED outputs this variable value.

Writing the program

Step 1: Define the knob potentiometer, LED pin, here we need to define ADC and VCC reference voltage, in order to calculate the angle value of the knob potentiometer.

```
#define ROTARY_ANGLE_SENSOR A0 //Define rotary potentiometer interface A0
#define LEDPIN 13 //Define LED interface 13
#define ADC_REF 3 //Reference voltage 3V
#define GROVE_VCC 3 //GROVE reference voltage 3V
#define FULL_ANGLE 300 //The maximum rotation angle of the rotary potentiometer is 
300°
```

Step 2: Initialize the serial port, set the status of the knob potentiometer and LED pin.

```
void setup()
{
    Serial.begin(9600); //Initialize serial communication
    pinMode(ROTARY_ANGLE_SENSOR, INPUT); //Set the rotary potentiometer pin to input
    pinMode(LEDPIN,OUTPUT); //Set the LED pin to output
}
```

Step 3: Read and calculate the rotation angle value of the knob potentiometer, and send it to the serial port.

```
void loop()
{
    float voltage; //Variable voltage of type float
     int sensor_value = analogRead(ROTARY_ANGLE_SENSOR); //Read the analog value at 
the rotary potentiometer pin
    voltage = (float)sensor_value*ADC_REF/1023; //Calculate the real-time voltage
    float degrees = (voltage*FULL_ANGLE)/GROVE_VCC; //Calculate the angle of rotation 
of the knob
    Serial.println(“The angle between the mark and the starting position:”); //Print 
character on serial monitor
    Serial.println(degrees); //Print the rotation angle value of the rotary potenti-
```

```
ometer on the serial monitor
    delay(100);
```

Step 4: Map the angle value of the knob potentiometer to the LED brightness value and store it in the brightness variable, and the LED outputs this variable value.

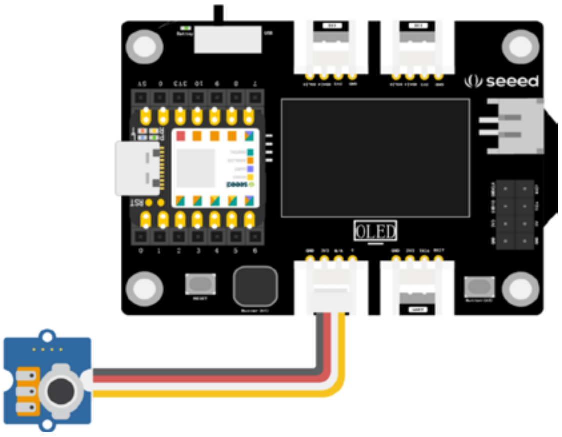
```
//After Step 3
    int brightness; //Define brightness variable
    brightness = map(degrees, 0, FULL_ANGLE, 0, 255); //Map the rotation angle value
of the rotary potentiometer to the brightness value of the LED and store it in the
brightness variable
    analogWrite(LEDPIN,brightness); //Output the variable value to the LED
    delay(500);
}
```

The final complete code is shown below:

```
#define ROTARY_ANGLE_SENSOR A0 //Define rotary potentiometer interface A0
#define LEDPIN 13 //Define LED interface 13
#define ADC_REF 3 //Reference voltage 3V
#define GROVE_VCC 3 //GROVE reference voltage 3V
#define FULL_ANGLE 300 //The maximum rotation angle of the rotary potentiometer is
300°

void setup()
{
    Serial.begin(9600); //Initialize serial communication
    pinMode(ROTARY_ANGLE_SENSOR, INPUT); //Set the rotary potentiometer pin to input
    pinMode(LEDPIN,OUTPUT); //Set the LED pin to output
}

void loop()
{
    float voltage; //Variable voltage of type float
     int sensor_value = analogRead(ROTARY_ANGLE_SENSOR); //Read the analog value at
the rotary potentiometer pin
    voltage = (float)sensor_value*ADC_REF/1023; //Calculate the real-time voltage
    float degrees = (voltage*FULL_ANGLE)/GROVE_VCC; //Calculate the angle of rotation
of the knob
    Serial.println(“The angle between the mark and the starting position:”); //Print
character on serial monitor
    Serial.println(degrees); //Print the rotation angle value of the rotary potenti-
ometer on the serial monitor
    delay(100);

    int brightness; //Define brightness variable
    brightness = map(degrees, 0, FULL_ANGLE, 0, 255); //Map the rotation angle value
of the rotary potentiometer to the brightness value of the LED and store it in the
brightness variable
    analogWrite(LEDPIN,brightness); //Output the variable value to the LED
    delay(500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L5_RotaryLed_XIAO_en

Uploading the Program

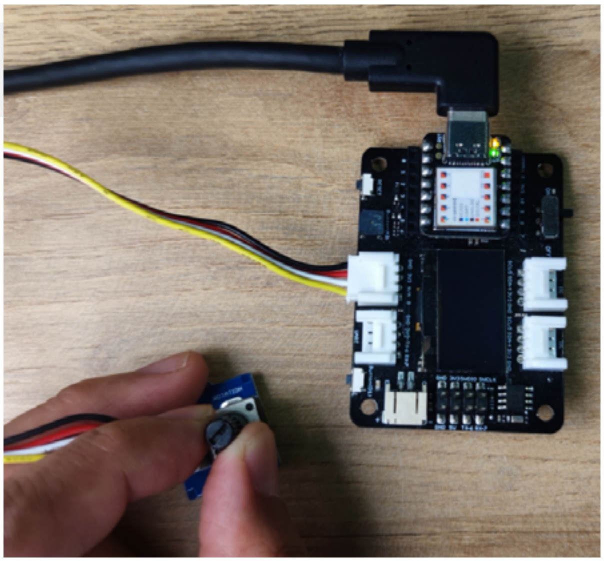


After writing the program, connect the rotary potentiometer to the `A0` interface using a four-color Grove wire, as shown in the following figure:

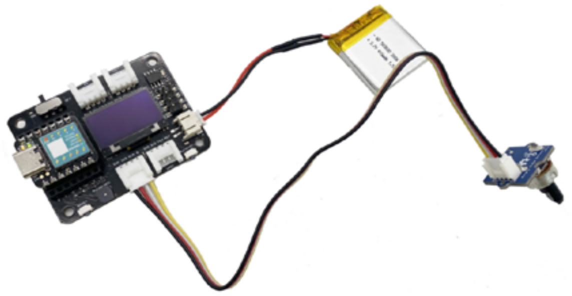
Connect the XIAO main control board to your computer with a data cable. After connecting, click ✓ (the verify button) in the Arduino IDE to check the program. If there are no errors, click → (the upload button) to upload the program to the hardware. When the debug area shows "Done uploading.", you can open the serial monitor to observe the rotation angle and LED brightness values as you rotate the potentiometer.

- Attention -

The onboard LED of the XIAO board is used in this example.

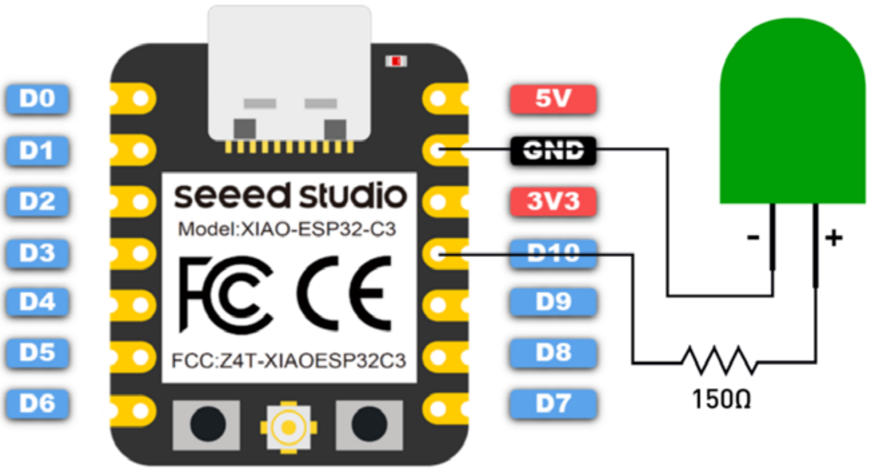


If you need to operate offline, you can connect a lithium battery to the expansion board, as shown in the following figure.



Controlling an External LED with a Knob on the XIAO ESP32C3

The Seeed XIAO ESP32C3 does not have an onboard LED for users. To run this program, you need to first connect an LED to the `D10` pin of the board, as shown below:



- Attention -

Be sure to connect a resistor (about 150Ω) in series with the LED to limit the current passing through the LED and prevent it from being damaged by overcurrent.

Next, copy the following program into the Arduino IDE:

```
#define ROTARY_ANGLE_SENSOR A0 // Define rotary potentiometer interface A0
#define LEDPIN D10 // Define LED light interface 10
#define ADC_REF 3 // Reference voltage 3V
#define GROVE_VCC 3 // GROVE reference voltage 3V
#define FULL_ANGLE 300 // The maximum rotation angle of the rotary potentiometer is 300°

void setup()
{
    Serial.begin(9600); // Initialize serial communication
     pinMode(ROTARY_ANGLE_SENSOR, INPUT); // Set the rotary potentiometer pin to input mode
    pinMode(LEDPIN, OUTPUT); // Set the LED light pin to output mode
}

void loop()
{
    float voltage; // Define voltage variable as float
     int sensor_value = analogRead(ROTARY_ANGLE_SENSOR); // Read the analog value on the rotary potentiometer pin
    voltage = (float)sensor_value*ADC_REF/1023; // Calculate real-time voltage
     float degrees = (voltage*FULL_ANGLE)/GROVE_VCC; // Calculate the angle of rotation of the knob
      Serial.println("The angle between the mark and the starting position:"); // Print string to serial port
     Serial.println(degrees); // Print the rotation angle value of the rotary potentiometer to the serial port
    delay(100);

    int brightness; // Define brightness variable
     brightness = map(degrees, 0, FULL_ANGLE, 0, 255); // Map the rotary potentiometer angle value to LED light brightness value and store it in the brightness variable
    analogWrite(LEDPIN, brightness); // Output brightness value to LED light
    delay(500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L5_RotaryLed_XIAO_ESP32C3_en

1.5.3 Task 2: Control a Servo Motor with a Rotary Potentiometer

Analysis

When controlling a servo motor with a rotary potentiometer, we can use the `servo.h` library and modify our first task slightly. The program can be divided into the following steps:

- Declare the servo library, define the servo rotation angle variable, define the rotary potentiometer pin and voltage.

- Initialize the serial port, set the status of the rotary potentiometer and servo pins.
- Read and calculate the rotation angle value of the rotary potentiometer, send it to the serial port, and drive the servo to rotate according to the angle value change.

Program Writing

Step 1: Declare the servo library, define the servo rotation angle variable, define the rotary potentiometer pin and voltage.

```
#include <Servo.h>// Declare the use of the servo library
#define ROTARY_ANGLE_SENSOR A0 // Define the rotary potentiometer pin as A0
#define ADC_REF 3 // ADC reference voltage is 3V
#define GROVE_VCC 3 // GROVE module reference voltage is 3V
#define FULL_ANGLE 300 // The maximum rotation angle of the rotary potentiometer is
300°
Servo myservo;  // Create a myservo object to control the servo
int pos = 0; // Variable to store the rotation angle of the servo
```

Step 2: Initialize the serial port, set the status of the rotary potentiometer and servo pins.

```
void setup() {
    Serial.begin(9600);// Initialize the serial port
    pinMode(ROTARY_ANGLE_SENSOR, INPUT);// Set the rotary potentiometer pin as input
     myservo.attach(5);  // The myservo signal is transmitted through pin 5, if you
are using XIAO RP2040/XIAO ESP32, please modify 5 to D5
}
```

Step 3: Read and calculate the rotation angle value of the rotary potentiometer, send it to the serial port, and drive the servo to rotate according to the angle value change.

```
void loop() {
    float voltage;// Set voltage as a floating point
     int sensor_value = analogRead(ROTARY_ANGLE_SENSOR);// Read the analog value at
the rotary potentiometer pin
     voltage = (float)sensor_value * ADC_REF / 1023;// Real-time voltage is the read
analog value multiplied by the reference voltage divided by 1023
     float degrees = (voltage * FULL_ANGLE) / GROVE_VCC;// The rotation angle of the
knob is the real-time voltage multiplied by the maximum rotation angle of the rotary
potentiometer divided by the voltage value of the GROVE module interface
    Serial.println(“The angle between the mark and the starting position:”);// Print
characters on the serial port
    Serial.println(degrees);// Print the rotation angle value of the rotary potenti-
ometer on the serial port
    delay(50);
    myservo.write(degrees); // Write the rotation angle value of the rotary potenti-
ometer into the servo
}
```

The final code is as follows:

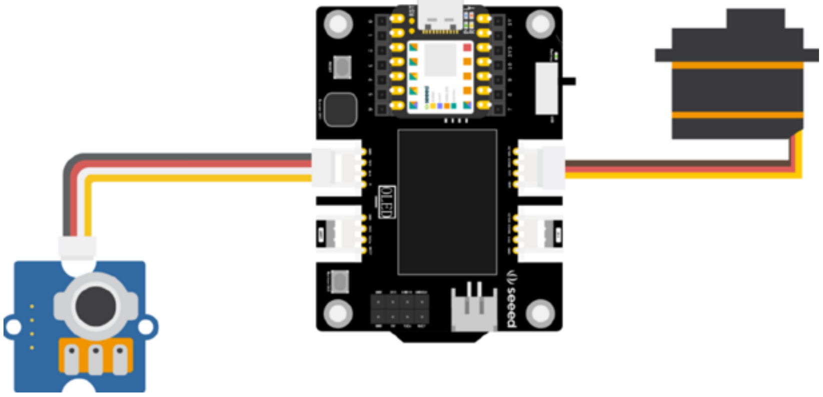
```
#include <Servo.h>// Declare the use of the servo library
#define ROTARY_ANGLE_SENSOR A0 // Define the rotary potentiometer pin as A0
#define ADC_REF 3 // ADC reference voltage is 3V
#define GROVE_VCC 3 // GROVE module reference voltage is 3V
```

```
#define FULL_ANGLE 300 // The maximum rotation angle of the rotary potentiometer is
300°
Servo myservo;  // Create a myservo object to control the servo
int pos = 0; // Variable to store the rotation angle of the servo

void setup() {
    Serial.begin(9600);// Initialize the serial port
    pinMode(ROTARY_ANGLE_SENSOR, INPUT);// Set the rotary potentiometer pin as input
     myservo.attach(5);  // The myservo signal is transmitted through pin 5, if you
are using XIAO RP2040/XIAO ESP32, please modify 5 to D5
}

void loop() {
    float voltage;// Set voltage as a floating point
     int sensor_value = analogRead(ROTARY_ANGLE_SENSOR);// Read the analog value at
the rotary potentiometer pin
     voltage = (float)sensor_value * ADC_REF / 1023;// Real-time voltage is the read
analog value multiplied by the reference voltage divided by 1023
     float degrees = (voltage * FULL_ANGLE) / GROVE_VCC;// The rotation angle of the
knob is the real-time voltage multiplied by the maximum rotation angle of the rotary
potentiometer divided by the voltage value of the GROVE module interface
    Serial.println(“The angle between the mark and the starting position:”);// Print
characters on the serial port
    Serial.println(degrees);// Print the rotation angle value of the rotary potenti-
ometer on the serial port
    delay(50);
    myservo.write(degrees); // Write the rotation angle value of the rotary potenti-
ometer into the servo
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L5_RotaryServo_XIAO_en

Upload Program

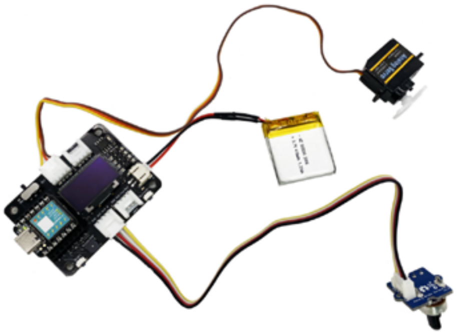
After writing the program, first connect the knob potentiometer and the servo to the XIAO expansion board as shown in the figure below. Then, connect the XIAO main control board to the computer with a data cable.



After the connection, click ✓ (the verify button) in the Arduino IDE to verify the program. If the verification is error-free, click → (the upload button) to upload the program to the hardware. When the debugging area shows “Done uploading.”, you can open the serial monitor, rotate the knob potentiometer, and observe the changes in angle value and the movement of the servo. What have you found?

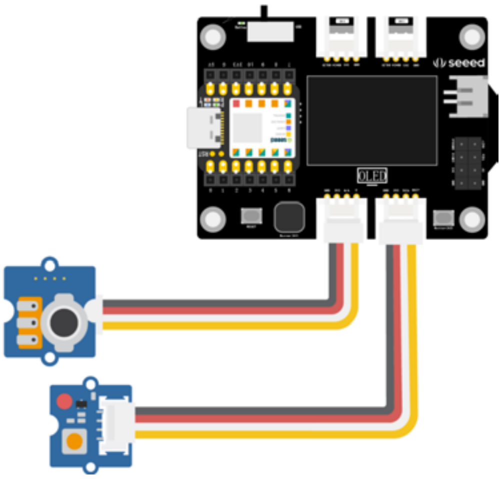
- Attention -

The rotation range of the servo is 0°-180°, so you will see in the serial monitor that when the angle value is greater than 180°, the servo stops rotating.



1.5.4 Extended Exercise

We have been using the LED on the XIAO board. If I want to use an external LED and control it with a knob potentiometer to create a breathing light effect, what should I do? The XIAO expansion board brings out two digital-analog Grove interfaces, and there is an A7/D7 interface. We can connect the external LED to this interface, as shown in the figure:



After the connection, we can slightly modify the program from Task 1, changing `#define LEDPIN 13 to #define LEDPIN 7`. Upload the modified program and see if it can achieve our desired effect.

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L5_RotaryLed_ledmodule_en

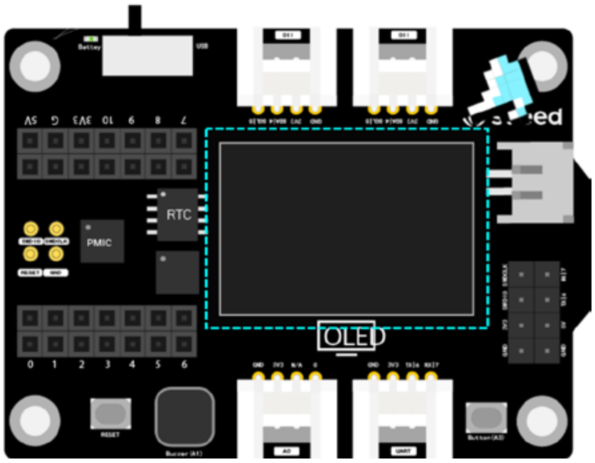
1.6
Displaying "Hello World" on OLED

In our daily life, we see displays everywhere - televisions, computers, phones, car displays, LCD billboards in shopping malls... Without a variety of screens, our lives would lose much of its fun. Of course, these screens, besides leisure and entertainment, are also indispensable tools for daily life. Common displays include LCD displays, OLED displays, etc. They all have their own strengths and weaknesses as display devices and can be applied in different fields and scenarios. The XIAO expansion board integrates an OLED display. In this lesson, we will learn how to use OLED to display text, patterns, and images.

1.6.1 Background Knowledge

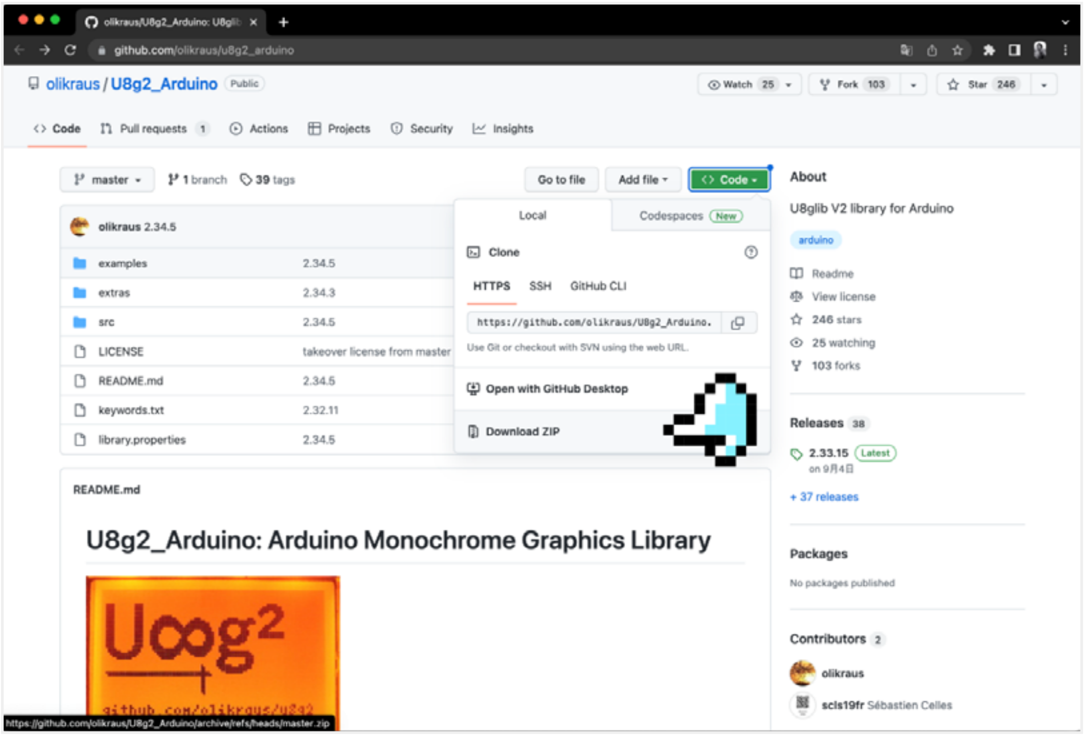
OLED Display

OLED, also known as Organic Light Emitting Diode, has advantages such as self-luminous, low power consumption, fast response speed, high resolution, light weight, etc. Its application field is very wide. The XIAO expansion board integrates a 0.96 inch 128x64 pixel OLED display, which can be used directly without wiring. During project production, we can display time, temperature and humidity, and other sensor return values through the OLED display, and we can also directly display letters, numbers, graphics, and even patterns, achieving visual interactive effects.

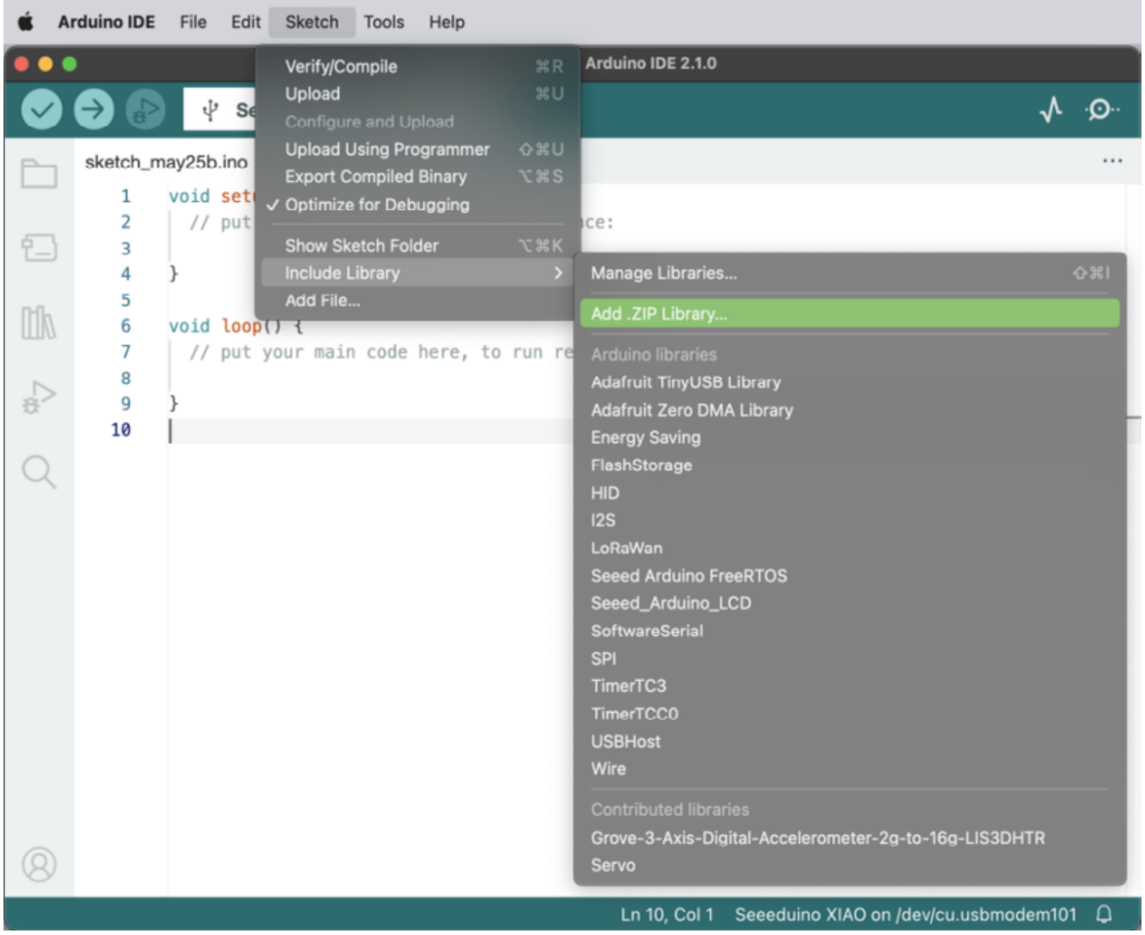


How to Download and Install the U8g2_Arduino Library

A library is a collection of program codes, which encapsulates some commonly used functions into a file for users to call. When we use OLED displays, temperature and humidity sensors, etc., we need to use the corresponding libraries. Where can these libraries be downloaded and how to install them? We will explain using the U8g2_Arduino library file of the OLED display as an example. Enter the website link https://github.com/olikraus/u8g2_arduino to enter the GitHub page, click Code→Download ZIP to download the resource package to the local, as shown in the figure below.



After the download is complete, open the Arduino IDE, click Sketch→Include Library→Add .ZIP Library, and select the ZIP file you just downloaded.



```
void set
  // put                          ce:

}

void loop() {
  // put your main code here, to run re

}
```

If the library is installed correctly, you can see the prompt information for successful library installation in the output window.

U8g2 Library for OLED

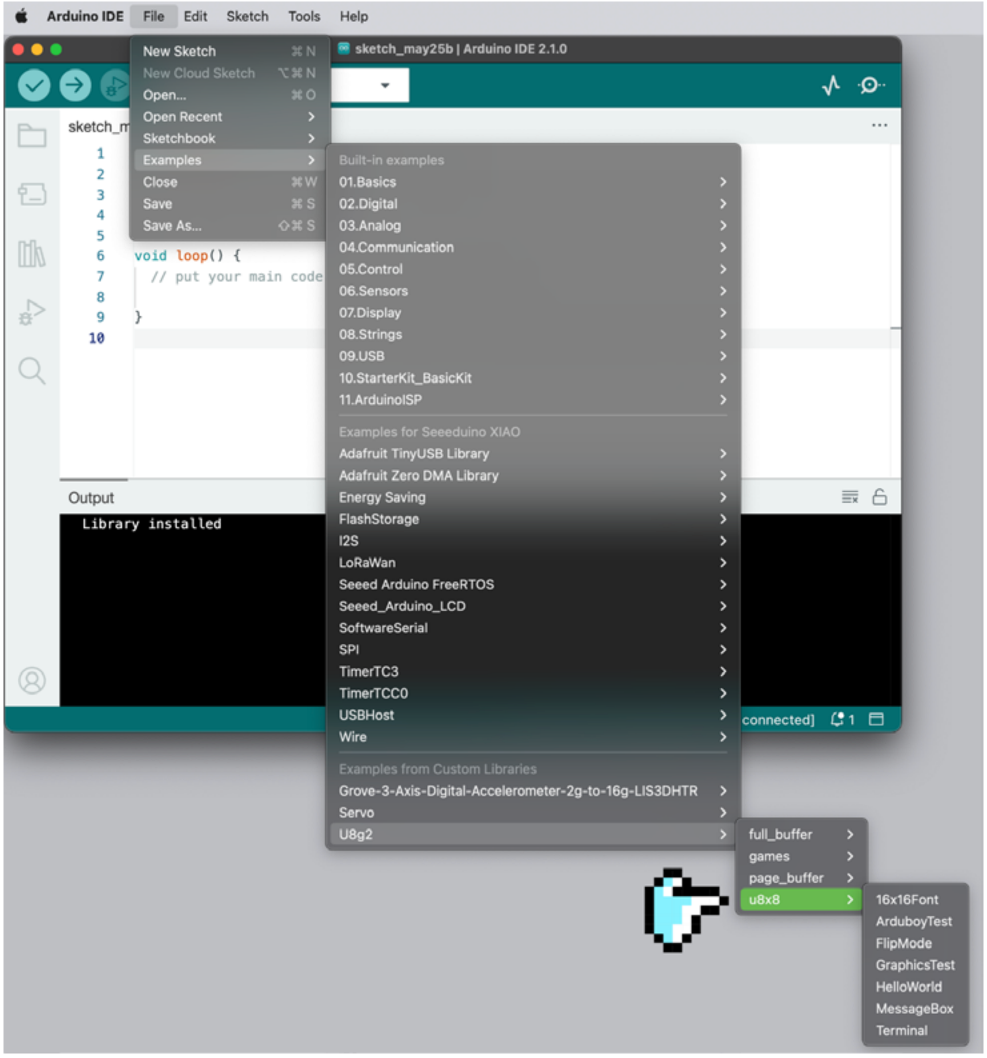
U8g2 is a monochrome graphics library for embedded devices, which supports various types of OLED displays, making it easy for us to write programs to achieve the desired effects. The U8g2 library also includes the U8x8 library, and the two libraries have different functions:

U8g2

Includes all graphic procedures (line/box/circle drawing); Supports various fonts, (almost) no restrictions on font height; Some memory in the microcontroller is needed to display.

U8x8

Only supports text (character) output; Only allows each character to use a fixed-size font (8x8 pixels); Writes directly to the display, no buffer is needed in the microcontroller. Simply put, when we want the OLED display to display various fonts, graphics, patterns, and present visual content more flexibly, we can use the U8g2 library; when we want to display characters more directly, with no font requirements, just to display sensor values, time, etc., we can use the U8x8 library, which is more efficient. We can find many example programs in "**File → Examples → U8g2**", and familiarize ourselves with the use of the library through the example programs.



```
void loop() {
  // put your main code
}
```

```
Library installed
```

Next, we will display characters and draw circles using two libraries respectively.

1.6.2 Task 1: Display Hello World! on the OLED of the XIAO expansion board

- Attention -

Before starting to write a program for the OLED of the XIAO expansion board, make sure the Arduino IDE has loaded the `U8g2_Arduino` *library file. The loading method can be referred to the description in the "How to Download and Install Arduino Library" section of this lesson.*

Analysis

If you just want to display "Hello World!" on the OLED, you can directly write characters with the U8x8 library. The steps are as follows:

- Declare the library file, set the constructor, and the constructor defines the display type, controller, RAM buffer size, and communication protocol.
- Initialize the display.
- Set the display font, set the print starting position, and output "Hello World!".

Write the program

Step 1: Declare the library file, set the constructor, and the constructor defines the display type, controller, RAM buffer size, and communication protocol.

```
#include <Arduino.h>
#include <U8x8lib.h>//Use U8x8 library file
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
//Set the constructor, define the display type, controller, RAM buffer size, and com-
munication protocol, generally determine according to the used display model
```

Step 2: Initialize the display. After declaring the library file in the previous step, you can use the functions in the library to set the OLED display.

```
void setup(void) {
    u8x8.begin();//Initialize u8x8 library
    u8x8.setFlipMode(1);//Flip the display 180 degrees, generally numbers 0 and 1
}
```

Step 3: Set the display font (there are various fonts to choose from in the u8x8 library, we can refer to https://github.com/olikraus/u8g2/wiki/fntlist8x8 to choose), set the print starting position, and output "Hello World!".

```
void loop(void) {
    u8x8.setFont(u8x8_font_chroma48medium8_r);//Define u8x8 font
    u8x8.setCursor(0, 0);//Set the position of the drawing cursor
    u8x8.print("Hello World!");//Draw content on OLED: Hello World！
}
```

The complete program is as follows:

```
#include <Arduino.h>
#include <U8x8lib.h>//Use U8x8 library file
```

```
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
//Set the constructor, define the display type, controller, RAM buffer size, and com-
munication protocol, generally determine according to the used display model

void setup(void) {
    u8x8.begin();//Initialize u8x8 library
    u8x8.setFlipMode(1);//Flip the display 180 degrees, generally numbers 0 and 1
}

void loop(void) {
    u8x8.setFont(u8x8_font_chroma48medium8_r);//Define u8x8 font
    u8x8.setCursor(0, 0);//Set the position of the drawing cursor
    u8x8.print("Hello World!");//Draw content on OLED: Hello World！
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L6_HelloWorld_XIAO_en

Program Upload

After the program is written, we connect the XIAO main control board to the computer interface using a data cable, as shown in the image below:

Click "Upload" to transfer the program to the main control board. Once the upload is complete, check if the OLED display shows "Hello World!".

1.6.3 Task 2: Draw a Circle on the OLED Display

Analysis

To draw a circle on the OLED display, we need to use the U8g2 library. Programming involves four steps:

- Declare the `U8g2` library file, determine whether to use SPI or I2C protocol, and set up the constructor to connect to the OLED display.
- The `draw()` function uses the `u8g2.drawCircle` function to draw a circle on the OLED.
- Initialize the `U8g2` library.
- In the `loop()` function, call related functions to draw images on the OLED.

Program Writing

Step 1: Declare the `U8g2` library file, determine whether to use SPI or I^{2}C protocol, and set up the constructor to connect to the OLED display.

```
#include<Arduino.h>
#include<U8g2lib.h>//Use U8g2 library
```

```
// Determine whether to use SPI or I2C protocol
#ifdef U8X8_HAVE_HW_SPI
#include<SPI.h>
#endif
#ifdef U8X8_HAVE_HW_I2C
#include<Wire.h>
#endif

U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);
// Set up the constructor, define display type, controller, RAM buffer size, and com-
munication protocol
```

Step 2: The `draw()` function uses the `u8g2.drawCircle` function to draw a circle on the OLED. The `u8g2.drawCircle(x0,y0,rad,opt)` function parameters are as follows:

- `x0,y0`: The position of the center of the circle.
- `rad`: Defines the size of the circle, with the diameter of the circle being 2*rad+1.
- `opt:` Choose a part or all of the circle.

```
void draw(void) {
    u8g2.drawCircle(20, 25, 10, U8G2_DRAW_ALL);// Draw a full circle with a diameter
of 21 at coordinates (20, 25)
}
```

Step 3: Initialize the `U8g2` library.

```
void setup(void) {
    u8g2.begin();// Initialize the library
}
```

Step 4: In the `loop()` function, call related functions to draw images on the OLED. Use the firstPage and nextPage functions to cycle through image content. They need to be used together, as shown in the program below:

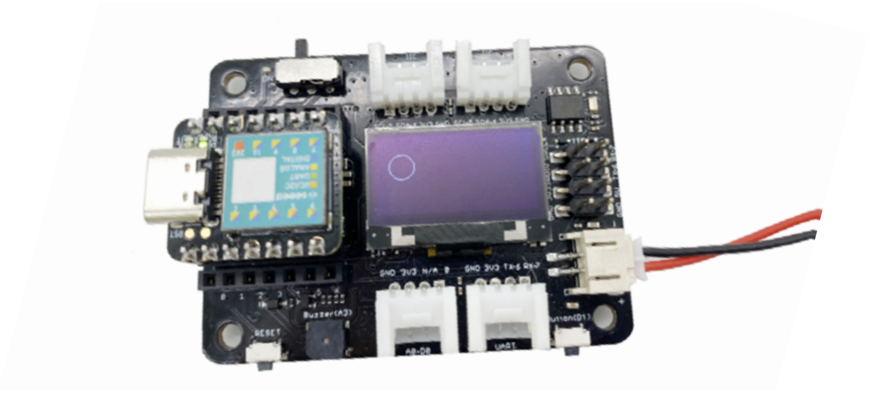
```
void loop(void) {
    // Cycle through image display
    u8g2.firstPage();
    do {
        draw();// Use draw function
    } while( u8g2.nextPage() );

    delay(1000);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L6_DrawCircle_XIAO_en

Uploading the Program

After writing the program, connect the XIAO main control board to the computer using a data cable, as shown below:

Once connected, click on the “Upload” button to upload the program to the hardware. When the debugging area displays “Upload Successful”, check if the OLED display screen has shown a circular pattern.



1.6.4 Extended Exercise

Try drawing some more complex patterns.

Chapter 2:
Project Practice for Beginners - Introduction to Prototype Design

This unit will delve into project practice with a few classic projects as case studies. We will learn how to create a quick verification prototype starting from an idea. Instead of analyzing code line by line as we've done previously, we will only explain critical steps in this unit. The focus will be more on the practical application of code. Arduino's libraries and example programs are abundant, as are community resources.

When working on a project, we should be adept at finding these resources, referring to example programs, and adjusting the code according to our needs to achieve the desired effects more quickly. Furthermore, this unit will begin to cover how to design appearances based on the effects achieved by the program. We will start by repurposing items around us, combining these items with electronic hardware to quickly form prototype works.

2.1
Introduction to Product Prototype Design

In the first unit, we've entered the realm of electronic hardware and programming, learning how to control electronic hardware through code to achieve desired effects, such as controlling an LED in various ways, making a buzzer sound, displaying text on an OLED screen, and more. Mastering this knowledge will help us turn the ideas in our minds into reality. In this section, we will learn about the process from an idea to a prototype and then to a product. Only when you have mastered this knowledge can you step into the world of product prototype design. If you have managed to stick with this course up to this point, there's no doubt that you are a "maker" at heart. The idea of "wanting to make something on your own" keeps swirling around in your mind. This section will provide you with some advice on how to become a maker and guidance on how to create electronic product prototype designs.

2.1.1 Cultivating the Maker Mindset

Becoming an excellent maker is not just about learning hardware modules and programming knowledge, but also consciously cultivating some habits.

Keep it playful

Play opens us to creative ideas and new experiences. While we play, we engage our bodies and our mind, and we often engage with others. While we play, learning feels natural and we can take risks to do things we didn't know we can do.

Be Curious

Ask questions – who, what, why, and how. How are things around you made? Who makes them and where are they made?

Dale Dougherty, the founder of Make: magazine, gave some advice on cultivating the maker mindset in his welcome speech at Maker Camp (refer to https://makercamp.com/get-started)

Get physical

Use your sense to experience the physical world all around you. What are the differences between the natural world and the built world?

Find a favorite tool

Tools exist for all kinds of applications. Given an area you're interested such as bicycles or music, what are some of the tools, both physical and digital, that you might want to learn to use? Choose a new tool and share it with us.

Do something you've never done before

Sometimes we decide that we're not good at something and we never try to do it. Part of the DIY spirit is to try something you've never tried before, even if you're not particularly good at it. Think of it as an experiment. See if you like it. Try cooking or gardening or playing a musical instrument. Or try to fix something that's broken. Share this new skill.

Make something

You might design something that solves a problem — it could be a problem for you or a problem for others. You might build something that's interactive such as a play toy, or a toy car or plane. Paper airplane launchers are popular, as are rockets.

What are the benefits of engaging young people in making?

Here are the five key competencies that we identified as outcomes for young people who participate in Start Making!

1. Identify as a creator or maker. Young people develop positive attitudes toward creating hands-on projects.
2. Develop confidence in creative expression. Young people feel capable of bringing their ideas to life by designing, experimenting, iterating, and persisting through failures.
3. Acquire technical tool literacy. Young people become familiar with a variety of tools and technologies that they can use to make projects.
4. Become aware of STEAM. Young people become aware of ideas and concepts that bridge science, technology, engineering, art, and math and demonstrate curiosity to learn more.
5. Learn collaboration and networking skills. Young people actively engage in collaborating and helping others.

Start Making! A Guide to Engaging Young People in Maker Activities By Danielle Martin and Alisha Panjwani edited by Natalie Rusk

What are the attributes of a maker? What is a maker mindset?

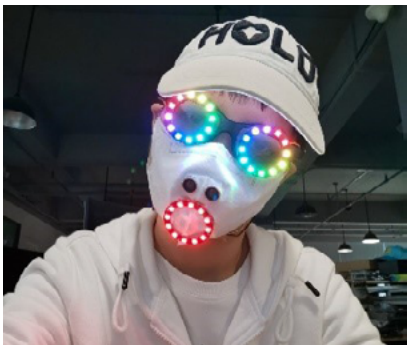
- Makers are curious. They are explorers. They pursue projects that they personally find interesting.
- Makers are playful. They often work on projects that show a sense of whimsy.
- Makers are willing to take on risk. They aren't afraid to try things that haven't been done before.
- Makers take on responsibility. They enjoy taking on projects that can help others.
- Makers are persistent. They don't give up easily.
- Makers are resourceful. They look for materials and inspiration in unlikely places.
- Makers share—their knowledge, their tools, and their support.
- Makers are optimistic. They believe that they can make a difference in the world.

Making Makers: Kids, Tools and the Future of Innovation By AnnMarie Thomas

2.1.2 Enlightening on Product Prototype Design

Author Introduction:

Wen Yanming, a post-90s female, graduated from the Chinese University of Hong Kong and South China University of Technology, with a master's degree in law. She is a hardware product manager, an inventor, an entrepreneur, with over a decade of technology practice and maker experience.



Basic Process of Product Prototype Design

From idea to product prototype and then to the product, this is a process that every product must go through. A product prototype allows us to quickly verify ideas, functionality, and product feasibility in a cost-effective way, providing the basis for product testing, optimization, and iterative updates. Behind every successful product we see, there may have been countless iterations of product prototypes. Therefore, creating a good product prototype is an essential process and solid foundation for a successful product.

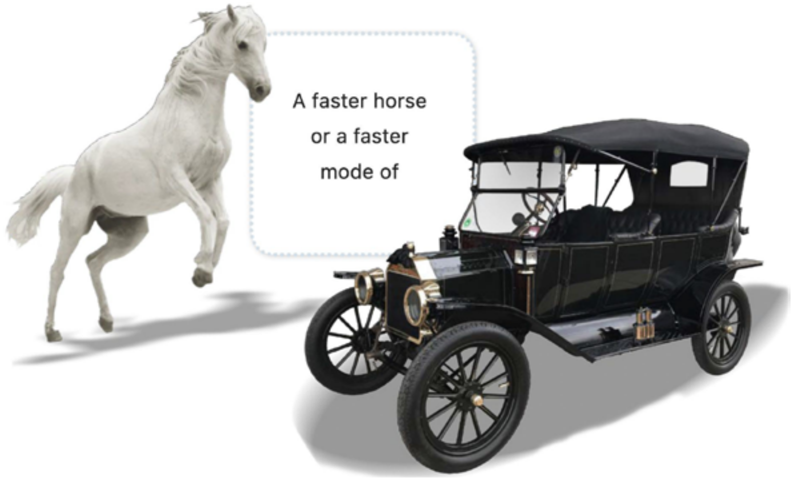
The prototypes needed for different types of products and different stages of the product are not the same. When we mention a product prototype, it may refer to a conceptual prototype, a functional prototype, a small batch production prototype, a factory hand model, etc. It should be noted that for electronic hardware products, the discussion here is mainly about product prototypes for product concepts and functional implementation.

Generally speaking, the design of a functional product prototype mainly includes the following processes:

1. Identify and Clarify the Problem to be Solved

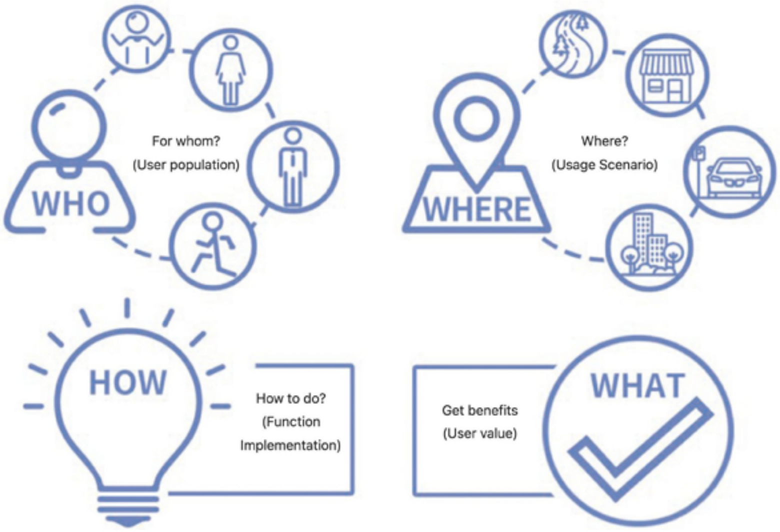
Einstein once said: “Posing a problem is often more important than solving a problem.” Every product must exist to solve a certain problem or to provide some benefit to people. Therefore, identifying and clarifying the problem to be solved is a prerequisite for clarifying product design needs and proceeding with product design.

It is important to note that just because we have identified a problem does not mean we truly understand and accurately define this problem. For example, over 100 years ago, when Henry Ford, the founder of Ford Motor Company, went around asking customers what kind of transportation they needed, almost everyone’s answer was, “I want a faster horse.” But do people really just need a faster horse? If Mr. Ford had defined the problem based on this, we might not have had faster and more comfortable cars so quickly.



2. Demand Analysis and Product Definition

Once the problem is clearly defined, we can extract unmet needs from it. Like the example above, the problem at that time was actually how to get to the destination faster, so the corresponding need was “a faster mode of transportation,” not “a faster horse.” Therefore, we need to be good at digging deeper from the problems we discover to find the real needs. Demand analysis generally requires an analysis of the user population and use scenarios, from which to derive the functions needed to solve the problem, that is, to clarify: for whom, in what scenarios, to achieve what functions, to gain what benefits.



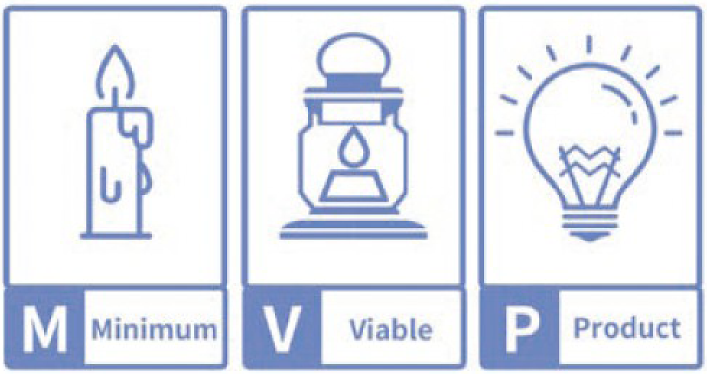
There are many types of needs: true user needs, superficial needs, urgent needs, ordinary needs, high-frequency needs, low-frequency needs, and so on. All of these need to be analyzed in light of the actual situation, which can then inform the correct definition of the product based on these needs.

Every product ultimately needs to be commercialized to realize its maximum value. Therefore, when designing a product for the market, we also need to conduct a series of market analyses, including market size, sales expectations, profit analysis, payback period, input-output ratio analysis, and so on.

3. Hardware Selection and Assembly

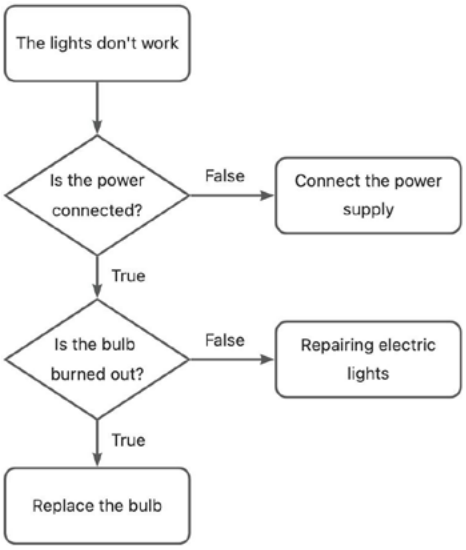
For the design of electronic products, once the needs are defined, we need to find hardware that can implement these functional needs. When choosing hardware, the elements that generally need to be considered include: feasibility, level of need satisfaction, cost, volume, weight, performance, lifespan, appearance, etc. One of the most important abilities of an excellent product designer is to take into account various factors based on the product definition and needs, balance these factors, and make trade-offs. Often, there is no single correct answer.

Generally speaking, when we build a prototype, the first thing we should consider is creating a minimum viable product (MVP). Its function is to use the least resources to quickly verify the product and quickly improve and iterate.



4. Software Development and Functional Implementation

Many experienced software development engineers will draw a functional implementation flowchart before software development. They will draw a functional implementation flowchart according to the functions to be implemented. This can help clarify the software design thinking, check the function logic, facilitate the identification of leaks and deficiencies, and refer to it at any time during programming, ensuring they have a clear understanding. Therefore, regardless of the complexity of software function development, it is recommended that everyone develop a good habit of drawing a functional implementation flowchart first. It can be a simple hand-drawn sketch, or a professional software like Visio, Axure can be used to draw it.

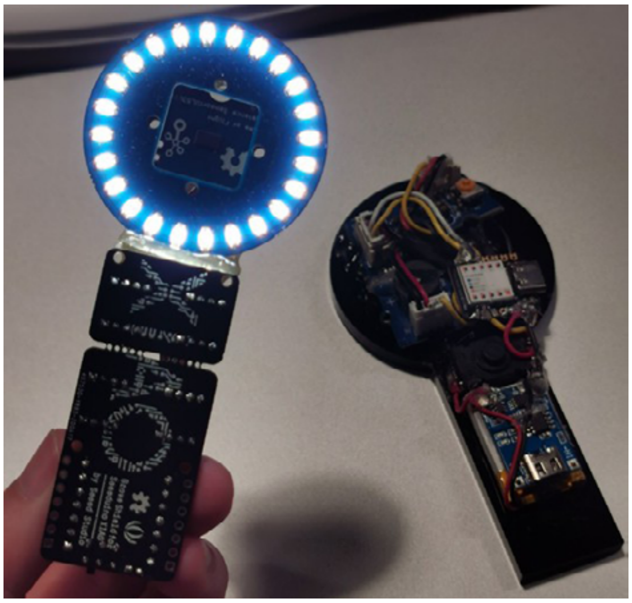


When developing software, try to be efficient and concise. Take full advantage of the benefits of the open-source community and learn to use existing hardware and software resources more effectively. For example, many pieces of hardware or applications already have many ready-made open-source libraries and routines. During development, you can refer to these, comply with the corresponding open-source agreements to use related resources, and avoid wasting time reinventing the wheel.

5. Prototype Testing and Optimization

After the prototype is made, we need to test it to verify its functional implementation and whether it meets the original design needs. This process should involve as many target users as possible to collect their feedback. In this way, we can better discover the defects in the product prototype, make remedial measures and improvements, update and iterate the design, and finally make a design scheme that meets user needs, laying a solid foundation for formal product design.

Product Prototype Practice - “One Meter Distance Alarm” Prototype



Next, let’s take the prototype manufacturing process of the “One Meter Distance Alarm” as an example to experience the product prototype design process.

1. Identifying and Defining the Problem to be Solved

At the beginning of 2020, the COVID-19 pandemic broke out globally, the situation was very severe. To prevent the virus from spreading through droplets and close-range airborne contact, governments and health departments around the world urged everyone to reduce gatherings and maintain social distancing of at least one meter whenever possible. However, it is not easy for everyone to constantly remember this and maintain an accurate social distance of more than one meter. For children, they often forget to maintain distance because they are playing happily, or they have no concept of how far they should keep their distance. When going out, there are also some strangers who, due to a lack of epidemic prevention awareness, unconsciously come close to us, and we need to find a polite way to remind them.

Therefore, we have derived a question from life: **How can we constantly remind people to maintain a social distance of more than one meter?**

2. Needs Analysis and Product Definition

With the problem defined, let's analyze the core needs that this problem triggers: an epidemic prevention reminder device for public use that sends out reminders when others enter within one meter, thus encouraging everyone to consciously maintain a one meter social distance.

Thinking further about the core needs, what kind of reminder should this be? We can think of the electronic products we usually use, what kind of reminders do they have? They are nothing more than sounds, lights, vibrations, screen text prompts, etc. Considering that the reminder needs to be timely, direct, and obvious, it's not easy to see the screen clearly at a distance of about one meter, and the volume will be larger and the cost higher after adding a screen, so we do not consider adding a screen. The remaining options are sound, light, and vibration. We can continue to balance and choose the necessary reminder method according to the cost, volume, and appearance. There is no single answer here. So, based on the needs analysis process, we tentatively define the product as: a device that emits light and vibrates to remind when it detects someone entering within a one meter distance.

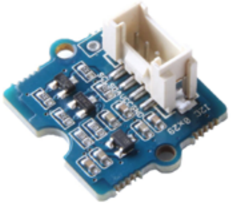
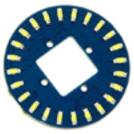
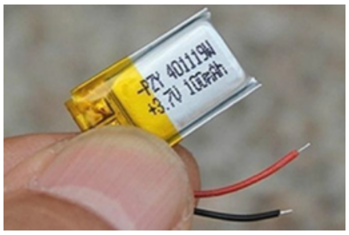
3. Hardware Selection and Assembly

With the product defined, we can decompose the core functional requirements:

1. Detect when a person enters within a one-meter distance
2. Alert self and others
3. Small size, easy to carry

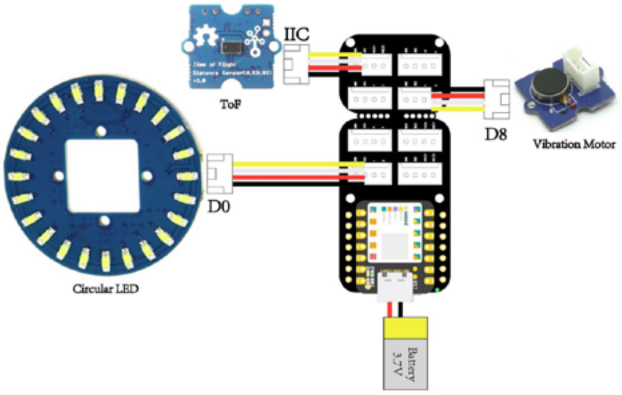
So, what kind of hardware should be used to implement these respectively? In the process of product prototype implementation, we usually choose open-source hardware with low cost, complete information, and many routines to implement hardware functions. After comprehensively considering the cost, function realization, assembly difficulty, volume, software development resources, and other elements, I have chosen the following hardware:

Functional Requirements	Hardware Product	Function Introduction
Main Board	Seeeduino XIAO (SAMD21)	This is a mini-main control board developed by Seed Technology based on SAMD21. The volume is very mini, only 20x17.5mm, the size of a thumb, the interface is rich, the performance is strong, very suitable for the development of various small volume devices.

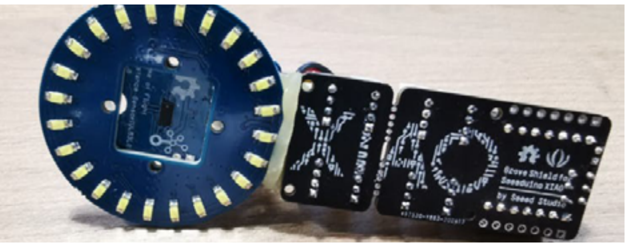
Expansion Board	 Seeed Studio Grove Base for XIAO	Grove Shield for Seeed Studio XIAO is a plug-and-play Grove extension board for Seeed Studio XIAO series. With the on-board battery management chip and battery bonding pad, you could easily power your Seeed Studio XIAO with lithium battery and recharge it. 8 Grove connectors onboard includes two Grove I2C and one UART. It acts as a bridge for Seeed Studio XIAO and Seeed's Grove system. Flash SPI bonding pad allows you add Flash to Seeed Studio XIAO to expand its memory space, providing Seeed Studio XIAO with more possibilities.
Distance Detection	 Grove - Time of Flight Distance Sensor (ToF)	There are many sensors to detect distance, most of which measure through ultrasound, infrared, lasers, etc. Among them, the Grove Time of Flight Distance Sensor is a new generation of ToF laser ranging module based on VL53L0X, which can provide accurate distance measurement up to 2 meters. The small size and high precision of this module made it my first choice.
Light Alarm	 Grove - Circular LED	A Grove - Circular LED with a circle of LEDs can light up a white light. It is aesthetically pleasing and provides a larger, more noticeable light reminder compared to a single LED.
Vibration Alarm	 Grove - Vibration Motor	A Grove module with a built-in vibration motor. It can be used plug-and-play, and it's convenient to generate continuous or intermittent vibration reminders by controlling the digital signal.
Power Supply	 3.7V lithium battery (401119)	A mini-sized 3.7V lithium battery commonly used for Bluetooth headset power supply. The model is 401119, which represents the thickness, width, and length of the battery as 4mm, 11mm, and 19mm respectively. After welding this size lithium battery to the lithium battery pad on the Grove expansion board, it can be placed directly in the gap between the Seeeduino XIAO and the Grove expansion board, making the product more tidy and beautiful.

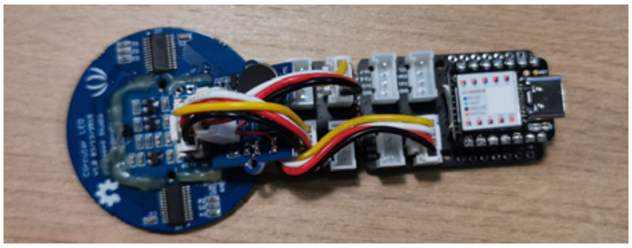
Writing	 Grove universal connection cable (5cm)	The Grove universal connector is a standard connector for the Grove system. It can be used conveniently plug-and-play, without soldering and considering the line sequence. The Grove line connects various sensors and actuators to the expansion board, making the project building as simple as building blocks and saving a lot of time. The 5cm short line is very suitable for space-compact product prototypes.

The module connection is as follows, as depicted in the image:



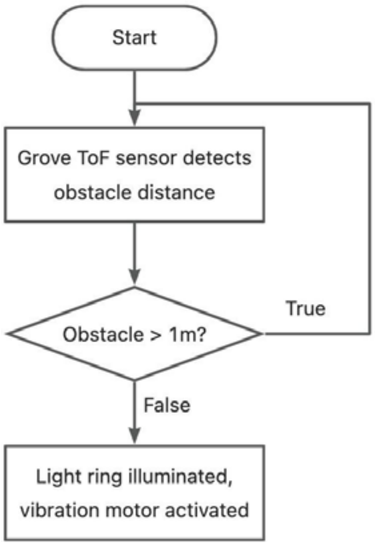
The chosen hardware modules have a great structural design, which can be directly used to build the distance alarm's form factor, saving time in making a shell. Thus, the production method is quite simple: all that's needed is to connect each piece of hardware to the appropriate interface, arrange their respective positions, and then bond them together with hot melt adhesive. This quickly completes the hardware connection and form factor building of a one-meter distance alarm. The completed hardware product is as follows:





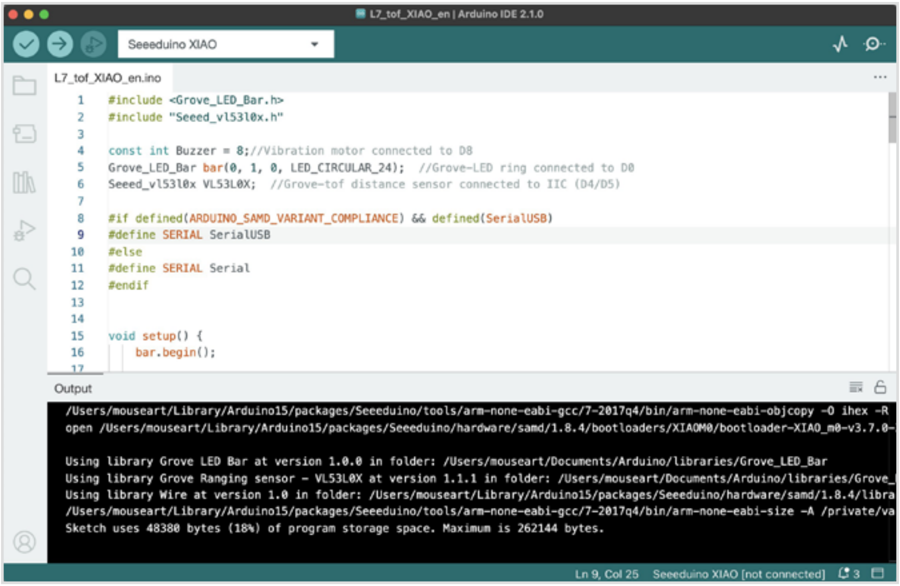
4. Software Development and Function Implementation

Before officially writing the program, I planned the functions and logic that the software needs to implement and drew the following functional implementation flow chart using Visio:


Start
Grove ToF sensor detects obstacle distance
Obstacle > 1m?
True
False
Light ring illuminated, vibration motor activated

Because Seeeduino XIAO supports Arduino IDE, I chose to program in the Arduino IDE. Most of the hardware provided by Seed Technology is open source, and they offer excellent documentation support for their products. Thus, during the programming process, I found the corresponding open-source hardware Wiki on the Seeedstudio official website, downloaded the relevant library files (note: library files are a collection of specific functionalities provided by developers that can be used by simply calling them, without having to rewrite the code), and referred to the example routines of the used modules. I completed the program swiftly.

After the program was written and compiled successfully, I connected the Seeeduino XIAO to the computer via a Type-C connection and downloaded the written code to the Seeeduino XIAO through the Arduino IDE. Once the code was successfully uploaded, the prototype was completed.


L7_tof_XIAO_en | Arduino IDE 2.1.0
Seeeduino XIAO
L7_tof_XIAO_en.ino
Output
Ln 9, Col 25 Seeeduino XIAO [not connected] 3

```
#include <Grove_LED_Bar.h>
#include "Seeed_vl53l0x.h"

const int Buzzer = 8;//Vibration motor connected to D8
Grove_LED_Bar bar(0, 1, 0, LED_CIRCULAR_24);  //Grove-LED ring connected to D0
Seeed_vl53l0x VL53L0X;  //Grove-tof distance sensor connected to IIC (D4/D5)

#if defined(ARDUINO_SAMD_VARIANT_COMPLIANCE) && defined(SerialUSB)
#define SERIAL SerialUSB
#else
#define SERIAL Serial
#endif


void setup() {
    bar.begin();
```

```
/Users/mouseart/Library/Arduino15/packages/Seeeduino/tools/arm-none-eabi-gcc/7-2017q4/bin/arm-none-eabi-objcopy -O ihex -R
open /Users/mouseart/Library/Arduino15/packages/Seeeduino/hardware/samd/1.8.4/bootloaders/XIAOM0/bootloader-XIAO_m0-v3.7.0-

Using library Grove LED Bar at version 1.0.0 in folder: /Users/mouseart/Documents/Arduino/libraries/Grove_LED_Bar
Using library Grove Ranging sensor - VL53L0X at version 1.1.1 in folder: /Users/mouseart/Documents/Arduino/libraries/Grove_
Using library Wire at version 1.0 in folder: /Users/mouseart/Library/Arduino15/packages/Seeeduino/hardware/samd/1.8.4/libra
/Users/mouseart/Library/Arduino15/packages/Seeeduino/tools/arm-none-eabi-gcc/7-2017q4/bin/arm-none-eabi-size -A /private/va
Sketch uses 48380 bytes (18%) of program storage space. Maximum is 262144 bytes.
```

5. Prototype Testing and Optimization

After completing the prototype, it was time for testing. First, I needed to test whether the prototype implemented the basic functionality, i.e., whether it would sound and light an alarm when a person was detected within a one-meter range. Then, I had to use it in an actual scenario to see if the user experience was good enough. If it could meet the product's requirements and definition satisfactorily, the product prototype could be deemed successful, and the next step in product development could be initiated. Of course, if issues were found during testing, adjustments and improvements were required, followed by retesting. This process is repeated until the product prototype meets the requirements, and the final scheme is determined.

Finishing the prototype is just the first step in making a successful product. The birth of each product requires a lot of effort, continual trial and error, and adjustment to achieve the best results. The final success of a product, in addition to meeting user needs, also needs to withstand many market tests. This requires students, when beginning to learn to make products, to always maintain the spirit of a craftsman, while also keeping a keen sense for the market, and learning knowledge beyond the product itself. There is a long way to go, and I hope everyone can stick to their original intentions, keep exploring, and ultimately make successful products.

The source code of the program is as follows:

```
#include <Grove_LED_Bar.h>
#include “Seeed_vl53l0x.h”

const int Buzzer = 8;//Vibration motor connected to D8
Grove_LED_Bar bar(0, 1, 0, LED_CIRCULAR_24);  //Grove-LED ring connected to D0
Seeed_vl53l0x VL53L0X;  //Grove-tof distance sensor connected to IIC (D4/D5)

#if defined(ARDUINO_SAMD_VARIANT_COMPLIANCE) && defined(SerialUSB)
#define SERIAL SerialUSB
#else
#define SERIAL Serial
#endif


void setup() {
    bar.begin();

    pinMode(Buzzer, OUTPUT);
    digitalWrite(Buzzer, LOW);   // turn the Buzzer on (HIGH is the voltage level)
    // Turn off all LEDs
    bar.setBits(0x0);

    VL53L0X_Error Status = VL53L0X_ERROR_NONE;
    SERIAL.begin(115200);
    Status = VL53L0X.VL53L0X_common_init();
    if (VL53L0X_ERROR_NONE != Status) {
        SERIAL.println(“Starting VL53L0X measurement failed!”);
        VL53L0X.print_pal_error(Status);
        while (1);
    }

    VL53L0X.VL53L0X_long_distance_ranging_init();

    if (VL53L0X_ERROR_NONE != Status) {
        SERIAL.println(“Starting VL53L0X measurement failed!”);
        VL53L0X.print_pal_error(Status);
        while (1);
    }

}

void loop() {
    VL53L0X_RangingMeasurementData_t RangingMeasurementData;
    VL53L0X_Error Status = VL53L0X_ERROR_NONE;

    memset(&RangingMeasurementData, 0, sizeof(VL53L0X_RangingMeasurementData_t));
    Status = VL53L0X.PerformSingleRangingMeasurement(&RangingMeasurementData);
    if (VL53L0X_ERROR_NONE == Status) {
        if (RangingMeasurementData.RangeMilliMeter >= 2000) {
            SERIAL.println(“Out of range!!”);
             digitalWrite(Buzzer, LOW);   // turn the Buzzer off (LOW is the voltage
level)

            // Turn off all LEDs
            bar.setBits(0x0);

        }
        else if (RangingMeasurementData.RangeMilliMeter <= 1000) {
            digitalWrite(Buzzer, HIGH);   // turn the Buzzer on (HIGH is the voltage
level)
            // Turn on all LEDs
            bar.setBits(0b1111111111111111111111111);

            SERIAL.print(“Distance:”);
            SERIAL.print(RangingMeasurementData.RangeMilliMeter);
            SERIAL.println(“ mm”);
        }
        else {
             digitalWrite(Buzzer, LOW);   // turn the Buzzer off (LOW is the voltage
level)

            // Turn off all LEDs
            bar.setBits(0x0);

            SERIAL.print(“Distance:”);
            SERIAL.print(RangingMeasurementData.RangeMilliMeter);
            SERIAL.println(“ mm”);
        }

    }
    else {
        SERIAL.print(“Measurement failed!! Status code =”);
        SERIAL.println(Status);
        digitalWrite(Buzzer, LOW);   // turn the Buzzer off (LOW is the voltage lev-
el)

        // Turn off all LEDs
        bar.setBits(0x0);
    }

    delay(250);

}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L7_tof_XIAO_en

2.2
Smart Hygrometer and Thermometer

Thermometers and hygrometers are ubiquitous in daily life, providing real-time measurement of temperature and humidity in our environment. We frequently use them to measure body temperature when we feel feverish or unwell. The invention of these devices has brought tremendous convenience to our lives. Despite their size, these devices hold a great deal of science. In this section, we will create a smart hygrometer and thermometer using a temperature and humidity sensor. Do you know what a temperature and humidity sensor is and what it can do?

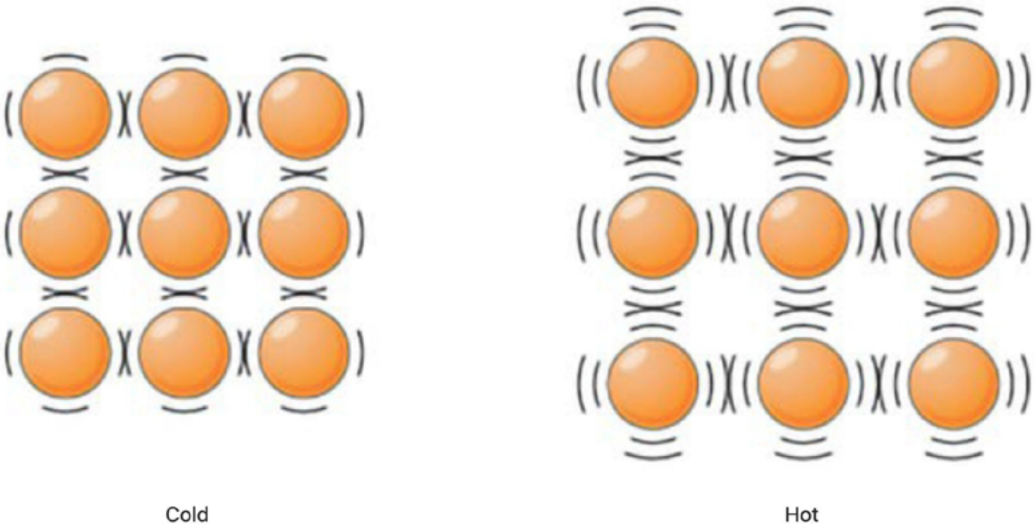
2.2.1 Background Knowledge

Temperature

Temperature is closely tied to our daily lives; it informs what clothes we wear before stepping out, and ensures the food or drink we consume is not too hot or too cold. When you step outside your home, you can sense the cold or heat, but to quantify exactly how cold or hot it is, we use "temperature".

Temperature is a physical quantity that indicates the degree of coldness or hotness of an object. The high and low temperature of an object is a macroscopic phenomenon that reflects the intensity of thermal motion of the molecules that make up the object at the microscopic level. Hence, the temperature is a manifestation of the intensity of thermal motion of a large number of molecules that constitute an object. The faster the molecular motion, the higher the temperature, and the hotter the object; the slower the molecular motion, the lower the temperature, and the colder the object.

To measure temperature accurately, we need to establish a temperature unit standard and design corresponding temperature measurement tools.



Temperature Scale

The unit standard for temperature is known as the temperature scale. Throughout the development of science, a variety of temperature scales have been devised, but their core methodology is the same: by stipulating the temperature values of certain phenomena or things, all other temperatures can be calibrated. Common temperature scales include Fahrenheit, Celsius, and Kelvin. Only five countries, including the United States and a few other English-speaking countries, still use the Fahrenheit scale. The vast majority of the world, including China, uses the Celsius scale. In research fields, scientists prefer to use the Kelvin scale.

- In the Fahrenheit scale, under standard atmospheric pressure, the temperature at which water begins to freeze is set at 32 degrees Fahrenheit and the temperature at which water boils is 212 degrees Fahrenheit. The scale is divided into 180 equal parts between these two points, with each part being one degree Fahrenheit, denoted as 1 ˚F . In the Fahrenheit scale, normal human body temperature is around 98 ˚F .
- In the Celsius scale, under standard atmospheric pressure, the temperature at which water begins to freeze is set at 0 degrees Celsius and the temperature at which water boils is 100 degrees Celsius. The scale is divided into 100 equal parts between these two points, with each part being one degree Celsius, denoted as 1℃ . In the Celsius scale, normal human body temperature is around 36.5℃ .
- The Kelvin scale is established on the basis of absolute zero. Scientists found that there is a minimum temperature in the universe, -273.15℃, which cannot be reached but only asymptotically approached. This minimum temperature was designated as absolute zero, and set as 0 Kelvin, denoted as 0K. The temperature at which water begins to freeze under standard atmospheric pressure is set at 273.15K, and the temperature at which water boils is 373.15K. In the Kelvin scale, normal human body temperature is around 309.7K.

Thermometer

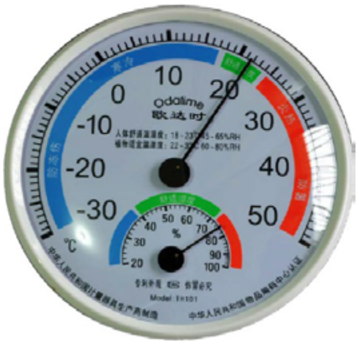
A thermometer is a tool for measuring temperature. Since temperature is not a physical quantity that can be seen directly, the measurement of temperature requires the assistance of physical phenomena directly related to temperature. For instance, in ancient China, there is a record of 'lustrous pure blue flame,' which is measured by observing the color of the flame.

Another example is the infrared thermometer, as shown in the image to the right, which measures temperature through the radiation differences of objects at different temperatures. Humans, like other organisms, also radiate infrared energy around them. This energy typically has a wavelength of 9-13μm and falls within the near-infrared band of 0.76-100μm. Since light within this wavelength range is not absorbed by air, the surface temperature of the human body can be accurately measured by simply measuring the infrared energy radiated by the human body. The human body infrared temperature sensor is designed and manufactured based on this principle.

Furthermore, the phenomenon of thermal expansion and contraction is often used in temperature measurement. Commonly seen thermometers and body thermometers are based on the principle of measuring temperature by the expansion and contraction of a liquid when heated or cooled. The image below shows a commonly used alcohol thermometer that measures temperature by the property of alcohol's expansion and contraction with temperature. The winter daytime temperature displayed in the image is -17°C.

Humidity

Humidity is a physical quantity that indicates the degree of dryness in the atmosphere. Under a certain temperature, the less water vapor a certain volume of air contains, the drier the air; the more water vapor, the more humid the air. The dryness or wetness of air is called "humidity." Weather forecasts typically report humidity values in terms of relative humidity, which is a percentage obtained by comparing the actual amount of water vapor in the air to the maximum amount of water vapor the air can hold at the same temperature.



Temperature and Humidity Sensor —— Grove Temperature and Humidity Sensor V2.0 (DHT20)

As the name implies, a temperature and humidity sensor is a sensor that can detect the temperature and humidity in the environment. There are many types of temperature and humidity sensors, and the one we chose is Grove Temperature and Humidity Sensor V2.0 (DHT20). This is a low-power, high-precision, and high-stability product with a fully calibrated digital I2C interface and a temperature measurement range of -40~80°C. Temperature and humidity sensors have a wide range of applications in the fields of agriculture, environmental protection, and home life.

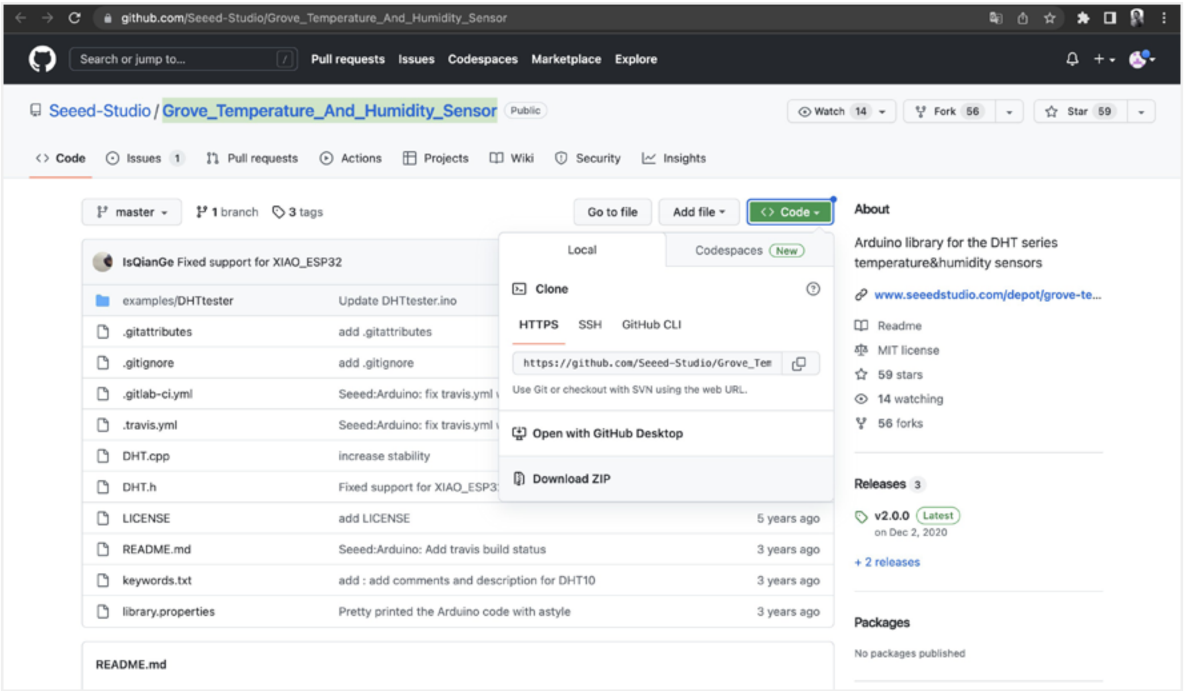
- Grove - Temperature and Humidity Sensor (DHT11) -

If you are using the Grove DHT11 version of the temperature and humidity sensor (with a blue sensor case), please refer to this version's Wiki document. DHT11 is a temperature and humidity sensor that outputs calibrated digital signals. The biggest difference between it and DHT20 is their communication method: DHT11 uses a single-bus digital signal, while DHT20 uses an I2C signal.

2.2.2 Task 1: Reading Temperature and Humidity Values in the Serial Monitor (Based on the DHT20 model)

Adding the `Grove_Temperature_And_Humidity_Sensor` Library File

Before starting to program the Grove Temperature and Humidity Sensor with the Arduino IDE, it is necessary to add the necessary library files for the sensor. Type the library file address in the browser address bar: https://github.com/Seeed-Studio/Grove_Temperature_And_Humidity_Sensor, enter the GitHub page, and click `Code→Download ZIP` to download the resource package `Grove_Temperature_And_Humidity_Sensor-master.zip` to your local machine, as shown in the image below.



IsQianGe Fixed support for XIAO_ESP32

examples/DHTtester	Update DHTtester.ino	
.gitattributes	add .gitattributes	
.gitignore	add .gitignore	
.gitlab-ci.yml	Seeed:Arduino: fix travis.yml	
.travis.yml	Seeed:Arduino: fix travis.yml	
DHT.cpp	increase stability	
DHT.h	Fixed support for XIAO_ESP3	
LICENSE	add LICENSE	5 years ago
README.md	Seeed:Arduino: Add travis build status	3 years ago
keywords.txt	add : add comments and description for DHT10	3 years ago
library.properties	Pretty printed the Arduino code with astyle	3 years ago

README.md

https://github.com/Seeed-Studio/Grove_Tem

Use Git or checkout with SVN using the web URL.

Arduino library for the DHT series temperature&humidity sensors

www.seeedstudio.com/depot/grove-te...

Add the resource package `Grove_Temperature_And_Humidity_Sensor-master.zip` downloaded in the previous step in the menu bar's `Sketch→Include Library→Add.ZIP Library` , until you see a prompt indicating the successful loading of the library.

Opening the "DHTtester" Example

Once the library file has been successfully added, the DHT library can be used. The "DHTtester" example can be opened through the following path: `File→Examples→Grove Temperature And Humidity Sensor→DHTtester`.

- *Attention* -

If the DHTtester example is not found in the menu after installing the library files, it can be viewed by closing and reopening the Arduino IDE.

After opening the example program, we can see a program like the one shown below. This program reads the temperature and relative humidity in the environment and displays real-time data in the serial monitor. Part of the example program's code needs to be modified.

```
// Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include “DHT.h”

// Uncomment whatever type you’re using!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
```

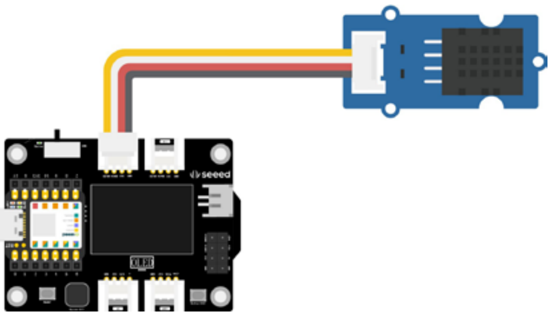
```
//#define DHTTYPE DHT10   // DHT 10
//#define DHTTYPE DHT20   // DHT 20

/*Notice: The DHT10 and DHT20 is different from other DHT* sensor ,it uses i2c interface rather than one wire*/
/*So it doesn’t require a pin.*/
#define DHTPIN 2     // what pin we’re connected to(DHT10 and DHT20 don’t need define it)
DHT dht(DHTPIN, DHTTYPE);   //   DHT11 DHT21 DHT22
//DHT dht(DHTTYPE);         //   DHT10 DHT20 don’t need to define Pin

// Connect pin 1 (on the left) of the sensor to +5V
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor


#if defined(ARDUINO_ARCH_AVR)
    #define debug  Serial

#elif defined(ARDUINO_ARCH_SAMD) ||  defined(ARDUINO_ARCH_SAM)
    #define debug  SerialUSB
#else
    #define debug  Serial
#endif

void setup() {

    debug.begin(115200);
    debug.println(“DHTxx test!”);
    Wire.begin();

    /*if using WIO link,must pull up the power pin.*/
    // pinMode(PIN_GROVE_POWER, OUTPUT);
    // digitalWrite(PIN_GROVE_POWER, 1);

    dht.begin();
}

void loop() {
    float temp_hum_val[2] = {0};
    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds ‘old’ (its a very slow sensor)


    if (!dht.readTempAndHumidity(temp_hum_val)) {
        debug.print(“Humidity: “);
        debug.print(temp_hum_val[0]);
        debug.print(“ %\t”);
        debug.print(“Temperature: “);
        debug.print(temp_hum_val[1]);
        debug.println(“ *C”);
    } else {
        debug.println(“Failed to get temprature and humidity value.”);
    }

    delay(1500);
```

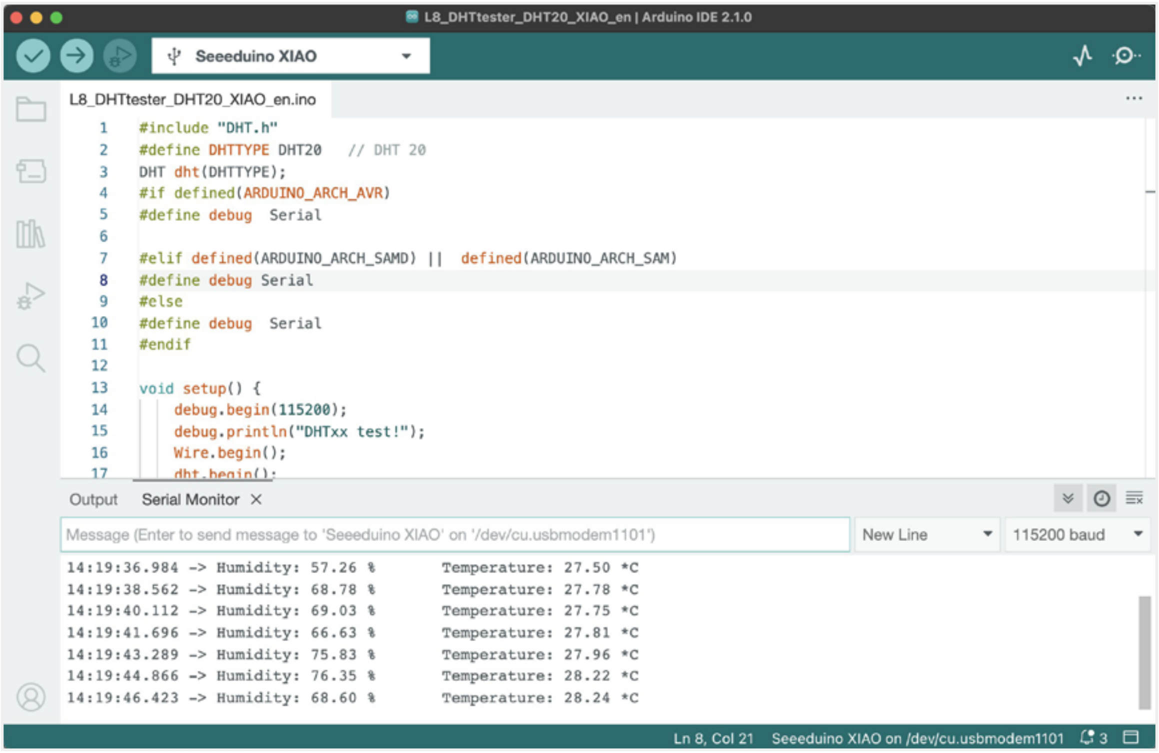
Pay attention to the document's comments. The program above provides several types of temperature and humidity sensor models (DHT22 is set as default), but we need the DHT20. So, uncomment the part for DHT20 and delete the definitions for other unneeded sensor models. DHT10 and DHT20 do not require pin definitions, so the revised code after modification is as follows:

```
#include "DHT.h"
#define DHTTYPE DHT20   // DHT 20
DHT dht(DHTTYPE);
#if defined(ARDUINO_ARCH_AVR)
#define debug  Serial

#elif defined(ARDUINO_ARCH_SAMD) ||  defined(ARDUINO_ARCH_SAM)
#define debug  Serial
#else
#define debug  Serial
#endif

void setup() {
    debug.begin(115200);
    debug.println("DHTxx test!");
    Wire.begin();
    dht.begin();
}

void loop() {
    float temp_hum_val[2] = {0};
    if (!dht.readTempAndHumidity(temp_hum_val)) {
        debug.print("Humidity: ");
        debug.print(temp_hum_val[0]);
        debug.print(" %\t");
        debug.print("Temperature: ");
        debug.print(temp_hum_val[1]);
        debug.println(" *C");
    } else {
        debug.println("Failed to get temprature and humidity value.");
    }

    delay(1500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L8_DHTtester_DHT20_XIAO_en

After modifying the code, first connect the temperature and humidity sensor to the I2C interface of the XIAO expansion board, as shown below. Then connect the XIAO development board to your computer, upload the modified example program to XIAO in the Arduino IDE, and open the serial monitor in Arduino IDE. You will now be able to see the values of temperature and humidity. Try placing the sensor in different environments to observe if the temperature and humidity values change.




L8_DHTtester_DHT20_XIAO_en | Arduino IDE 2.1.0
Seeeduino XIAO
L8_DHTtester_DHT20_XIAO_en.ino
Output
Serial Monitor
Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem1101')
Ln 8, Col 21
Seeeduino XIAO on /dev/cu.usbmodem1101
3

```
#include "DHT.h"
#define DHTTYPE DHT20   // DHT 20
DHT dht(DHTTYPE);
#if defined(ARDUINO_ARCH_AVR)
#define debug  Serial

#elif defined(ARDUINO_ARCH_SAMD) ||  defined(ARDUINO_ARCH_SAM)
#define debug Serial
#else
#define debug  Serial
#endif

void setup() {
    debug.begin(115200);
    debug.println("DHTxx test!");
    Wire.begin();
    dht.begin();
```

New Line | 115200 baud

```
14:19:36.984 -> Humidity: 57.26 %      Temperature: 27.50 *C
14:19:38.562 -> Humidity: 68.78 %      Temperature: 27.78 *C
14:19:40.112 -> Humidity: 69.03 %      Temperature: 27.75 *C
14:19:41.696 -> Humidity: 66.63 %      Temperature: 27.81 *C
14:19:43.289 -> Humidity: 75.83 %      Temperature: 27.96 *C
14:19:44.866 -> Humidity: 76.35 %      Temperature: 28.22 *C
14:19:46.423 -> Humidity: 68.60 %      Temperature: 28.24 *C
```

It appears the temperature and humidity sensor is functioning correctly.

Reading Temperature and Humidity Values in the Serial Monitor (Based on the DHT11 Sensor)

If you are using the Grove DHT11 Temperature and Humidity Sensor with a blue casing, parts of the program code need to be modified as follows:

`#define DHTPIN 0` needs to be modified according to the actual pin number the sensor is connected to.

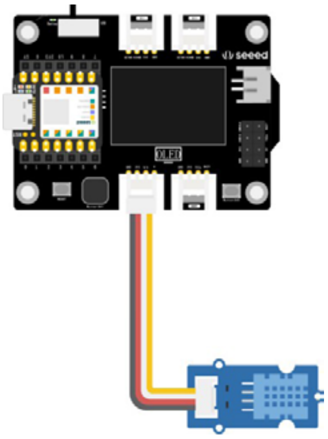
`#define DHTTYPE DHT11` should be set because there are different models of temperature and humidity sensors, and you need to choose the correct one, i.e., DHT11.

The example code after modification is shown below:

```
#include “DHT.h”
#define DHTTYPE DHT11   // DHT 11
#define DHTPIN 0
DHT dht(DHTPIN, DHTTYPE);

#if defined(ARDUINO_ARCH_AVR)
    #define debug  Serial

#elif defined(ARDUINO_ARCH_SAMD) ||  defined(ARDUINO_ARCH_SAM)
    #define debug  SerialUSB
#else
    #define debug  Serial
#endif

void setup() {
    debug.begin(115200);
    debug.println(“DHTxx test!”);
    Wire.begin();
    dht.begin();
}

void loop() {
    float temp_hum_val[2] = {0};
    if (!dht.readTempAndHumidity(temp_hum_val)) {
        debug.print(“Humidity: “);
        debug.print(temp_hum_val[0]);
        debug.print(“ %\t”);
        debug.print(“Temperature: “);
        debug.print(temp_hum_val[1]);
        debug.println(“ *C”);
    } else {
        debug.println(“Failed to get temprature and humidity value.”);
    }

    delay(1500);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L8_DHTtrster_DHT11_XIAO_en

After modifying the code, first connect the temperature and humidity sensor to the A0 port of the XIAO expansion board, as shown in the figure below. Then, connect the XIAO development board to the computer, upload the modified example program to XIAO in the Arduino IDE, and open the serial monitor in the Arduino IDE to see the values of temperature and humidity. You can place the temperature and humidity sensor in different environments to see if the temperature and humidity values will change.



2.2.3 Project Creation: Smart Temperature and Humidity Meter

Project Description

We are going to make a portable mini temperature and humidity detector that detects temperature and humidity values through a temperature and humidity sensor and displays the values on the OLED display of the XIAO expansion board. However, it is not rich enough to have only the display function. We can add a buzzer alarm function. When the detected temperature and humidity exceed a certain range, an alarm will be sounded as a reminder. The value range can be adjusted according to different application scenarios. For example, in a home life scenario, set a comfortable temperature and humidity range based on human feelings; or use it in plant planting places, set the temperature and humidity value range based on suitable plant growth, exceed the alarm, and remind people to adjust.

Program Writing

Referencing the example program above, one of the effects we want to achieve is to display the temperature and humidity values on the OLED display of the XIAO expansion board. The code for reading the temperature and humidity sensor detection values can be reused by just changing the display medium. In combination with Section 1.6, we have learned how to display characters on the OLED, so we just need to add an if...else condition judgment statement to judge the temperature and humidity values. The program writing idea is as follows:

- Declare the DHT.h library, U8x8 library, etc., and connect the buzzer pin as a reminder to sound the device.
- Initialize the library file, define the buzzer pin state.
- Define temperature and humidity variables to store readings and display them on the OLED screen, add logical judgment, and implement buzzer alarm.

To facilitate understanding and implementation, we divide the program implementation into two tasks:

1. Detect temperature and humidity and display them on the OLED screen of the XIAO expansion board.
2. Add alarm function.

Task 1: Use the Grove DHT20 sensor to detect temperature and humidity and display them on the OLED screen of the XIAO expansion board

Step 1: Headers, declare the library files to be called.

```
#include “DHT.h”
#include “DHT.h”    //Use DHT library
#include <Arduino.h>
#include <U8x8lib.h>    //Use u8x8 library
#define DHTTYPE DHT20
DHT dht(DHTTYPE);    //DHT20 does not need to define pins

U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);   //Setup con-
structor to connect to OLED screen
```

Step 2: Initialize the DHT library and the u8x8 library.

```
void setup() {
    Wire.begin();    //Initialize wire library, and join I2C network
    dht.begin();    //DHT starts working
    u8x8.begin();   //u8x8 starts working
    u8x8.setPowerSave(0);  //Turn off power saving mode, 1 is on, and nothing can be
seen on the screen after power saving mode is on
    u8x8.setFlipMode(1);
}
```

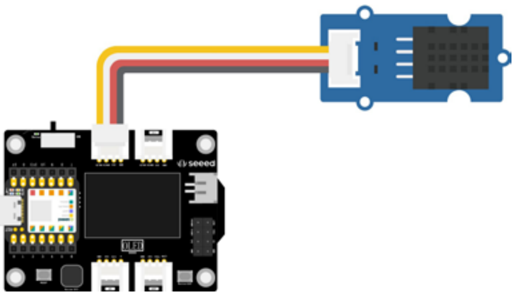
Step 3: Define temperature and humidity variables to store readings, read temperature and humidity values and display them on the OLED screen. Pay attention to the coordinate positions of temperature and humidity display.

```
void loop() {
    float temp, humi;   //Set the variables temp and humi to floating point type, rep-
resenting temperature and humidity respectively
    temp = dht.readTemperature();   //Read temperature value and store it in temp
    humi = dht.readHumidity();  //Read humidity value and store it in humi
    u8x8.setFont(u8x8_font_chroma48medium8_r);  //Set display font
    u8x8.setCursor(0, 33);  //Set the position of the drawing cursor (0,33)
```

```
      u8x8.print("Temp:");    //Display Temp at the position (0,33)
      u8x8.print(temp);   //Display real-time temperature value
      u8x8.print("C");    //Display the unit "C" of temperature
      u8x8.setCursor(0,50);
      u8x8.print("Humidity:");
      u8x8.print(humi);
      u8x8.print("%");
      u8x8.refreshDisplay();
      delay(200);
  }
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L8_dht20_tem_humi_XIAO_en

Step 4: Connect the hardware, upload the program Connect the temperature and humidity sensor to the I2C interface of the XIAO expansion board, as shown in the figure:



Use the data cable to connect XIAO to the computer, click the "upload" button in the Arduino IDE, and upload the program to the hardware. When the debugging area shows "upload successful", you can observe whether the temperature and humidity values are displayed on the OLED screen, and you can hold the black part of the sensor with your palm to observe whether the values change.

Task 2: Add an alarm function

Step 1: Add alarm function code. The alarm function requires a buzzer to be integrated into the circuit, which can be facilitated using the on-board buzzer of the XIAO expansion board. The program needs to set the buzzer pin state, add a part for condition judgment - when the temperature exceeds a certain value or the humidity falls below a certain value, the buzzer will sound an alarm. Here, a logical expression needs to be written using the "&&" logical operator "and".

Boolean Operators

&&: *Logical AND, represents "and",*`if (expression1 && expression2)`*, only when all expressions in the parentheses are* `true` *will it execute the statements in* `if {}`.

`||`: *Logical OR, represents "or",*`if (expression1 || expression2)`*, if either of the expressions are satisfied, the entire expression is true, and the statements in* `if {}` *are executed.*

!: Logical NOT, represents "not", `if (!expression1)`*, only when the value of expression1 in the parentheses is* `false` *will it execute the statements in* `if {}`.

Usage example:

When the temperature exceeds 30 or the humidity falls below 40, satisfying either condition will make the buzzer sound an alarm.

```
  if (temp > 30 || humi < 40) {
      tone(buzzerPin, 200, 200);
  }
```

The added part of the program mainly sets the buzzer and makes decisions based on temperature and humidity, controlling the buzzer to make a sound.

```
// Part of the program, will not run
int buzzerPin = A3; // Connects the buzzer to pin A3

void setup() {
    pinMode(buzzerPin , OUTPUT); // Sets the buzzer pin as output
}

void loop() {
    float temp, humi;
    temp = dht.readTemperature();
    humi = dht.readHumidity();
    if (temp > 30 || humi < 40) {  // When the temperature exceeds 30 or the humidi-
ty falls below 40, satisfying either condition will make the buzzer sound an alarm.
        tone(buzzerPin, 200, 200);
    }
```

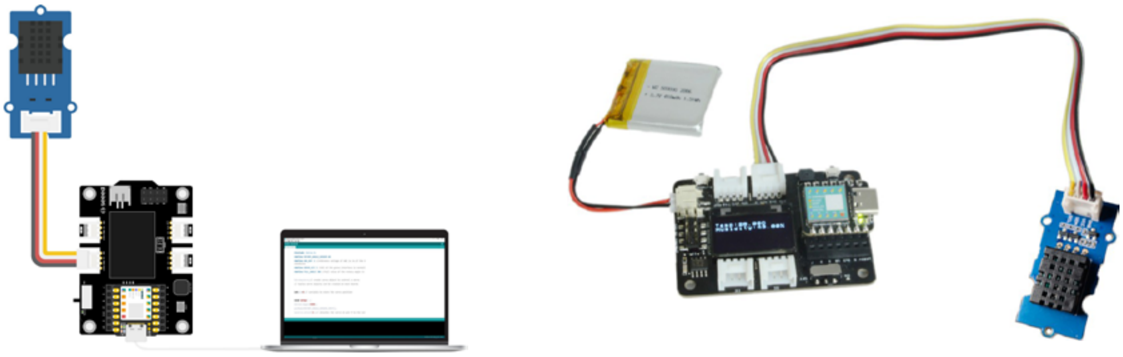
Add the above code to the corresponding location of the Task 1 program to realize all functions. The complete program is shown below:

```
#include "DHT.h" // Use DHT library
#include <Arduino.h>
#include <U8x8lib.h> // Use u8x8 library
#define DHTTYPE DHT20
DHT dht(DHTTYPE); // DHT20 does not require pin definition
int buzzerPin = A3;
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE); // Set construc-
tor to connect OLED display

void setup() {
    pinMode(buzzerPin , OUTPUT); // Set buzzer pin to output mode
    Wire.begin(); // Initialize Wire library and join to I2C network
    dht.begin(); // DHT begins operation
    u8x8.begin(); // u8x8 begins operation
     u8x8.setPowerSave(0);  // Disable power save mode, 1 is enable. After enabling
power save mode, nothing will be seen on the screen
    u8x8.setFlipMode(1);
}

void loop() {
     float temp, humi; // Set variables temp and humi to floating point type, repre-
senting temperature and humidity respectively
    temp = dht.readTemperature(); // Read temperature value and store it in temp
    humi = dht.readHumidity(); // Read humidity value and store it in humi
    if (temp > 30 || humi < 40) {  // When the temperature is above 30 or the humid-
ity is below 40, if either condition is met, the buzzer will sound an alarm
        tone(buzzerPin, 200, 200);
    }

    u8x8.setFont(u8x8_font_chroma48medium8_r); // Set display font
    u8x8.setCursor(0, 33); // Set the position of the drawing cursor (0,33)
    u8x8.print("Temp:"); // Display "Temp:" at the position (0,33)
    u8x8.print(temp); // Then display the real-time temperature value
    u8x8.print("C"); // Then display the unit of temperature "C"
    u8x8.setCursor(0,50);
    u8x8.print("Humidity:");
    u8x8.print(humi);
    u8x8.print("%");
    u8x8.refreshDisplay();
    delay(200);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L8_dht20_alarm_XIAO_en

Step 2: Upload the program.

After writing the program, connect the XIAO main control board to the computer using a data cable, as shown in the image below:



After connection, click the “Verify” button to check the program. If the verification is successful, click the “Upload” button to upload the program to the hardware. When the debugging area shows “Upload Successful”, it is complete. To verify whether the alarm function runs smoothly, tightly grip the temperature and humidity sensor with your hand, observe the value change on the OLED display, and listen for the buzzer alarm when the temperature exceeds 30℃ .

Task 2-2: Use Grove DHT11 sensor to display temperature and humidity on the XIAO extension board’s OLED and add an alarm function.

For the Grove DHT11 sensor with a blue casing, the program is shown below:

```
#include "DHT.h"//Use DHT library
#include <Arduino.h>
#include <U8x8lib.h>//Use u8x8 library
#define DHTPIN 0
#define DHTTYPE DHT11//Specify using DHT11
DHT dht(DHTPIN, DHTTYPE);
int buzzerPin = A3;
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);//Set constructor
to connect OLED display

void setup() {
  pinMode(buzzerPin , OUTPUT);//Set buzzer pin to output mode
  Wire.begin();//Initialize wire library and join to I2C network
  dht.begin();//DHT begins operation
  u8x8.begin();//u8x8 begins operation
  u8x8.setPowerSave(0);  //Disable power save mode, 1 is enable. After enabling pow-
er save mode, nothing will be seen on the screen
  u8x8.setFlipMode(1);
}

void loop() {
  float temp, humi;//Set variables temp and humi to floating point type, representing
temperature and humidity respectively
```

```
    temp = dht.readTemperature();//Read temperature value and store it in temp
    humi = dht.readHumidity();//Read humidity value and store it in humi
    if (temp > 30 || humi < 40) {  //When the temperature is above 30 or the humidity
  is below 40, if either condition is met, the buzzer will sound an alarm
    tone(buzzerPin, 200, 200);
    }

    u8x8.setFont(u8x8_font_chroma48medium8_r);//Set display font
    u8x8.setCursor(0, 33);//Set the position of the drawing cursor (0,33)
    u8x8.print("Temp:");//Display "Temp:" at the position (0,33)
    u8x8.print(temp);//Then display the real-time temperature value
    u8x8.print("C");//Then display the unit of temperature "C"
    u8x8.setCursor(0,50);
    u8x8.print("Humidity:");
    u8x8.print(humi);
    u8x8.print("%");
    u8x8.refreshDisplay();
    delay(200);
  }
```

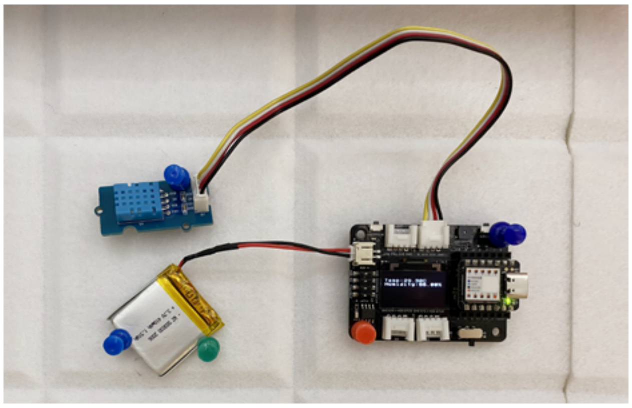
Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L8_dht11_alarm_XIAO_en

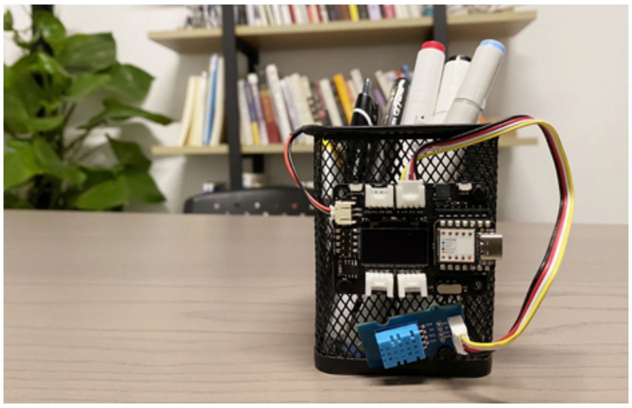
2.2.4 Appearance Design

Starting from this section, we will add the part of appearance design, beginning to explore the complete prototype product manufacturing. Initially, we can try to draw design sketches and make a simple modification with the materials at hand. Returning to the smart temperature and humidity meter in this section, please design the appearance of the prototype work based on the product characteristics and functions.

Product Name	**Smart Temperature and Humidity Meter**
Product Features	Small, portable, high sensitivity.
Product Functions	Real-time display of temperature and humidity values, and emits an alarm when temperature and humidity values exceed the comfortable range.
Product Appearance	(For example, made into a pendant to hang on the backpack that is carried around, stick on the tissue storage box in the bedroom, etc.)

Case reference





2.3
Surprise Gift Box Based on Light Sensor

Are you thinking about gifting a special birthday present to your friend? Instead of buying one, you can create it with the modules we have at hand. In this section, we are going to create a surprise gift box for a good friend. What kind of surprise will appear when the gift box is opened? What kind of modules do we need to complete such a surprise gift box? Start today's class with these questions.

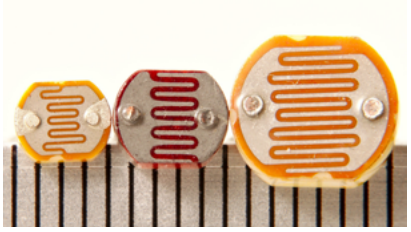
2.3.1 Background Knowledge

Light Sensor

Light sensors can detect the light intensity in the surrounding environment and convert the detected light energy into electrical energy. Light sensors are divided into types such as photoresistive, photodiode, and photoelectric transistor. Here, we will simply introduce two commonly used light sensors, photoresistive and photodiode.

Photoresistive Type

Firstly, the photoresistive type, its module will integrate a photoresistor, as shown below. The photoresistor is extremely sensitive to light, any light visible to our eyes can cause its reaction. High-intensity light will cause the resistance value to decrease, and low-intensity light will cause the resistance value to increase. By adjusting the resistance value in the circuit through the light intensity, it can control other devices, such as controlling the LED light on and off.



Photodiode Type

Photodiodes, also known as photoelectric sensors or photodetectors, when a beam of light hits the diode, the electrons in the tube will quickly scatter to form electron holes, thereby causing current to flow. The stronger the light, the stronger the current. Since the current generated by the photodiode is proportional to the intensity of light, it is very beneficial for light detection that requires a rapid change in light response. The light sensor we are going to use in this lesson is of this type.

Talking about the uses of light sensors, we can build a light-controlled switch through a light sensor, such as controlling the light on and off through a light sensor, turning off the light during the day, and turning on the light at night. The main purpose of the light control device is to save energy, improve efficiency through intelligent automation, the most common in life is probably the light control light, light control desk lamp, light control street lamp, highway tunnel lighting, etc., bringing convenience to our life and also contributing to environmental protection and energy conservation.

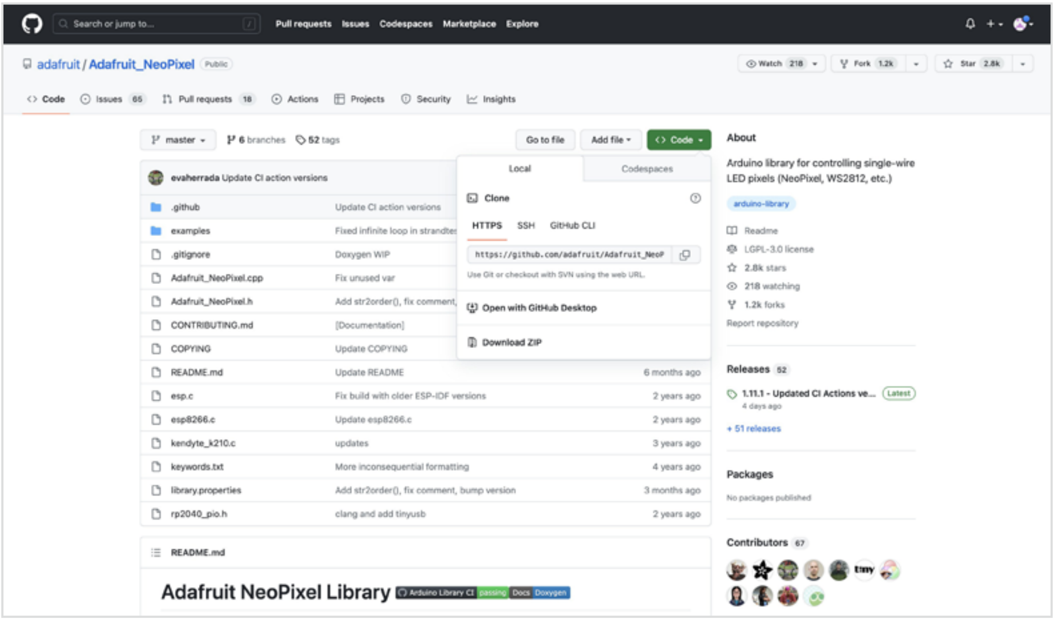
RGB LED Strip

The project in this class is paired with an RGB LED strip. The strip integrates multiple color-adjustable light beads. Compared with a single LED, it can achieve more lighting effects and cool visual impacts, making it ideal for creating surprises. RGB LED strips come in various styles and models. The one we are going to use is the Grove - WS2813 RGB LED Strip, 30-bead model. We can control the RGB LED strip to achieve a rich lighting effect through programming, and build more interesting lighting projects.

2.3.2 Task 1: Light up RGB LED Strip To get started with RGB LED strips, start by installing and understanding its library.

Add the `Adafruit_NeoPixel` Library

Before starting to program the RGB LED strip with the Arduino IDE, you need to add the necessary library files. Enter the library file address https://github.com/adafruit/Adafruit_NeoPixel in the browser address bar, enter the GitHub page, click `Code→Download ZIP` to download the resource package `Adafruit_NeoPixel-master.zip` to your local machine.



https://github.com/adafruit/Adafruit_NeoP

Use Git or checkout with SVN using the web URL.

Arduino library for controlling single-wire LED pixels (NeoPixel, WS2812, etc.)

evaherrada Update CI action versions

.github	Update CI action versions	
examples	Fixed infinite loop in strandte	
.gitignore	Doxygen WIP	
Adafruit_NeoPixel.cpp	Fix unused var	
Adafruit_NeoPixel.h	Add str2order(), fix comment,	
CONTRIBUTING.md	[Documentation]	
COPYING	Update COPYING	
README.md	Update README	6 months ago
esp.c	Fix build with older ESP-IDF versions	2 years ago
esp8266.c	Update esp8266.c	2 years ago
kendyte_k210.c	updates	3 years ago
keywords.txt	More inconsequential formatting	4 years ago
library.properties	Add str2order(), fix comment, bump version	3 months ago
rp2040_pio.h	clang and add tinyusb	2 years ago

Next, add the resource package `Adafruit_NeoPixel-master.zip` downloaded in the previous step via the menu bar `Sketch→Include Library→Add .ZIP` Library until you see the library loaded successfully.

Open the Simple Example

You can open the simple example through the following path: File → Examples → Adafruit NeoPixel → simple. Once the example program is opened, we can see the following program:

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
// Released under the GPLv3 license to match the rest of the
// Adafruit NeoPixel library

#include <Adafruit_NeoPixel.h>
```

```
#ifdef __AVR__
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN        6 // On Trinket or Gemma, suggest changing this to 1

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 16 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels

void setup() {
    // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
    // Any other board, you can remove this part (but no harm leaving it):
    #if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
    clock_prescale_set(clock_div_1);
    #endif
    // END of Trinket-specific code.

    pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

void loop() {
    pixels.clear(); // Set all pixel colors to ‘off’

    // The first NeoPixel in a strand is #0, second is 1, all the way up
    // to the count of pixels minus one.
    for(int i=0; i<NUMPIXELS; i++) { // For each pixel...

        // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
        // Here we’re using a moderately bright green color:
        pixels.setPixelColor(i, pixels.Color(0, 150, 0));

        pixels.show();   // Send the updated pixel colors to the hardware.

        delay(DELAYVAL); // Pause before next pass through loop
    }
}
```

This program allows the strip to light up 30 beads (green light) in sequence. This is a simple light strip example, and we need to modify some parameters:

`#define PIN 0`, you need to modify the pin connected to the light strip according to the actual situation. It is connected to the `A0` interface of the XIAO expansion board, so it is `PIN 0`.

`#define NUMPIXELS 30`, defines the number of LEDs in the light strip. Since the light strip has different models and the number of integrated beads is different, we use a light strip with 30 beads, so it is `NUMPIXELS 30`.

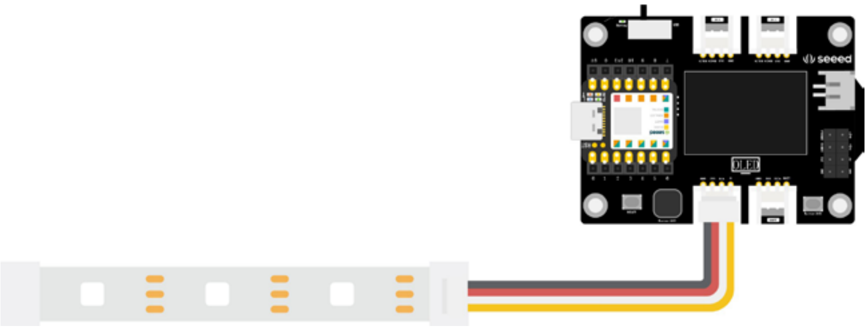
After modifying the parameters, you can remove the English comments for a clearer view of the code. It occupies a large amount of space.

```
#include <Adafruit_NeoPixel.h> // Header file, declaring the library
#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIN 0 // The light strip is connected to pin 0. If you are using XIAO RP2040,
please change 0 to A0
#define NUMPIXELS 30 // The number of LED lights on the light strip
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800); // Create a new
light strip object, define data mode
#define DELAYVAL 500 // The interval time for each light to light up

void setup() {
    #if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
    clock_prescale_set(clock_div_1);
    #endif
    pixels.begin(); // The light strip is ready to output data
}

void loop() {
    pixels.clear(); // All beads on the light strip are turned off
    for(int i=0; i<NUMPIXELS; i++) {
         pixels.setPixelColor(i, pixels.Color(0, 150, 0)); // Light up the beads in
sequence, the color is green
        pixels.show(); // Display the light strip
        delay(DELAYVAL);
    }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L9_NeoPixel30_simple_XIAO_en

In the code above, `pixels.Color(0,150,0)` is a function to set the color of the LED light strip. The numbers in the parentheses represent the three primary colors (red, green, blue) respectively. If it is `(0,150,0)`, it means that the brightness of red is `0`, the brightness of green is `150`, and the brightness of blue is `0`. The entire light strip will show a green effect. The larger the number, the brighter it will be, with a maximum of `255`. Next, connect the light strip to the `A0/D0` interface of the XIAO expansion board, as shown in the following figure:

Connect the XIAO main board to the computer with a data cable, and upload the program to the main board. After the upload is successful, observe the effect of the light strip.



The light strip can change color, flicker, and present various lighting effects such as breathing. We can refer to the sample program in the library: **File → Example → Adafruit NeoPixel → buttoncycler**. This sample program switches different lighting effects on the light strip through buttons. We can find the code for various lighting effects in it, such as flickering, rainbow lights, chasing, etc.

2.3.3 Project Making: Surprise Gift Box

Project Description

The program for the surprise gift box wants to realize: Use a light sensor to control the on and off of the RGB LED light strip, just like a light-controlled lamp, but the effect is opposite. When the value detected by the light sensor is less than a fixed value, that is, it is in a dim environment, the RGB LED light strip is off. When the value detected by the light sensor is greater than a fixed value, that is, in a bright environment, the RGB LED light strip lights up the rainbow light.

Program Writing

The program writing idea is as follows:

- Declare the files to be called, create a new light strip object, define the sensor pin and the number of LEDs on the light strip.
- Initialize the light strip and set the light sensor pin mode.
- Read the light value. If the light value is greater than 100, the light strip will present a rainbow and breathing light effect. Otherwise, the light strip will turn off.

The program is completed in two tasks:

Task 1: Make the Light Strip Present Rainbow and Breathing Light Effect

Step 1: Declare the files to be called, declare the light strip object, and define the pin and the number of LEDs on the light strip.

```
#include <Adafruit_NeoPixel.h> // Header file, declaring the library
#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIXEL_PIN 0 // The light strip is connected to pin A0. If you are using XIAO
RP2040, please change 0 to A0
#define PIXEL_COUNT 30 // The number of LED lights on the light strip
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800);
// Declare a new light strip object and define the data mode
```

Step 2: Initialize the light strip.

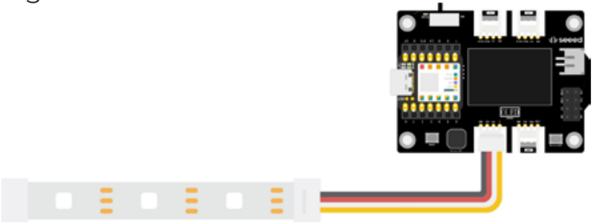
```
void setup() {
    strip.begin(); // Initialize the light strip, the light strip is ready to output
data
}
```

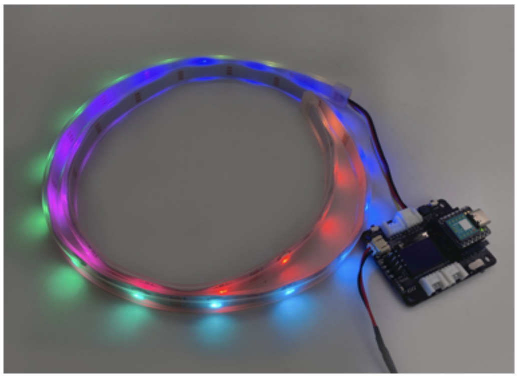
Step 3: The light strip presents a rainbow and breathing light effect. This part uses the `for()` function to present the breathing effect. For example, `for( i = 0; i<5; i ++ ){}` means that the initial value of `i` is `0`, when `i` is less than `5`, the statement in the loop body `{}` is run, each time the loop is run, i is incremented by `1`. This loop will run `5` times.

```
void loop() {
    strip.clear();// Turn off all the lights on the light strip
     rainbow(10);// The light strip displays a rainbow light effect. The number in
the parenthesis represents the speed of the rainbow light circulation. The smaller
the number, the faster the circulation speed
}
// The following is the code for the rainbow light effect, presenting the breathing
light effect. This code can be found in the example program buttoncycler
void rainbow(int wait) {
    for(long firstPixelHue = 0; firstPixelHue < 3*65536; firstPixelHue += 256) {
        for(int i=0; i<strip.numPixels(); i++) {
            int pixelHue = firstPixelHue + (i * 65536L / strip.numPixels());
            strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(pixelHue)));
        }
        strip.show(); // The light strip presents a light effect
        delay(wait);  // Delay
    }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L9_Rainbow_XIAO_en

Step 4: Connect the hardware and upload the program. First, connect the RGB LED light strip to the `A0/D0` interface of the XIAO expansion board, as shown in the figure:





Use a data cable to connect XIAO to the computer, click the “Upload” button, and upload the program to the hardware. When the debugging area shows “Upload successful”, you can observe the light effect of the light strip.

Task 2: Adding Light Control Switch Function

Step 1: Add code.

The added function is mainly to read the light value detected by the light sensor, and use the `if…else…` statement to judge the light value. When it is greater than 100 (this value can be adjusted according to the actual environment), the RGB LED light strip will show a rainbow breathing light effect.

Part of the program added:

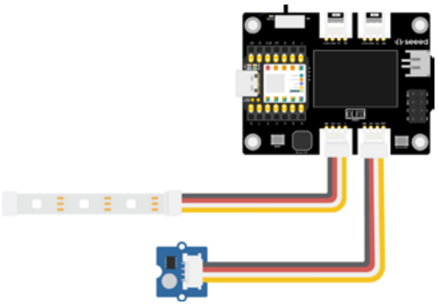
```
// This is an added part of the program, it cannot run directly
#define LIGHT_PIN 7// Define the light sensor connected to A7. If you are using XIAO
RP2040, please change 7 to A3. If you are using XIAO BLE, please change 7 to 5
#define PIXEL_PIN 0// Define light strip. If you are using XIAO RP2040, please change
0 to A0
int readValue = 0;// Define the variable readValue to store the light value
void setup() {
    pinMode(LIGHT_PIN , INPUT); // Set the pin of the light sensor as input status
}
void loop() {
     readValue = analogRead(A7);// Read the analog value of the A7 pin light and
store it in the readValue variable. If you are using XIAO RP2040, please change A7
to A3. If you are using XIAO BLE, please change A7 to A5
     if(readValue > 500){ // Condition judgment, if the light value is greater than
500, then the light strip presents a rainbow light effect, otherwise, the light
strip is turned off
        rainbow(10);
    }else {
        strip.clear();
        strip.show();
    }
}
```

We add the entered statement to the corresponding position of Task 1's program. See the complete program:

```
#include <Adafruit_NeoPixel.h>// Header file, declare library
#ifdef __AVR__
#include <avr/power.h>
#endif
#define LIGHT_PIN 7// Define the light sensor connected to A7. If you are using XIAO
RP2040, please change 7 to A3. If you are using XIAO BLE, please change 7 to 5
#define PIXEL_PIN 0 // The light strip is connected to the A0 pin. If you are using
XIAO RP2040, please change 0 to A0
#define PIXEL_COUNT 30 // The number of LEDs on the light strip
int readValue = 0;// Define variable readValue to store light values
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800);
// Declare the light strip object and define the data mode
void setup() {
  strip.begin(); // Initialize the light strip and prepare the light strip to output
data
  pinMode(LIGHT_PIN , INPUT); // Set the pin of the light sensor to input state
}
void loop() {
  strip.clear();// Turn off all the beads on the light strip
  rainbow(10);// The light strip shows a rainbow light effect. The number in the pa-
rentheses represents the speed of the rainbow light rotation. The smaller the num-
ber, the faster the rotation speed
  readValue = analogRead(A7);// Read the analog value of the light on the A7 pin and
store it in the readValue variable. If you are using XIAO RP2040, please change A7
to A3. If you are using XIAO BLE, please change A7 to A5
    if(readValue > 500){ // Conditional judgment, if the light value is greater than
500, then the light strip presents a rainbow light effect, otherwise, the light
strip is turned off
        rainbow(10);
    }else {
        strip.clear();
        strip.show();
    }
}
// The following is the code for the rainbow light effect, presenting the breathing
light effect, this code can be found in the sample program buttoncycler
void rainbow(int wait) {
  for(long firstPixelHue = 0; firstPixelHue < 3*65536; firstPixelHue += 256) {
    for(int i=0; i<strip.numPixels(); i++) {
      int pixelHue = firstPixelHue + (i * 65536L / strip.numPixels());
      strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(pixelHue)));
    }
    strip.show(); // The light strip presents a light effect
```

```
      delay(wait);  // Delay
    }
  }
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L9_StripLight_XIAO_en



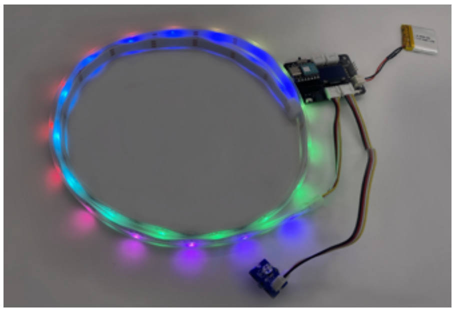
Step 2: Connect the hardware and upload the program. First, connect the RGB LED light strip to the `A0` interface of the XIAO expansion board, and connect the light sensor to the `A7` interface, as shown in the figure below:

- Attention -

If you are using XIAO BLE, please connect the light sensor to the I2C interface of the XIAO expansion board.

If you are using XIAO RP2040, due to the limited number of pins exposed, you need to connect the SIG pin of the light sensor and the A3 pin of XIAO RP2040 with Dupont wires on your own.

Next, connect XIAO to your computer with a data cable, click the “Upload” button in the Arduino IDE to upload the program to the hardware. When the debugging area shows “Upload successful”, you can cover the light sensor with your hand, then release the light sensor, and observe the changes in the light strip. Note that because it takes a certain amount of time for the light strip to display light effects, the light strip will not turn off immediately when you cover the light sensor.



2.3.4 Exterior Design

Combining the program design of the surprise gift box, when the light sensor is in a dim environment, the RGB LED light strip is off, and when the light sensor is in a bright environment, the RGB LED light strip lights up with rainbow lights. We can imagine that the electronic part is placed in a closed box, which can match the function implemented by the program and can also meet the positioning of the gift. Of course, you can also have other designs.

Product Name	**Surprise Gift Box**
Product Features	Cool light effects, photocontrol, surprise, birthday
Product Functions	Control the lighting of the RGB LED light strip with a light sensor
Product Appearance	

Case reference

2.4 Rhythmic Dance with a Triaxial Accelerometer

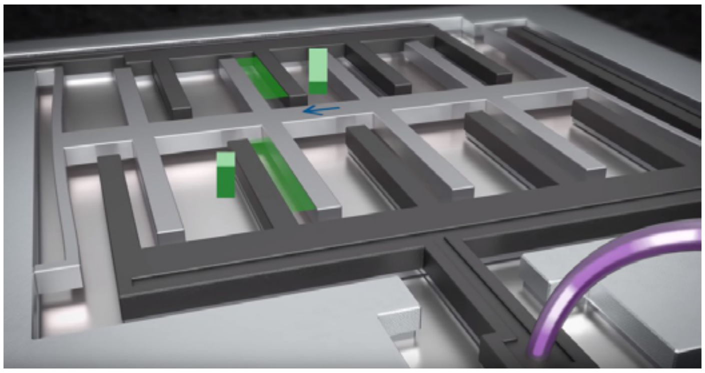
When we use smartphones or tablets, we notice that the screen display automatically flips depending on whether the device is vertical or horizontal. In racing or flying games, phones and tablets can be used as steering wheels, with turning accomplished by tilting the device. Increasingly popular drones, for the most part, can now fly more and more steadily by detecting and controlling the attitude of the aircraft. All of these feats are thanks to the triaxial accelerometer. In this section, we will learn to use programming to retrieve data from a triaxial accelerometer and use this data for display and control.

2.4.1 Background Knowledge

Triaxial Accelerometer

As people pay more and more attention to their health, an increasing number of people are starting to wear wristbands, pedometers, or use smartphones to record their steps, which has become a lifestyle habit for many. So how exactly does a pedometer work? The answer lies in a tiny chip called a triaxial accelerometer which is usually found in modern phones or wristbands and is the key component in step counting. An accelerometer is a sensor capable of measuring acceleration. It usually consists of a mass, a damper, an elastic element, a sensitive element, and an adjustment circuit. During acceleration, the sensor measures the inertial force applied to the mass, and uses Newton's second law to determine acceleration. Depending on the different sensitive elements of the sensor, common accelerometers include capacitive, inductive, strain, piezoresistive, piezoelectric, and others.

The capacitive accelerometer, based on the principle of capacitance, is a common type of accelerometer and is indispensable in certain fields, such as safety airbags, mobile devices like phones, etc. Capacitive accelerometers employ Micro-Electro-Mechanical Systems (MEMS) technology, which makes them very economical when mass-produced, thus ensuring low cost.



Applications of Accelerometers

Accelerometers can help robots understand their environment. Are they climbing a hill? Or going downhill, or have they fallen? For balance cars or drones, accelerometers can help them maintain balance. In addition to everyday areas like smartphones and health wristbands, accelerometers have also found wide application in other fields.

Accelerometers in seismic probe design: Seismic detectors are special sensors used for geological exploration and engineering measurements. They are sensors that convert ground vibration into electrical signals, turning ground movement caused by seismic waves into electrical signals, which are then converted into binary data through an analog/digital converter, organized, stored, and processed.

Monitoring high-voltage line dancing: Currently, domestic monitoring of line dancing mainly adopts two main technical schemes: video image acquisition and motion acceleration measurement. The former requires high reliability and stability of video equipment under high-temperature, high-humidity, severe cold, dense fog, dust storms, and other weather conditions, and the effects of the video images taken will also be affected. Hence, it can only serve as auxiliary monitoring means and cannot quantitatively analyze line motion parameters. Using an accelerometer to monitor line dancing allows for quantitative analysis of the up-and-down vibration and left-right swing of transmission lines at a certain point, but it can only measure the amplitude and frequency of line linear motion and not accurately measure complex circular motion.

Automotive Safety: Accelerometers are mainly used in automotive safety airbags, anti-lock braking systems, traction control systems, and other safety features. In safety applications, the rapid response of the accelerometer is crucial. It must be quickly determined when a safety airbag should deploy, so the accelerometer must respond instantly. A sensor design that can quickly reach a stable state rather than continuing to vibrate can shorten the device's response time.

Drones: Accelerometers are also key components of drone control, positioning, and stability.

Game Control: Accelerometers can detect changes in the tilt angles up, down, left, and right, so it becomes straightforward to control the directions of objects in games by tilting handheld devices forward and backward. Many new game console controllers and VR device controllers incorporate accelerometers.

Image Auto-flip: Accelerometers detect the rotation movements and directions of handheld devices, making the displayed images upright.

Compensation for GPS Navigation System Blind Spots: GPS systems determine an object's location by receiving signals from three satellites distributed at 120 degrees. In special circumstances and terrains, like tunnels, dense buildings, jungle areas, GPS signals may weaken or even be completely lost, creating blind spots. By adding an accelerometer and using inertial navigation we previously mentioned, we can measure system dead zones. By integrating the accelerometer once, we change it into the speed change per unit time, thereby measuring the movement of an object in the dead zone.

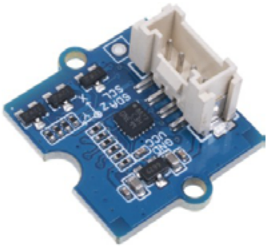
Pedometer Function: Accelerometers can detect AC signals and object vibrations. When people walk, they produce regular vibrations, and the accelerometer can detect the zero crossing of the vibrations, thereby calculating the number of steps walked or run, and thus calculating the displacement moved by the person. Using certain formulas, we can also calculate the calories burned.

Image Stabilization and Shooting Stabilizers: The image stabilization function uses an accelerometer to detect the vibration/swing amplitude of handheld devices, and when the vibration/swing amplitude is too large, it locks the camera shutter to ensure the images taken are always clear. The shooting stabilizer uses an accelerometer to maintain the stability of the entire device.

Hard Drive Protection: By detecting the state of free fall with an accelerometer, necessary protection can be implemented for hard drives. It is well known that when a hard drive is reading data, the gap between the read/write head and the platter is minuscule, and even minor external vibrations can have severe consequences for the hard drive, leading to data loss. By using an accelerometer, the state of free fall can be detected. When the state of free fall is detected, the read/write head is reset to reduce the extent of hard drive damage.

Grove Three-Axis Accelerometer

In our kit, we have a three-axis accelerometer module - Grove Three-Axis Accelerometer Module. This tiny, incredible three-axis accelerometer supports I2C, SPI, and ADC GPIO interfaces, which means you can choose any way to connect to your development board. Additionally, the accelerometer can also monitor the surrounding temperature to adjust for errors caused by it.



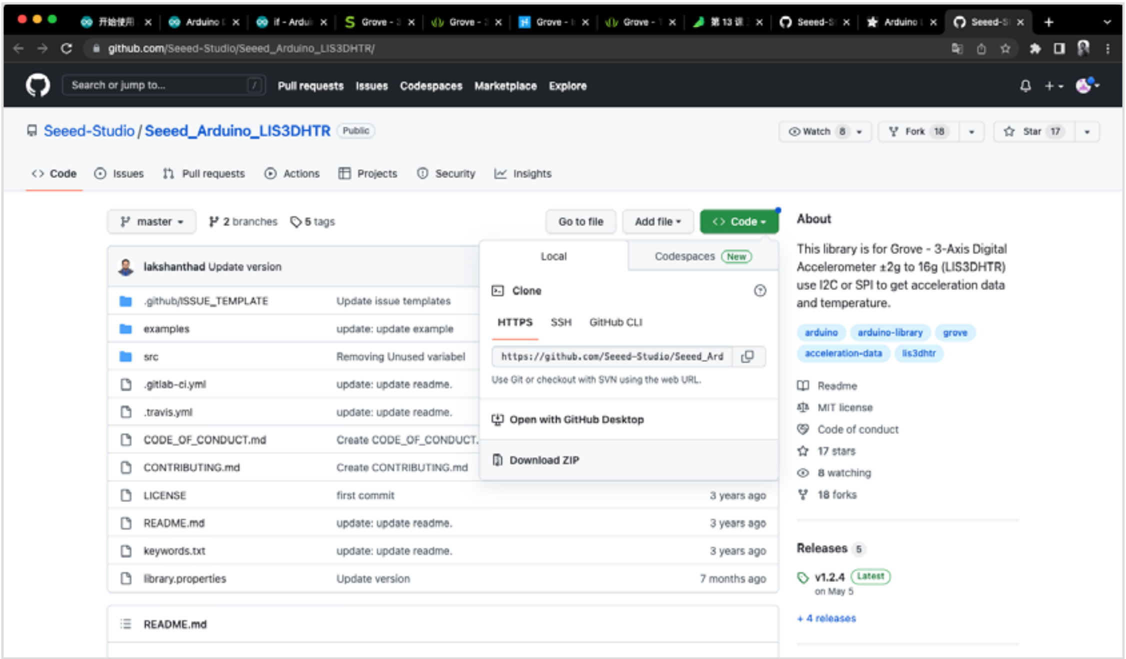
2.4.2 Task1: Reading Values from the XYZ Axes of the Three-Axis Accelerometer

The key to using a three-axis accelerometer for project creation is learning how to read the values of the X, Y, Z axes of the accelerometer.

Add the Seeed_Arduino_LIS3DHTR Library

Before starting to program the Grove Three-Axis Accelerometer with the Arduino IDE, it is necessary to add the required library for the sensor. Enter the library address https://github.

com/Seeed-Studio/Seeed_Arduino_LIS3DHTR/ in the browser address bar, enter the GitHub page, click `Code→Download ZIP` to download the resource package `Seeed_Arduino_LIS3DHTR-master.zip` to local, as shown in the figure below.



```
https://github.com/Seeed-Studio/Seeed_Ard
```

Use Git or checkout with SVN using the web URL.

This library is for Grove - 3-Axis Digital Accelerometer ±2g to 16g (LIS3DHTR) use I2C or SPI to get acceleration data and temperature.

lakshanthad Update version		
.github/ISSUE_TEMPLATE	Update issue templates	
examples	update: update example	
src	Removing Unused variabel	
.gitlab-ci.yml	update: update readme.	
.travis.yml	update: update readme.	
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.	
CONTRIBUTING.md	Create CONTRIBUTING.md	
LICENSE	first commit	3 years ago
README.md	update: update readme.	3 years ago
keywords.txt	update: update readme.	3 years ago
library.properties	Update version	7 months ago

Add the previously downloaded resource package `Seeed_Arduino_LIS3DHTR-master.zip` through `Sketch→Include Library→Add .ZIP` Library in the menu bar, until you see a library load successful prompt.

Open the Sample File

Similarly, you can refer to the library file and open the `LIS3DHTR_IIC` sample through the following path: **File→Examples→Grove-3-Axis-Digital-Accelerometer-2g-to-16g-LIS3DHTR→LIS3DHTR_IIC**.

```
// This example use I2C.
#include “LIS3DHTR.h”
#include <Wire.h>
LIS3DHTR<TwoWire> LIS; //IIC
#define WIRE Wire

void setup()
{
  Serial.begin(115200);
  while (!Serial)
  {
  };
  LIS.begin(WIRE); //IIC init dafault :0x18
  //LIS.begin(WIRE, 0x19); //IIC init
   LIS.openTemp();  //If ADC3 is used, the temperature detection needs to be turned
off.
  //  LIS.closeTemp();//default
  delay(100);
  //  LIS.setFullScaleRange(LIS3DHTR_RANGE_2G);
  //  LIS.setFullScaleRange(LIS3DHTR_RANGE_4G);
  //  LIS.setFullScaleRange(LIS3DHTR_RANGE_8G);
  //  LIS.setFullScaleRange(LIS3DHTR_RANGE_16G);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_1HZ);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_10HZ);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_25HZ);
  LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_100HZ);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_200HZ);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_1_6KHZ);
  //  LIS.setOutputDataRate(LIS3DHTR_DATARATE_5KHZ);
  LIS.setHighSolution(true); //High solution enable
}
void loop()
{
  if (!LIS)
  {
    Serial.println(“LIS3DHTR didn’t connect.”);
    while (1)
      ;
    return;
  }
  //3 axis
  //  Serial.print(“x:”); Serial.print(LIS.getAccelerationX()); Serial.print(“  “);
  //  Serial.print(“y:”); Serial.print(LIS.getAccelerationY()); Serial.print(“  “);
  //  Serial.print(“z:”); Serial.println(LIS.getAccelerationZ());
  //ADC
  //    Serial.print(“adc1:”); Serial.println(LIS.readbitADC1());
  //    Serial.print(“adc2:”); Serial.println(LIS.readbitADC2());
  //    Serial.print(“adc3:”); Serial.println(LIS.readbitADC3());

  //temperature
  Serial.print(“temp:”);
  Serial.println(LIS.getTemperature());
  delay(500);
}
```

The sample program can read the values of the X, Y, Z axes of the three-axis accelerometer and output through the serial monitor. The sample program provides us with different setting choices using the “//” comment method, but you need to manually select the required parts, as follows:

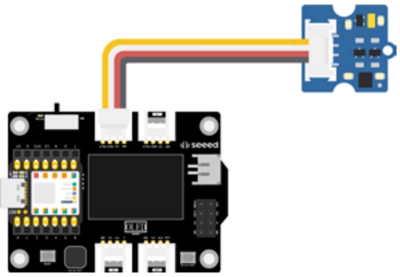
`LIS.begin(WIRE)`: initializes the default values, you can choose between 0×18 and 0×19, we need to choose `LIS.begin(WIRE,0×19);`.

`LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ)`: The accelerometer’s output rate has multiple choices, choose 50Hz. The three-axis accelerometer can also monitor the ambient temperature, we temporarily do not need to delete the related code, the complete program is as follows:

```
// This example shows the 3-axis acceleration.
#include “LIS3DHTR.h” // Declare library
#include <Wire.h>
LIS3DHTR<TwoWire> LIS;
#define WIRE Wire // Initialize the module above using hardware I2C

void setup()
{
    Serial.begin(9600);
    while (!Serial) { }; // If you can’t open the serial monitor, the code will stop
here
    LIS.begin(WIRE, 0x19); // Initialize I2C with default value
    delay(100);
    LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ); // Set the accelerometer’s output
rate to 50Hz.
}
void loop()
{
    if (!LIS) {
        Serial.println(“LIS3DHTR didn’t connect.”);
        while (1);
        return;
    }
    // Read the values of the X, Y, Z axes from the sensor, and display them on the
serial monitor
    Serial.print(“x:”); Serial.print(LIS.getAccelerationX()); Serial.print(“  “);
    Serial.print(“y:”); Serial.print(LIS.getAccelerationY()); Serial.print(“  “);
    Serial.print(“z:”); Serial.println(LIS.getAccelerationZ());
    delay(500);
}
```

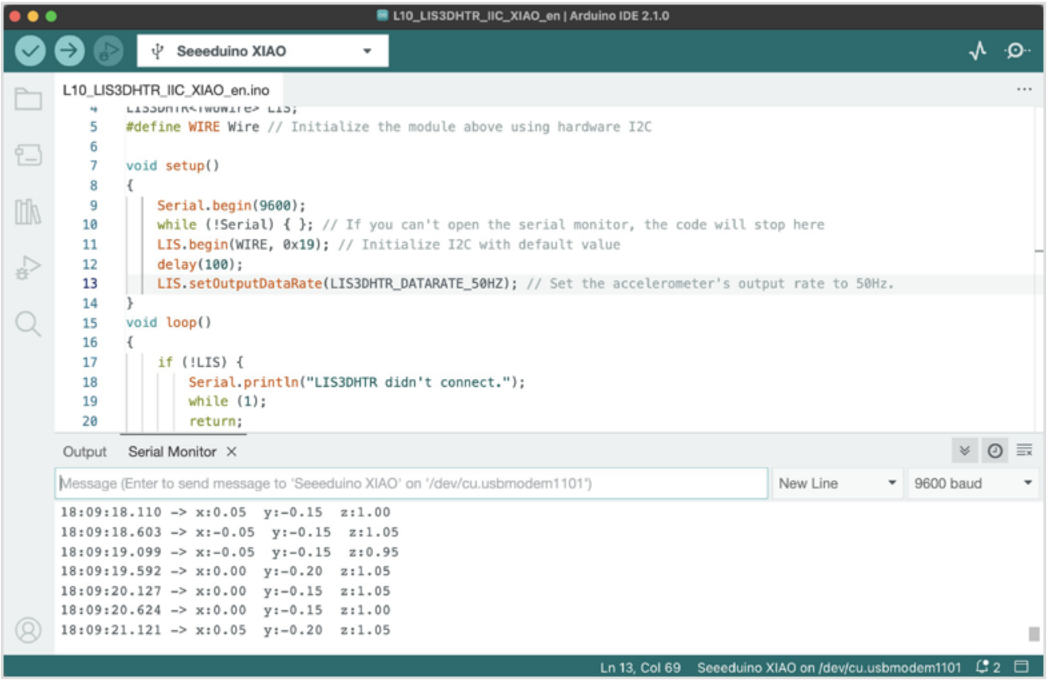
Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L10_LIS3DHTR_IIC_XIAO_en

Next, connect the three-axis accelerometer to the I2C interface. There are two I2C interfaces on the XIAO expansion board, as shown in the picture below:



Monitor Data Changes via Serial Monitor

Connect XIAO to your computer with a data cable, upload the program, wait for the program to upload successfully, then open the serial monitor. Move the three-axis accelerometer in the X, Y, Z axis direction and observe the changes in the readings.



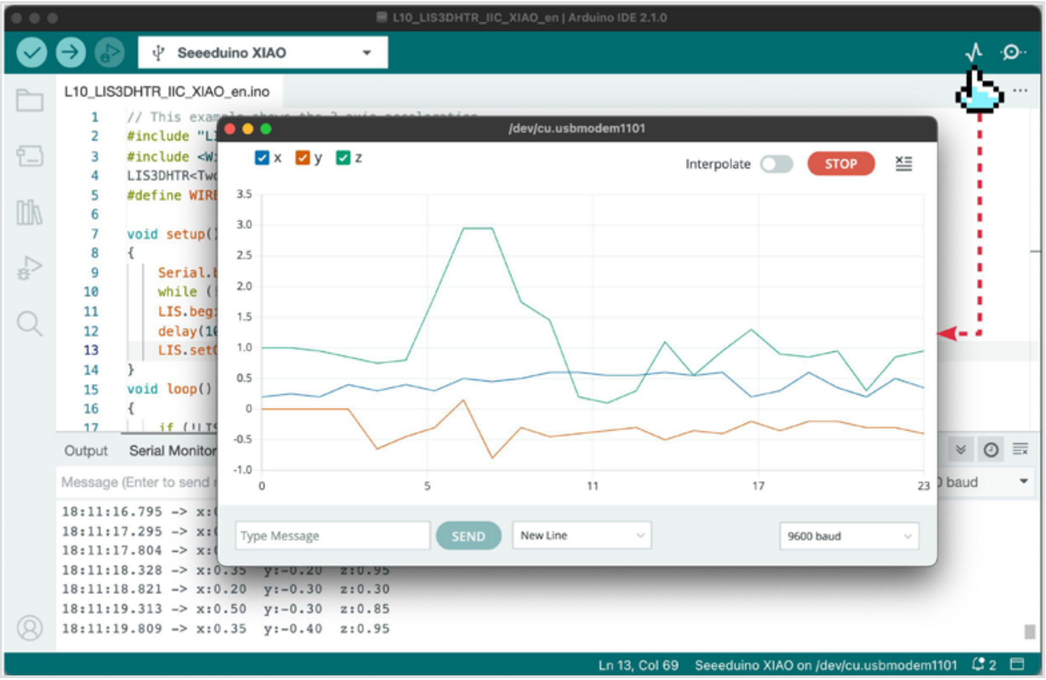
L10_LIS3DHTR_IIC_XIAO_en | Arduino IDE 2.1.0
Seeeduino XIAO
L10_LIS3DHTR_IIC_XIAO_en.ino
Output
Serial Monitor
Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem1101')
New Line
9600 baud
Ln 13, Col 69
Seeeduino XIAO on /dev/cu.usbmodem1101
2

```
LIS3DHTR<TwoWire> LIS;
#define WIRE Wire // Initialize the module above using hardware I2C

void setup()
{
    Serial.begin(9600);
    while (!Serial) { }; // If you can't open the serial monitor, the code will stop here
    LIS.begin(WIRE, 0x19); // Initialize I2C with default value
    delay(100);
    LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ); // Set the accelerometer's output rate to 50Hz.
}
void loop()
{
    if (!LIS) {
        Serial.println("LIS3DHTR didn't connect.");
        while (1);
        return;
```

```
18:09:18.110 -> x:0.05  y:-0.15  z:1.00
18:09:18.603 -> x:-0.05  y:-0.15  z:1.05
18:09:19.099 -> x:-0.05  y:-0.15  z:0.95
18:09:19.592 -> x:0.00  y:-0.20  z:1.05
18:09:20.127 -> x:0.00  y:-0.15  z:1.05
18:09:20.624 -> x:0.00  y:-0.15  z:1.00
18:09:21.121 -> x:0.05  y:-0.20  z:1.05
```

Monitor Data Changes with Serial Plotter

The numerical way of presenting the changes in the accelerometer's 3-axis values is not very intuitive. You can open the serial plotter, as shown in the picture below.



```
// This exam
#include "LI
#include <Wi
LIS3DHTR<Two
#define WIRE

void setup()
{
    Serial.b
    while (
    LIS.beg
    delay(1
    LIS.set
}
void loop()
{
    if (!LIS
```

```
18:11:16.795 -> x:
18:11:17.295 -> x:
18:11:17.804 -> x:
18:11:18.328 -> x:0.35  y:-0.20  z:0.95
18:11:18.821 -> x:0.20  y:-0.30  z:0.30
18:11:19.313 -> x:0.50  y:-0.30  z:0.85
18:11:19.809 -> x:0.35  y:-0.40  z:0.95
```

2.4.3 Project Production: Rhythmic Dance

Project Description

We can add an RGB LED strip in the project to achieve cool light effects changes. The three-axis accelerometer is used to detect movement, and different light effects are triggered based on different values on the X, Y, Z axes of the accelerometer.

Program Writing

To control the RGB LED strip to change the light effects via the three-axis accelerometer, follow these steps:

- Declare the library files that need to be invoked, define the strip pin and LED quantity.
- Initialize the three-axis accelerometer and the strip.
- Set the light effect of the strip to red, green and blue flashing, set the conditional judgment, and control the change by different value intervals on the X, Y, Z axis of the three-axis accelerometer.

Task: Control RGB LED Strip to Change Light Effects via Three-Axis Accelerometer

Step 1: Declare the library files that need to be invoked, define the strip pin and the number of LEDs.

```
#include "LIS3DHTR.h" // Declare the library file of the three-axis accelerometer
#include <Adafruit_NeoPixel.h> // Declare the strip's library file
#ifdef __AVR__
```

```
#include <avr/power.h>
#endif
// Below are to initialize the module using software I2C or hardware I2C
#ifdef SOFTWAREWIRE
#include <SoftwareWire.h>
SoftwareWire myWire(3, 2);
LIS3DHTR<SoftwareWire> LIS;
#define WIRE myWire
#else
#include <Wire.h>
LIS3DHTR<TwoWire> LIS;
#define WIRE Wire
#endif

#define PIXEL_PIN 0 // Define the pin of the strip, if you use XIAO RP2040/XIAO ESP32,
please modify 0 to A0
#define PIXEL_COUNT 30 // Define the number of LEDs in the strip as 30
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800); // Declare
the strip object, set the data type
```

Step 2: Initialize the three-axis accelerometer and the strip. Here, you need to initialize the accelerometer and set the rate to 50HZ.

```
void setup() {
    Serial.begin(9600); // Initialize the serial monitor
     while (!Serial) {}; // If the serial monitor isn't opened, the code will stop
here, so please open the serial monitor
    LIS.begin(WIRE, 0x19); // Initialize I2C
    delay(100);
    LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ); // Set the accelerometer's output
rate to 50Hz
    strip.begin(); // Start the strip
    strip.show(); // Display the strip
}
```

Step 3: Set the light effects to flash in red, green, and blue, respectively. Conditionals are used to change the color of the light strip according to the varying readings on the X, Y, and Z axes of the 3-axis accelerometer. These readings can be viewed via the serial monitor. By observing the change in values when the accelerometer is moved along the X, Y, and Z axes, we can determine the appropriate settings for the light strip. Since the readings may sometimes be negative, we take the absolute value of the readings. The `abs()` function can be used to get the absolute value, for example, `abs(LIS.getAccelerationX())` would give the absolute value of the reading on the X-axis.

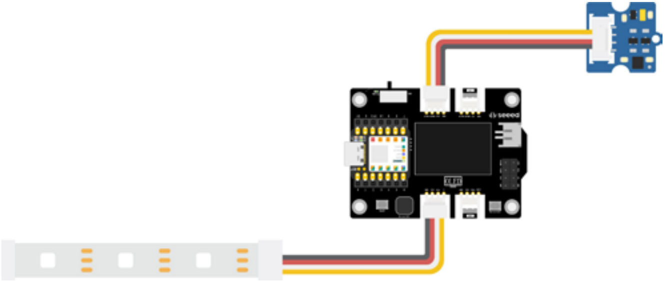
```
void loop() {
    if (!LIS) {  // Check if the 3-axis accelerometer is connected properly
        Serial.println(“LIS3DHTR didn’t connect.”);
        while (1);
        return;
    }

    if ((abs(LIS.getAccelerationX()) > 0.2)) {
        theaterChase(strip.Color(127, 0, 0), 50); // The light strip turns red
    }
    if ((abs(LIS.getAccelerationY()) > 0.2)) {
        theaterChase(strip.Color(0, 127, 0), 50); // The light strip turns green
    }
    if ((abs(LIS.getAccelerationZ()) > 1.0)) {
        theaterChase(strip.Color(0, 0, 127), 50); // The light strip turns blue
    }
    else
    {
        strip.clear();
        strip.show();
    }

     // Read the values of the X, Y, and Z axes from the sensor and display them on
the serial monitor
    Serial.print(“x:”); Serial.print(LIS.getAccelerationX()); Serial.print(“  “);
    Serial.print(“y:”); Serial.print(LIS.getAccelerationY()); Serial.print(“  “);
    Serial.print(“z:”); Serial.println(LIS.getAccelerationZ());

    delay(500);
}
// Set theaterChase for flashing light effects
void theaterChase(uint32_t color, int wait) {
    for(int a=0; a<10; a++) {
        for(int b=0; b<3; b++) {
            strip.clear();
            for(int c=b; c<strip.numPixels(); c += 3) {
                strip.setPixelColor(c, color);
            }
            strip.show();
            delay(wait);
        }
    }
}
```

Complete program as follows:

```
#include “LIS3DHTR.h”// Declare the library file for the 3-axis accelerometer
#include <Adafruit_NeoPixel.h>// Declare the library file for the light strip
#ifdef __AVR__
#include <avr/power.h>
#endif
// The following is to initialize the module using software I2C or hardware I2C
#ifdef SOFTWAREWIRE
#include <SoftwareWire.h>
SoftwareWire myWire(3, 2);
LIS3DHTR<SoftwareWire> LIS;
#define WIRE myWire
#else
#include <Wire.h>
LIS3DHTR<TwoWire> LIS;
#define WIRE Wire
#endif

#define PIXEL_PIN 0 // Define the pin of the light strip, if you are using XIAO
RP2040/XIAO ESP32, please change 0 to A0
#define PIXEL_COUNT 30 // Define the number of LEDs on the light strip as 30
Adafruit_NeoPixel strip(PIXEL_COUNT, PIXEL_PIN, NEO_GRB + NEO_KHZ800); // Declare
the light strip object and set the data type

void setup() {
    Serial.begin(9600); // Initialize the serial monitor
    while (!Serial) {};// If you do not open the serial monitor, the code will stop 
here, so please open the serial monitor
    LIS.begin(WIRE, 0x19); // IIC initialization
    delay(100);
    LIS.setOutputDataRate(LIS3DHTR_DATARATE_50HZ); // Set the output rate of the ac-
celerometer to 50Hz
    strip.begin(); // The light strip starts working
    strip.show(); // The light strip displays
}
void loop() {
    if (!LIS) {  // Check if the 3-axis accelerometer is connected correctly
        Serial.println(“LIS3DHTR didn’t connect.”);
        while (1);
        return;
    }

    if ((abs(LIS.getAccelerationX()) > 0.2)) {
        theaterChase(strip.Color(127, 0, 0), 50); // The light strip turns red
    }
    if ((abs(LIS.getAccelerationY()) > 0.2)) {
        theaterChase(strip.Color(0, 127, 0), 50); // The light strip turns green
    }
    if ((abs(LIS.getAccelerationZ()) > 1.0)) {
        theaterChase(strip.Color(0, 0, 127), 50); // The light strip turns blue
    }
    else
    {
        strip.clear();
        strip.show();
    }

    // Read the values of the X, Y, and Z axes from the sensor and display them on 
the serial monitor
    Serial.print(“x:”); Serial.print(LIS.getAccelerationX()); Serial.print(“  “);
    Serial.print(“y:”); Serial.print(LIS.getAccelerationY()); Serial.print(“  “);
    Serial.print(“z:”); Serial.println(LIS.getAccelerationZ());

    delay(500);
}
// Set theaterChase for flashing light effects
void theaterChase(uint32_t color, int wait) {
    for(int a=0; a<10; a++) {
        for(int b=0; b<3; b++) {
            strip.clear();
            for(int c=b; c<strip.numPixels(); c += 3) {
                strip.setPixelColor(c, color);
            }
            strip.show();
            delay(wait);
        }
    }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L10_MovementRGBLED_XIAO_en

Step 4: Connect the hardware and upload the program. First, connect the RGB LED light strip to the `A0/D0` interface of the XIAO expansion board, and the three-axis accelerometer to the I2C interface, as shown in the figure:



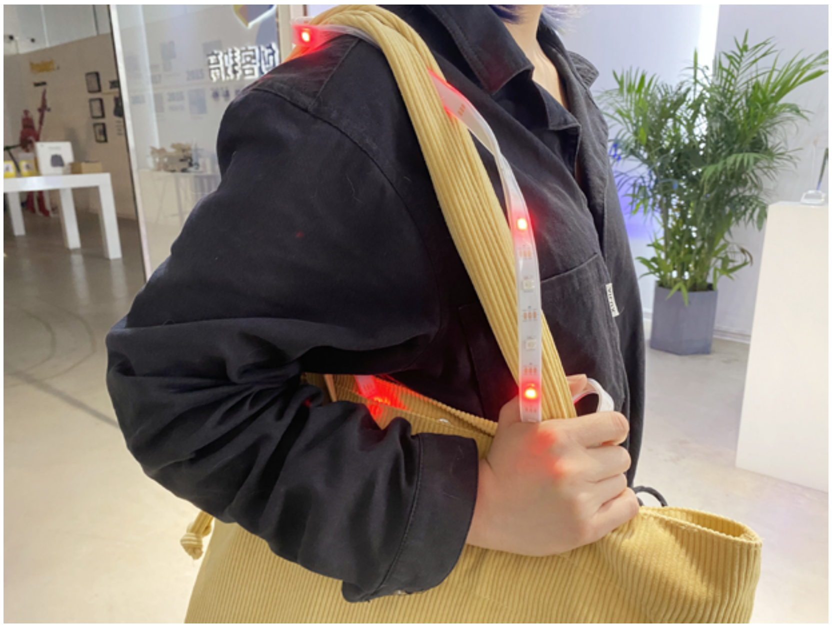
Use a data cable to connect XIAO to your computer, click the "Upload" button in Arduino IDE, and upload the program to the hardware. Once the debugging area shows "Upload Successful", you can open the serial monitor and try swinging the three-axis accelerometer left, right, up, and down to feel the light effect changes of the light strip.

2.4.4 Exterior Design

Imagine how cool it would be if there were lights flashing with your dance steps as you passionately swing your arms. That's where the inspiration for Rhythm Dance comes from. It can be combined with clothes or accessories to create a wearable style.

Product Name	**Rhythm Dance**
Product Features	Wearable, Cool light effects, Posture detection
Product Functions	RGB LED light strip displays different light effects based on the values detected by the three-axis accelerometer
Product Appearance	(For example: The waterproof layer on the outside of the RGB LED light strip can be removed, and it can be sewn together with clothes or a belt, etc.)

Reference for the case



Chapter 3:
Intermediate Project Practice - Complex Projects

In this unit, we will delve into more intricate and comprehensive projects, striving towards mature works in terms of program implementation and design of appearance structure. These include miniaturized smart homes, wearable electronic devices, interactive electronic instruments, Wi-Fi connectivity, and applications enabled by XIAO ESP32C3 or telemetry and command via the MQTT protocol. We will provide the laser-cut design blueprints for the first three cases for your reference. Of course, you're not limited to these examples; you could use other, more accessible materials, such as corrugated cardboard or cardstock, for crafting. Feel free to unleash your creativity and design the work you wish to present!

3.1
Smart Remote Control Door

In everyday life, privacy and security are matters of great concern to everyone. In recent years, the doors of residential areas have become increasingly smart, only accessible through electronic keys or passwords, preventing outsiders from gaining entry. In public areas or parking lot entrances, having a smart remote control to operate doors could facilitate the work of security personnel. A simple smart remote control door can be implemented using an infrared transmitter and an infrared receiver, which sends and receives infrared signals to open and close the door. In this section, we will build such a smart remote control door.

3.1.1 Background Knowledge

Infrared Receivers and Transmitters

An infrared receiver is used to receive infrared signals, and is also used for remote control detection. The infrared receiver has an infrared detector for picking up the infrared light emitted by the infrared transmitter. The Grove - IR Infrared Receiver Module has a range of 10 meters; signals cannot be received beyond this effective range. Generally, the infrared receiver and infrared transmitter work together.

An infrared transmitter is a type of remote control device with a remote control function. It emits light through an infrared emission tube within a certain range, thereby achieving control signal functions. The remote controls we use to control TVs, air conditioners, and car doors in daily life are infrared transmitters. Common infrared transmitters include modular ones like the Grove - Infrared Emitter Module, as well as regular remote controls, each with corresponding usage scenarios and methods. For our smart remote control door, we will be using an infrared remote control.

To explain simply, the principle of infrared transmission and reception is that the infrared transmitter inputs the signal, amplifies it, and sends it through the infrared transmission tube. The infrared receiver then receives this infrared signal, amplifies it and converts it back to an electrical signal, thereby realizing infrared control.

Grove - IR Infrared Receiver Module

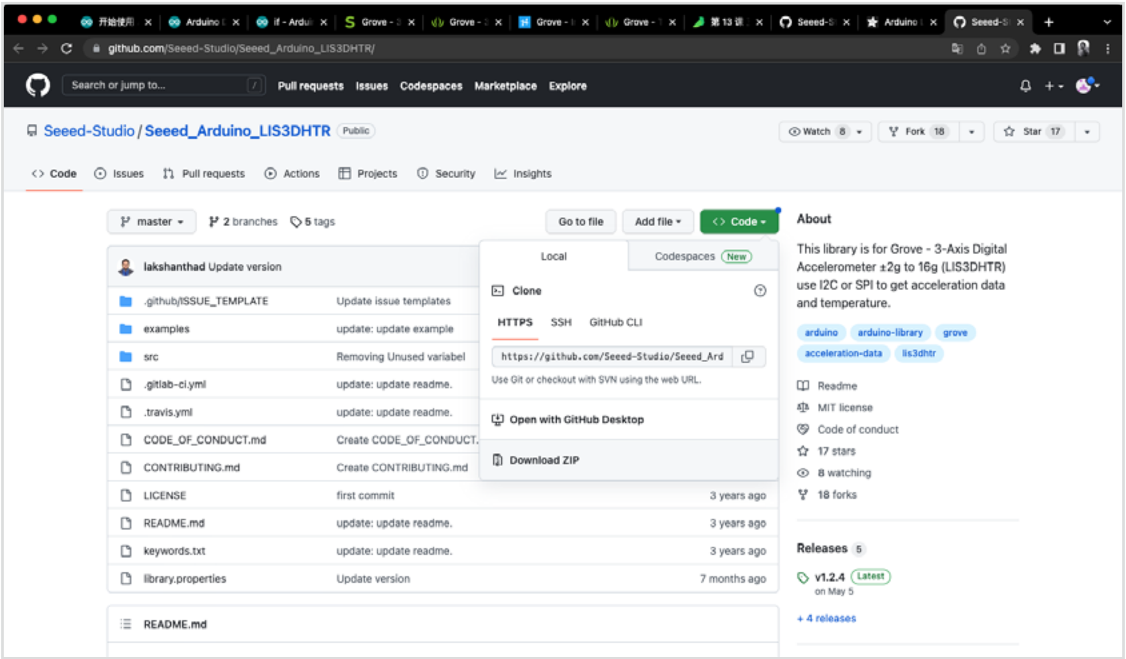
Grove - Infrared Emitter Infrared Transmitter Module

Grove - Infrared Emitter Infrared Transmitter Module

3.1.2 Task 1: Reading Remote Control Key Codes

Adding the Arduino-IRremote Library File

Before we begin programming the Grove - IR Infrared Receiver with the Arduino IDE, we need to add the necessary library files. Enter the library file address https://github.com/Arduino-IRremote/Arduino-IRremote in your browser address bar, go to the GitHub page, and `click Code→Download ZIP` to download the resource package `Arduino-IRremote-master.zip` to your local machine, as shown in the image below:



lakshanthad Update version

.github/ISSUE_TEMPLATE	Update issue templates	
examples	update: update example	
src	Removing Unused variabel	
.gitlab-ci.yml	update: update readme.	
.travis.yml	update: update readme.	
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.	
CONTRIBUTING.md	Create CONTRIBUTING.md	
LICENSE	first commit	3 years ago
README.md	update: update readme.	3 years ago
keywords.txt	update: update readme.	3 years ago
library.properties	Update version	7 months ago

https://github.com/Seeed-Studio/Seeed_Ard

Use Git or checkout with SVN using the web URL.

This library is for Grove - 3-Axis Digital Accelerometer ±2g to 16g (LIS3DHTR) use I2C or SPI to get acceleration data and temperature.

Add the resource package `Arduino-IRremote-master.zip` you just downloaded through `Sketch→Include Library→Add .ZIP` Library in the Arduino IDE menu bar until you see a message indicating successful library loading.

Open the Example File

If you want to control other devices through the infrared remote control, such as pressing the left key on the mini infrared remote control to rotate the servo to the left, or pressing the right key to rotate the servo to the right, you first need to know what kind of code each key on the remote control will emit. This way, you can set it through the program. But how do you read the codes of different keys on the remote control? You can use the `IRremote` library and open the `IRrecvDemo` example via the following path: `**File→Examples→IRremote→ReceiveDemo**`. This example program can read the key codes of the remote control, but some parameters need to be modified:

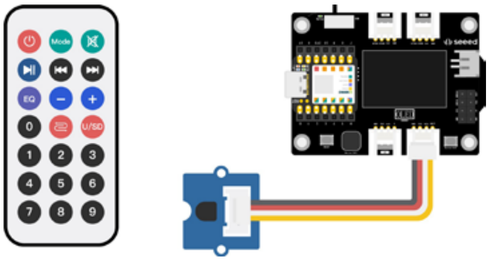
`**int RECV_PIN = 7**`, change the number according to the hardware connection pin. We have connected the infrared receiver to pin 7. Next, we select useful code. We only need to define the header file and the part that reads the remote control key codes. After reducing, the program is as follows:

```
#include <Arduino.h>
#include <IRremote.h>

const byte IR_RECEIVE_PIN=7; // The infrared receiver is connected to pin 7. If you
are using XIAO RP2040/XIAO ESP32, please change 7 to A0

void setup() {
    Serial.begin(115200);
    Serial.println(F(“Enabling IRin”));
```

```
     IrReceiver.begin(IR_RECEIVE_PIN,ENABLE_LED_FEEDBACK); // Start infrared decoding
     Serial.print(F(“Ready to receive IR signals at pin “));
     Serial.println(IR_RECEIVE_PIN);
     delay(1000);
}

void loop() {
     if (IrReceiver.decode()) // Decode successfully, receive a set of infrared sig-
nals
   {
        Serial.println(IrReceiver.decodedIRData.command, HEX); // Output infrared de-
coding result (hexadecimal)
        Serial.println(IrReceiver.decodedIRData.command); // Output infrared decoding
result (octal)
       IrReceiver.resume(); // Receive the next set of values
   }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L11_IRrecvDemo_en

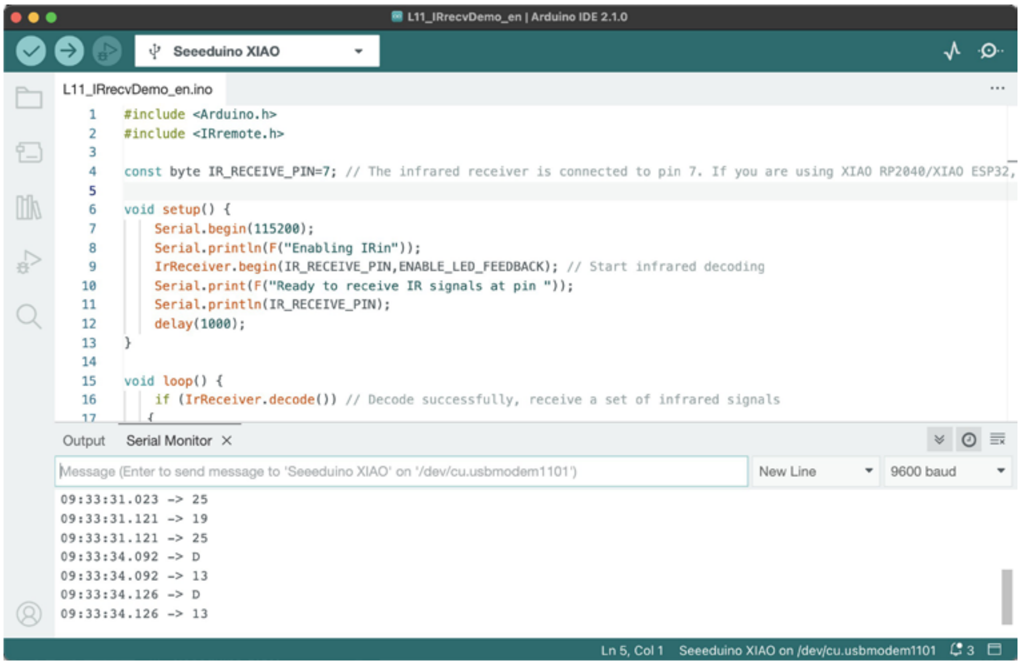
The infrared receiver module is connected to the port 7, as shown in the following figure:



- Attention -

If you are using XIAO RP2040/XIAO ESP32, please change `7` *to* `A0`.

After the code is uploaded, open the serial monitor, aim the remote control at the black component of the infrared receiver at a close distance, press any key, and observe the characters output by the serial monitor. The hexadecimal code appears in the first line, and the octal code appears in the second line. The two lines form one group, representing one key. Please note that if you press the key for too long, “FFFFFFFF” will appear, and this line of code and the numeric code below are invalid.



L11_IRrecvDemo_en | Arduino IDE 2.1.0

Seeeduino XIAO

L11_IRrecvDemo_en.ino

```
#include <Arduino.h>
#include <IRremote.h>

const byte IR_RECEIVE_PIN=7; // The infrared receiver is connected to pin 7. If you are using XIAO RP2040/XIAO ESP32,

void setup() {
    Serial.begin(115200);
    Serial.println(F("Enabling IRin"));
    IrReceiver.begin(IR_RECEIVE_PIN,ENABLE_LED_FEEDBACK); // Start infrared decoding
    Serial.print(F("Ready to receive IR signals at pin "));
    Serial.println(IR_RECEIVE_PIN);
    delay(1000);
}

void loop() {
    if (IrReceiver.decode()) // Decode successfully, receive a set of infrared signals
    {
```

Output Serial Monitor ×

Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem1101') New Line 9600 baud

```
09:33:31.023 -> 25
09:33:31.121 -> 19
09:33:31.121 -> 25
09:33:34.092 -> D
09:33:34.092 -> 13
09:33:34.126 -> D
09:33:34.126 -> 13
```

Ln 5, Col 1 Seeeduino XIAO on /dev/cu.usbmodem1101 3

- Attention -

Different remote controls may give different values.3.1.3 Project Creation: Smart Remote Door

3.1.3 Project Creation: Smart Remote Door

Project Description

How can we recreate a smart remote control door? With a remote control and an infrared receiver, the next step is to control the opening and closing of the door. Recall how the remote control doors in our life work? When the remote control is pressed, the door slowly opens. When it opens to a certain angle, it slowly closes. We can use a servo to control the rotation of the door. When closing the door, the servo rotates from 90° to 0°. When opening the door, the servo rotates from 0° to 90°. By transmitting the signals to open and close the door with a remote control, we can implement the function of a smart remote control door.

Program Writing

To control the rotation of the servo with an infrared remote control, you need to follow these steps:

- Declare the IRremote library and Serve library to be called, and define variables.
- Initialize the library files, initialize the servo.
- Read the infrared decoding result and control the rotation of the servo according to the instructions to the left and right.

Task 2: Control the Rotation of the Servo with an Infrared Remote Control

Step 1: Declare the IRremote library and Serve library to be called, and define variables.

```
#include <IRremote.h>
#include <Servo.h>

Servo myservo; // Create a servo object myservo to control the servo
int RECV_PIN = 7; // The infrared receiver is connected to pin 7. If you are using
XIAO RP2040/XIAO ESP32, please change 7 to A0
IRrecv irrecv(RECV_PIN); // Define an IRrecv object to receive infrared signals
decode_results results; // Decoding results are placed in results

int pos = 90; // Define pos as 90°
```

Step 2: Initialize the library files, initialize the servo.

```
void setup()
{
    Serial.begin(9600);
    Serial.println(“Enabling IRin”);
    irrecv.enableIRIn();
     myservo.attach(5); // Connect the servo on pin 5 to myservo. If you are using
XIAO RP2040/XIAO ESP32, please change 5 to D5
}
```

Step 3: Read the infrared decoding result and control the rotation of the servo according to the instructions to the left and right. If you have questions about the program, you can refer to the comment section.

- Attention -

In the example, the infrared signal value of the right key `16761405`*, and the infrared signal value of the left key* `16712445`*, need to be replaced by the values obtained from the "Read Remote Control Key Code" example using the remote control in your hand. Otherwise, there will be no response after pressing the key.*

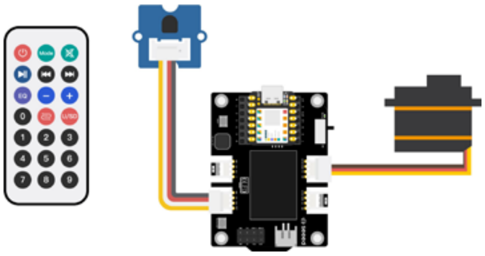
```
void loop() {
    if (irrecv.decode(&results)) {  // If decoding is successful, a set of infrared
signals is received
        if (results.value == 16761405) {  // If the received signal is 16761405 (right
key)
             for (pos; pos <= 89; pos += 1) { // Then the servo is incremented from
0° to 90° in sequence
                 myservo.write(pos);               // Write the rotation angle value
to the servo pin
                delay(40);
                 // The following is to interrupt the above instruction and exit the
loop
                if (irrecv.decode(&results)) {
                    irrecv.resume();
                    if (results.value == 16712445)
                        break;
                }
            }
        }

         if (results.value == 16712445) {     // If the received signal is 16712445
(left key)
             for (pos; pos >= 1; pos -= 1) { // Then the servo is decremented from
90° to 0° in sequence
                 myservo.write(pos);               // Write the rotation angle value
to the servo pin
                delay(40);
                 // The following is to interrupt the above instruction and exit the
loop
                if (irrecv.decode(&results)) {
                    irrecv.resume();
                    if (results.value == 16761405)
                        break;
                }
            }
        }
        // Display hexadecimal and octal codes in the serial port
        Serial.println(pos);
        Serial.println(results.value, HEX);
        Serial.println(results.value);
        irrecv.resume();

    }
    delay(100);
}
```

Complete program details:

```
#include <IRremote.h>
#include <Servo.h>

Servo myservo; // Create a servo object myservo to control the servo
int RECV_PIN = 7; // The infrared receiver is connected to pin 7. If you are using
XIAO RP2040/XIAO ESP32, please change 7 to A0
IRrecv irrecv(RECV_PIN); // Define an IRrecv object to receive infrared signals
decode_results results; // Decoding results are placed in results

int pos = 90; // Define pos as 90°

void setup()
{
    Serial.begin(9600);
    Serial.println(“Enabling IRin”);
    irrecv.enableIRIn();
     myservo.attach(5); // Connect the servo on pin 5 to myservo. If you are using
XIAO RP2040/XIAO ESP32, please change 5 to D5
}

// Note: Left 16712445 Right 16761405, please replace with the key values read from
your own remote control
void loop() {
    if (irrecv.decode(&results)) {  // If decoding is successful, a set of infrared
signals is received
        if (results.value == 16761405) {  // If the received signal is 16761405 (right
key)
             for (pos; pos <= 89; pos += 1) { // Then the servo is incremented from
0° to 90° in sequence
                 myservo.write(pos);               // Write the rotation angle value
to the servo pin
                delay(40);
                 // The following is to interrupt the above instruction and exit the
loop
                if (irrecv.decode(&results)) {
                    irrecv.resume();
                    if (results.value == 16712445)
                        break;
                }
            }
        }

         if (results.value == 16712445) {     // If the received signal is 16712445
(left key)
             for (pos; pos >= 1; pos -= 1) { // Then the servo is decremented from
90° to 0° in sequence
                 myservo.write(pos);               // Write the rotation angle value
to the servo pin
                delay(40);
                 // The following is to interrupt the above instruction and exit the
loop
                if (irrecv.decode(&results)) {
                    irrecv.resume();
                    if (results.value == 16761405)
                        break;
                }
            }
        }
        // Display hexadecimal and octal codes in the serial port
        Serial.println(pos);
        Serial.println(results.value, HEX);
        Serial.println(results.value);
        irrecv.resume();

    }
    delay(100);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L11_IR_Servo_ino_XIAO_en

Step 4: Connect the hardware and upload the program. First, connect the infrared receiving module to the 7th interface of the XIAO expansion board, and connect the servo to the I2C interface, as shown in the figure below:



- Attention -

> *If you are using XIAO RP2040, please connect the infrared receiving module to the* `A0` *interface.*

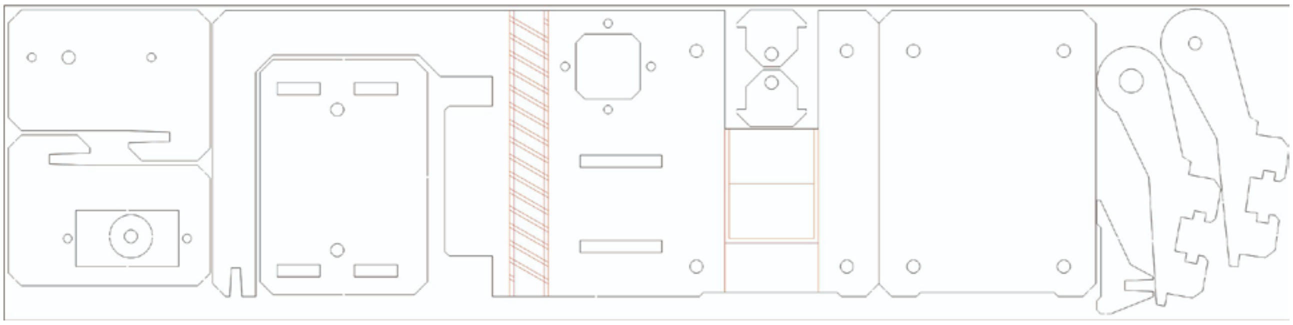
Connect XIAO to the computer with a data cable, click the “Upload” button, upload the program to the hardware, and when the debug area shows “Upload Successful”, open the serial monitor, aim the remote control at the infrared receiver, press the “Left” key and the “Right” key, observe the rotation of the servo, and check the encoding information output by the serial monitor.

3.1.4 Exterior Design

In this unit, we need to implement a more complete project, combining the functions implemented by the program, the modules, and the appearance of the structure to form a prototype. Going back to the smart remote control door project, we need to control the rotation of the servo through the remote control, simulate the opening and closing of the door. When making the appearance, we need to focus on the following issues:

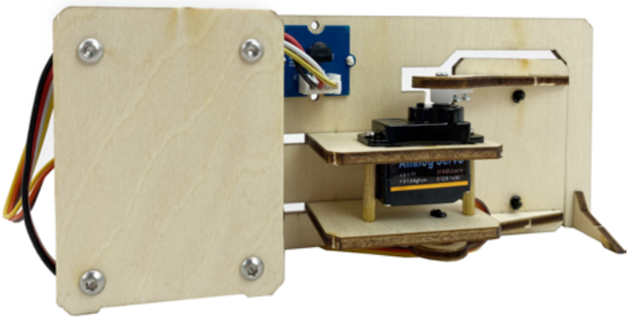
- How to combine the servo and the door panel to make the rotation of the servo drive the rotation of the door panel.
- The infrared receiver should be exposed in a conspicuous position, without any cover.
- Whether the main control, expansion board, and connecting wires are covered to keep the appearance neat.
- How to make the work stand steadily.

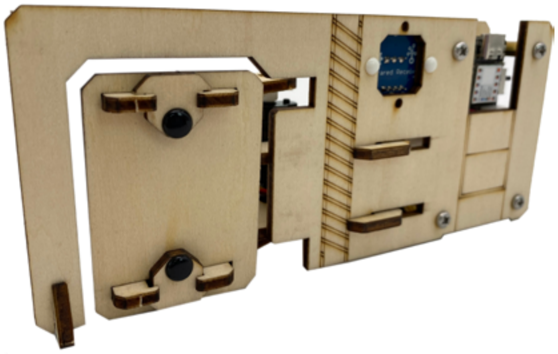
The figure below provides an appearance case, which is laser cut from basswood, and provides cutting files for reference. If you can use drawing software, you can process and design it yourself. If you don’t have a laser cutting machine, you can also use corrugated paper, cardstock, non-woven fabric, and other handmade materials to make it, which tests your hands-on ability more.

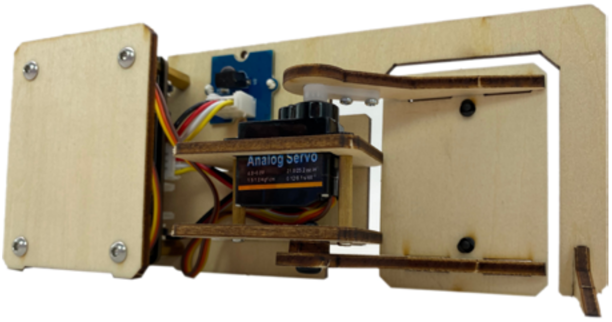


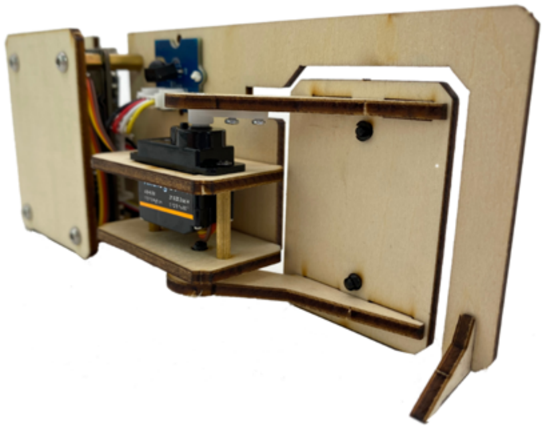
Download files for use with a laser cutter

https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/dxf/XIAO_ADR.dxf.









3.2
Smart Watch

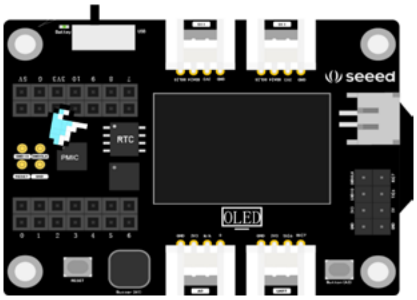
The watch is a common item in life. Even though various electronic devices now have timing functions, and mobile phones can replace watches as timing tools, watches are still a popular item. They are not only timing tools, but also have fashion matching functions. Although watches are delicate and small, they involve complex craftsmanship. Now with XIAO and its expansion board, we can easily make them.

3.2.1 Background Knowledge

RTC Clock

RTC stands for Real_Time Clock, which is an integrated circuit used to display time, also called an RTC clock chip. RTCs are widely used, and we can find RTC in almost any electronic device. In the XIAO expansion board, there is an RTC clock chip, as shown in the following figure. We can display the date and time on the OLED display on the expansion board, and it can be powered by a button battery or lithium battery. Even if we disconnect, it can continue to track time. When we reconnect the power supply, we will find that the time is still moving. With the RTC clock, we can make timed reminder devices, such as timed watering, timed pet feeding, and so on.

Grove also has an RTC module: Grove - DS1307 RTC (Real Time Clock) for Arduino, as shown in the figure below.



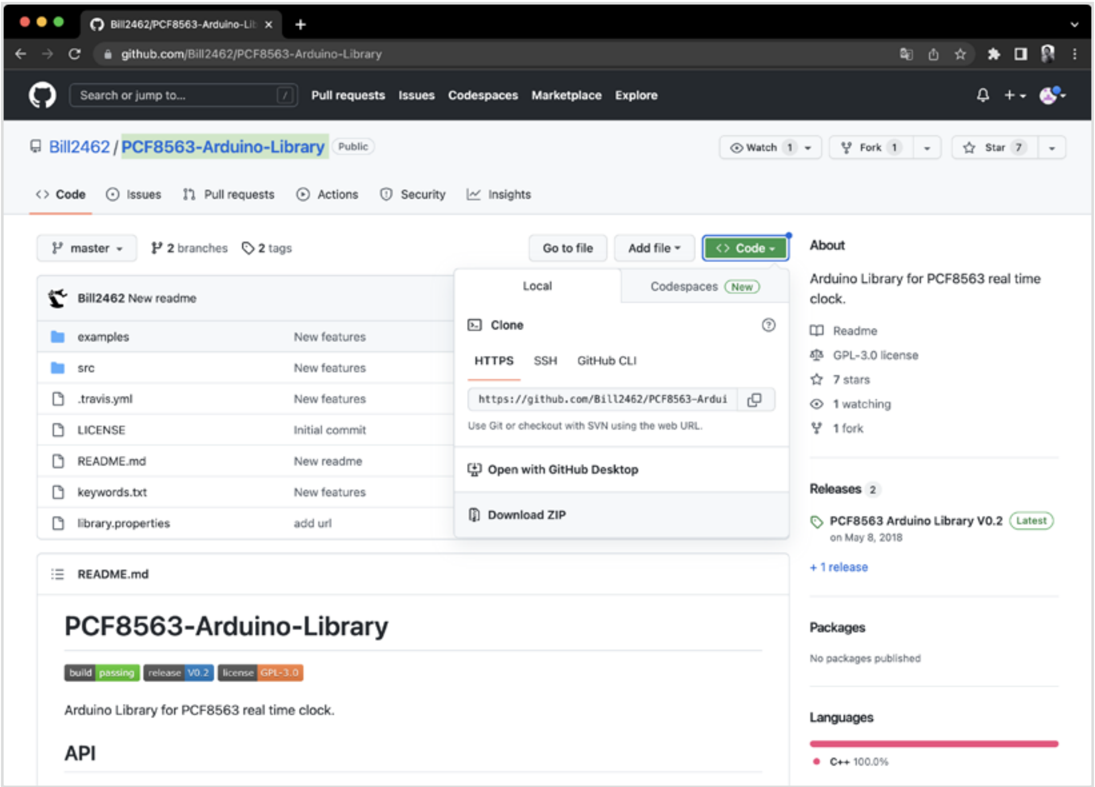
Position of the RTC clock chip on the XIAO expansion board

Grove RTC Module

3.2.2 Task 1: Displaying RTC Clock in the Serial Monitor

Adding PCF8563-Arduino-Library Library

Before starting to program the RTC on the XIAO expansion board with Arduino IDE, you need to add the necessary library files. Enter the library file address https://github.com/Bill2462/PCF8563-Arduino-Library in the browser address bar, go to the GitHub page, click `Code→Download ZIP` to download the resource package `PCF8563-Arduino-Library-master.zip` to the local, as shown in the figure below.



Bill2462 New readme	
examples	New features
src	New features
.travis.yml	New features
LICENSE	Initial commit
README.md	New readme
keywords.txt	New features
library.properties	add url

https://github.com/Bill2462/PCF8563-Ardui

Use Git or checkout with SVN using the web URL.

Arduino Library for PCF8563 real time clock.

README.md

PCF8563-Arduino-Library

Arduino Library for PCF8563 real time clock.

API

Add the previously downloaded resource package `PCF8563-Arduino-Library-master.zip` in `Sketch→Include Library→Add .ZIP Library` in the menu bar until you see the library loading success prompt.

Opening the Sample File

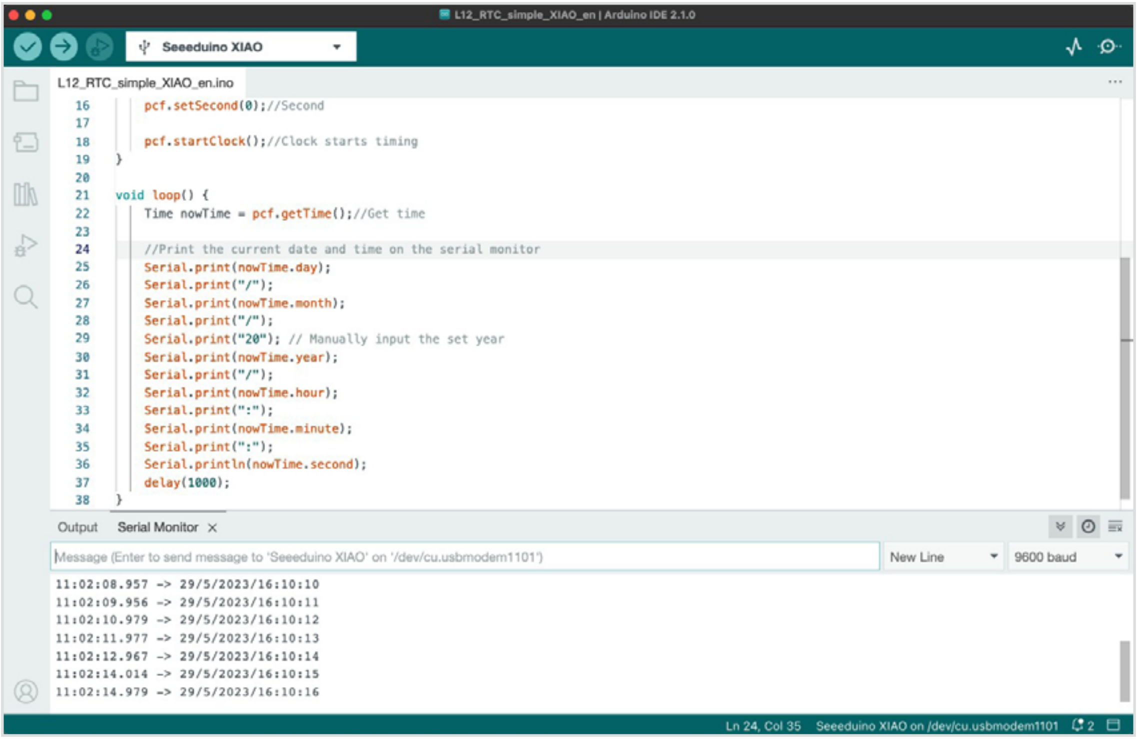
Creating an RTC clock can't be without the powerful library file. Open the `simple` example through the following path: `File→Examples→PCF8563→simple`. This example program can display the RCT clock through the serial monitor. After opening the example program, we only need to modify the current date and start time:

```
#include <PCF8563.h> //Declare library file
PCF8563 pcf;//Define variable pcf

void setup() {
    Serial.begin(9600);
    pcf.init();//Initialize the clock
    pcf.stopClock();//Stop the clock

     //Set the current date and time. After setting, it will start timing from this
moment

    pcf.setYear(23);//Year
    pcf.setMonth(05);//Month
    pcf.setDay(29);//Day
    pcf.setHour(16);//Hour
    pcf.setMinut(10);//Minute
    pcf.setSecond(0);//Second

    pcf.startClock();//Clock starts timing
}
```

```
void loop() {
    Time nowTime = pcf.getTime();//Get time

    //Print the current date and time on the serial monitor
    Serial.print(nowTime.day);
    Serial.print(“/”);
    Serial.print(nowTime.month);
    Serial.print(“/”);
    Serial.print(“20”); // Manually input the set year
    Serial.print(nowTime.year);
    Serial.print(“/”);
    Serial.print(nowTime.hour);
    Serial.print(“:”);
    Serial.print(nowTime.minute);
    Serial.print(“:”);
    Serial.println(nowTime.second);
    delay(1000);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L12_RTC_simple_XIAO_en

Without connecting other electronic modules, you can click the upload program button. After the code is uploaded, open the serial monitor, and you will be able to see the time.



L12_RTC_simple_XIAO_en | Arduino IDE 2.1.0
Seeeduino XIAO
L12_RTC_simple_XIAO_en.ino

```
    pcf.setSecond(0);//Second

    pcf.startClock();//Clock starts timing
}

void loop() {
    Time nowTime = pcf.getTime();//Get time

    //Print the current date and time on the serial monitor
    Serial.print(nowTime.day);
    Serial.print("/");
    Serial.print(nowTime.month);
    Serial.print("/");
    Serial.print("20"); // Manually input the set year
    Serial.print(nowTime.year);
    Serial.print("/");
    Serial.print(nowTime.hour);
    Serial.print(":");
    Serial.print(nowTime.minute);
    Serial.print(":");
    Serial.println(nowTime.second);
    delay(1000);
}
```

Output
Serial Monitor ×
Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem1101')
New Line
9600 baud

```
11:02:08.957 -> 29/5/2023/16:10:10
11:02:09.956 -> 29/5/2023/16:10:11
11:02:10.979 -> 29/5/2023/16:10:12
11:02:11.977 -> 29/5/2023/16:10:13
11:02:12.967 -> 29/5/2023/16:10:14
11:02:14.014 -> 29/5/2023/16:10:15
11:02:14.979 -> 29/5/2023/16:10:16
```

3.2.3 Project Making: Smart Watch

Project Description

In this section, we are going to make a smart watch that can display the date, time, temperature, and humidity in real time. To display the date and time, we just need XIAO and the expansion board. To display the temperature and humidity, we need to add a temperature and humidity sensor.

Programming

The program consists of the following steps:

- Declare the necessary libraries and define variables.
- Initialize the libraries, and set the current time.
- Read temperature and humidity variables, get the current time, and display the temperature, humidity, and date/time on the OLED screen.

- Attention -

Before starting to program for the OLED of the XIAO expansion board, make sure that the `U8g2_Arduino` *library has been loaded into the Arduino IDE. The loading method can be referred to the instructions in the "How to Download and Install Arduino Libraries" section of Section 1.1.*

Before starting to program for the Grove temperature and humidity sensor, make sure that the Arduino IDE has loaded the `Grove_Temperature_And_Humidity_Sensor` *library. The loading method can be referred to the instructions in the "Adding the Grove_Temperature_And_Humidity_Sensor Library" section of Section 2.2.*

Task 2: Display the current time and temperature/humidity values on the OLED display of the XIAO expansion board (based on the DHT20 sensor)

Step 1: Declare the necessary libraries and define variables.

```
#include <Arduino.h>
#include <U8x8lib.h> //use u8x8 library
#include <PCF8563.h> //RTC library
PCF8563 pcf; //define variable pcf
#include <Wire.h>
#include “DHT.h” //DHT library
#define DHTTYPE DHT20 //The type of the temperature and humidity sensor is DHT20
DHT dht(DHTTYPE);
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE); //OLED’s con-
structor, set data type, connect OLED display
```

Step 2: Initialize the libraries, and set the current time.

```
void setup() {
    Serial.begin(9600);
    u8x8.begin(); //u8x8 starts working
    u8x8.setFlipMode(1);
    Wire.begin();
    pcf.init(); //Initialize the clock
    pcf.stopClock(); //Stop the clock
    //Set the current time and date:
    pcf.setYear(23);
    pcf.setMonth(05);
    pcf.setDay(29);
    pcf.setHour(18);
    pcf.setMinut(53);
    pcf.setSecond(0);
    pcf.startClock(); //The clock starts timing
}
```

Step 3: Read temperature and humidity variables, get the current time, and display the temperature, humidity, and date/time on the OLED screen.

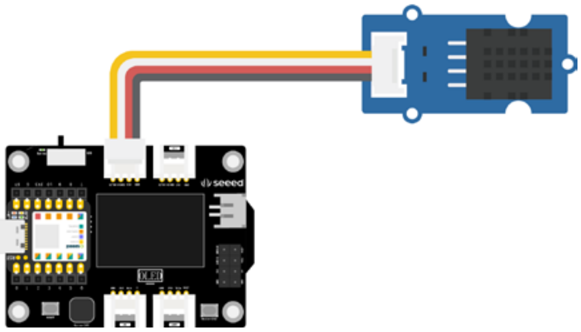
```
void loop() {
    float temp, humi; //Define temperature and humidity variables
    temp = dht.readTemperature(); //Read the temperature value
    humi = dht.readHumidity(); //Read the humidity value
    Time nowTime = pcf.getTime(); //Get the time
    u8x8.setFont(u8x8_font_chroma48medium8_r); //u8x8 font

    //Display the current date, time, temperature, and humidity at different coordi-
nates on the OLED screen.
    u8x8.setCursor(0, 0);
    u8x8.print(nowTime.day);
    u8x8.print(“/”);
    u8x8.print(nowTime.month);
    u8x8.print(“/”);
    u8x8.print(“20”);
    u8x8.print(nowTime.year);
    u8x8.setCursor(0, 1);
    u8x8.print(nowTime.hour);
    u8x8.print(“:”);
    u8x8.print(nowTime.minute);
    u8x8.print(“:”);
    u8x8.println(nowTime.second);
    delay(1000);
    u8x8.setCursor(0, 2);
    u8x8.print(“Temp:”);
    u8x8.print(temp);
    u8x8.print(“C”);
    u8x8.setCursor(0,3);
    u8x8.print(“Humidity:”);
    u8x8.print(humi);
    u8x8.print(“%”);
    u8x8.refreshDisplay();
    delay(200);
}
```

For the complete program, please refer to:

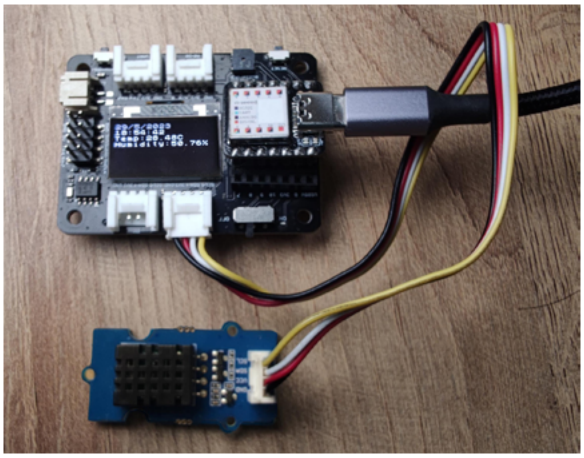
```
#include <Arduino.h>
#include <U8x8lib.h> //use u8x8 library
#include <PCF8563.h> //RTC library
PCF8563 pcf; //define variable pcf
#include <Wire.h>
#include “DHT.h” //DHT library
#define DHTTYPE DHT20 //The type of the temperature and humidity sensor is DHT20
DHT dht(DHTTYPE);
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE); //OLED’s con-
structor, set data type, connect OLED display

void setup() {
    Serial.begin(9600);
    u8x8.begin(); //u8x8 starts working
    u8x8.setFlipMode(1);
    Wire.begin();
    pcf.init(); //Initialize the clock
    pcf.stopClock(); //Stop the clock
    //Set the current time and date:
    pcf.setYear(23);
    pcf.setMonth(05);
    pcf.setDay(29);
    pcf.setHour(18);
    pcf.setMinut(53);
    pcf.setSecond(0);
    pcf.startClock(); //The clock starts timing
}
void loop() {
    float temp, humi; //Define temperature and humidity variables
    temp = dht.readTemperature(); //Read the temperature value
    humi = dht.readHumidity(); //Read the humidity value
    Time nowTime = pcf.getTime(); //Get the time
    u8x8.setFont(u8x8_font_chroma48medium8_r); //u8x8 font

    //Display the current date, time, temperature, and humidity at different coordi-
nates on the OLED screen.
    u8x8.setCursor(0, 0);
    u8x8.print(nowTime.day);
    u8x8.print("/");
    u8x8.print(nowTime.month);
    u8x8.print("/");
    u8x8.print("20");
    u8x8.print(nowTime.year);
    u8x8.setCursor(0, 1);
    u8x8.print(nowTime.hour);
    u8x8.print(":");
    u8x8.print(nowTime.minute);
    u8x8.print(":");
    u8x8.println(nowTime.second);
    delay(1000);
    u8x8.setCursor(0, 2);
    u8x8.print("Temp:");
    u8x8.print(temp);
    u8x8.print("C");
    u8x8.setCursor(0,3);
    u8x8.print("Humidity:");
    u8x8.print(humi);
    u8x8.print("%");
    u8x8.refreshDisplay();
    delay(200);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L12_SmartWatch_DHT20_XIAO_en

Step 4: Connect the hardware and upload the program. First, connect the DHT20 temperature and humidity sensor to the I2C interface of the XIAO expansion board, and connect the XIAO to the computer with a data cable, as shown in the picture:



Click the "Upload" button in the Arduino IDE to upload the program to the hardware. When the debugging area shows "Upload successful", observe whether the OLED display correctly outputs the current time and starts timing, as well as the real-time temperature and humidity.



Task 3: Display the current time and temperature and humidity values on the OLED display of the XIAO expansion board (based on the DHT11 sensor)

If you are using the Grove DHT11 temperature and humidity sensor with a blue casing, part of the program code needs to be modified as follows:

`#define DHTPIN 0`, this needs to be modified according to the actual pin number to which the temperature and humidity sensor is connected.

`#define DHTTYPE DHT11`, since there are different models of temperature and humidity sensors, you need to select the correct model — DHT11. The modified example code is as follows:

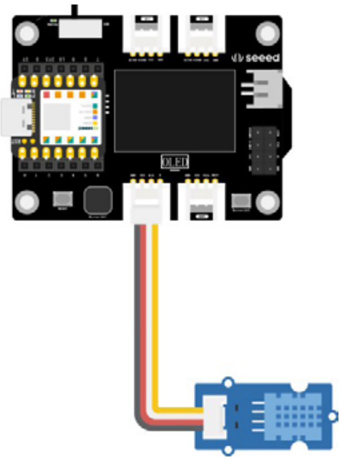
```
#include <Arduino.h>
#include <U8x8lib.h>
#include <PCF8563.h>
PCF8563 pcf;
#include <Wire.h>
#include "DHT.h"
#define DHTPIN 0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
//U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(/* clock=*/ SCL, /* data=*/ SDA, /* re-
set=*/ U8X8_PIN_NONE);   // OLEDs without Reset of the Display

void setup() {
  Serial.begin(115200);
  u8x8.begin();
  u8x8.setFlipMode(1);
  Wire.begin();
  pcf.init(); //initialize the clock
  pcf.stopClock(); //stop the clock
  pcf.setYear(23); //set year
  pcf.setMonth(05); //set month
  pcf.setDay(29); //set date
  pcf.setHour(18); //set hour
  pcf.setMinut(53); //set minute
  pcf.setSecond(0); //set second
  pcf.startClock(); //start the clock
}
```

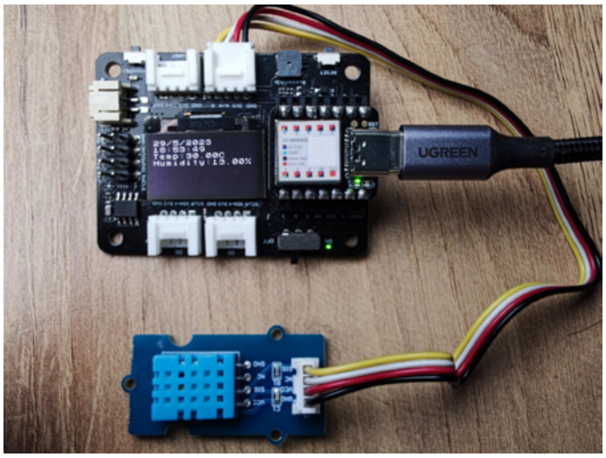
```
void loop() {
  float temp, humi;
  temp = dht.readTemperature();
  humi = dht.readHumidity();
  Time nowTime = pcf.getTime(); //get current time
  u8x8.setFont(u8x8_font_chroma48medium8_r);   // choose a suitable font

  u8x8.setCursor(0, 0);
  u8x8.print(nowTime.day);
  u8x8.print(“/”);
  u8x8.print(nowTime.month);
  u8x8.print(“/”);
  u8x8.print(“20”);
  u8x8.print(nowTime.year);
  u8x8.setCursor(0, 1);
  u8x8.print(nowTime.hour);
  u8x8.print(“:”);
  u8x8.print(nowTime.minute);
  u8x8.print(“:”);
  u8x8.println(nowTime.second);
  delay(1000);
  u8x8.setCursor(0, 2);
  u8x8.print(“Temp:”);
  u8x8.print(temp);
  u8x8.print(“C”);
  u8x8.setCursor(0,3);
  u8x8.print(“Humidity:”);
  u8x8.print(humi);
  u8x8.print(“%”);
  u8x8.refreshDisplay();
  delay(200);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L12_SmartWatch_DHT11_XIAO_en

After modifying the code, first connect the DHT11 temperature and humidity sensor to the `A0` interface of the XIAO expansion board, as shown in the image below.

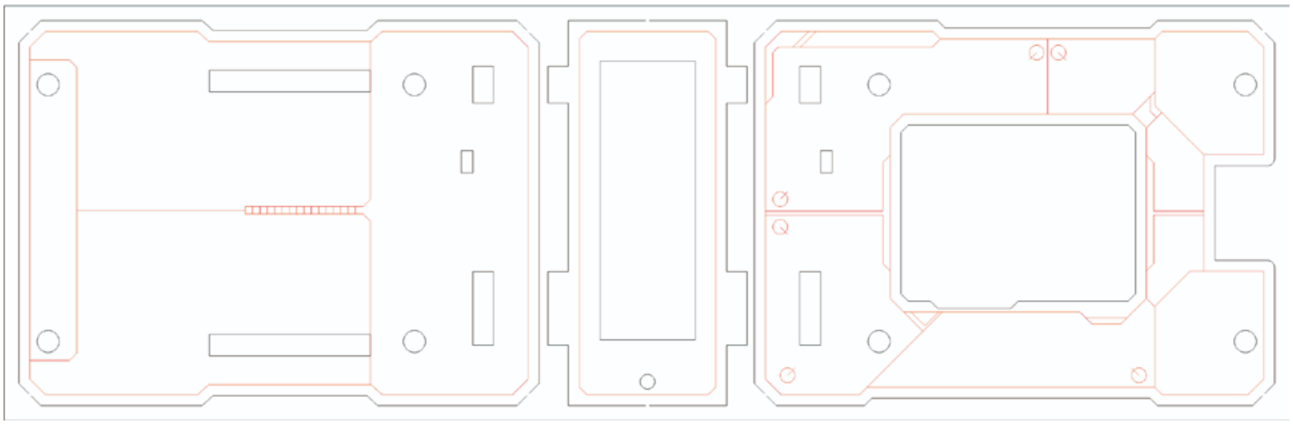
Then connect the XIAO development board to your computer, upload the modified sample program to the XIAO via Arduino IDE, and you should be able to see the time, temperature, and humidity readings on the OLED of the XIAO expansion board. You can place the temperature and humidity sensor in different environments to observe changes in temperature and humidity readings.



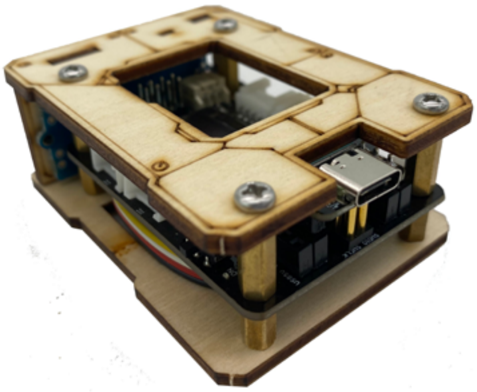


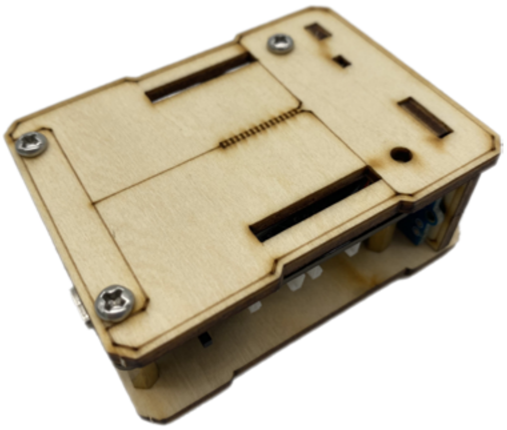
3.2.4 Exterior Design

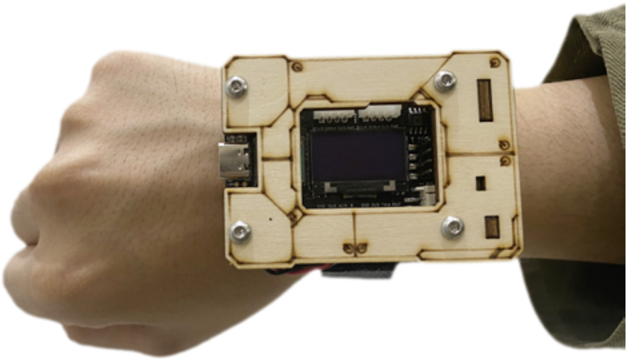
Given its compact size, XIAO is especially suitable for creating wearable devices. The expansion board incorporates an RTC chip, a buzzer, and an OLED display screen, which means you can create a variety of applications even without adding other modules. In this section, we have made a smart watch using the on-board OLED display, RTC chip, and an external temperature and humidity sensor. When creating the appearance, we only need to consider wearability, organization of modules and connecting wires, and the exposure of the OLED display screen. As shown below, we provide a wearable watch style and the laser cutting files for it. With just a simple installation, your wearable device is ready.

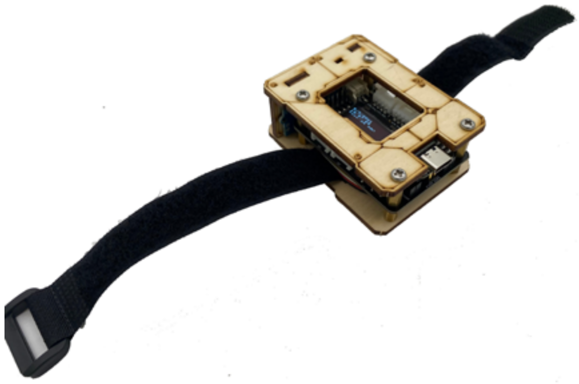


Download files for laser cutting machine https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/dxf/XIAO_X_watch.dxf









3.3
Air Piano

Normally, when we play a musical instrument, we have to pluck strings or press keys to produce musical notes. However, with electronic modules, playing music can become more exciting. For instance, you can simulate piano playing with push-button switches and even integrate light effects for interactive music. But if you use push-button switches as piano keys, you need to integrate many modules into the circuit. Is there a simpler and more unique idea? A combination of ultrasonic distance sensor and passive buzzer can do just that - detecting different distances with ultrasonics to trigger different notes, just like playing a piano in the air.

3.3.1 Background Knowledge

Grove Ultrasonic Distance Sensor

The Grove Ultrasonic Distance Sensor is a non-contact distance measurement module. Thanks to its strong directivity, the ultrasonic waves it emits can travel long distances in a medium. The calculations are simple and it is easy to control, so it's often used for distance measurements. When the ultrasonic distance sensor works, the transmitter emits ultrasonic waves in a certain direction. When the waves hit an obstacle, they reflect back. The ultrasonic receiver stops timing as soon as it receives the reflected waves. The actual distance from the emission point to the obstacle is calculated based on the time difference between emission and reception, much like bat echolocation. The application range of ultrasonic waves is becoming broader, commonly seen in reverse radar systems, intelligent guidance systems, robot obstacle avoidance systems, medical ultrasound examinations, and more.

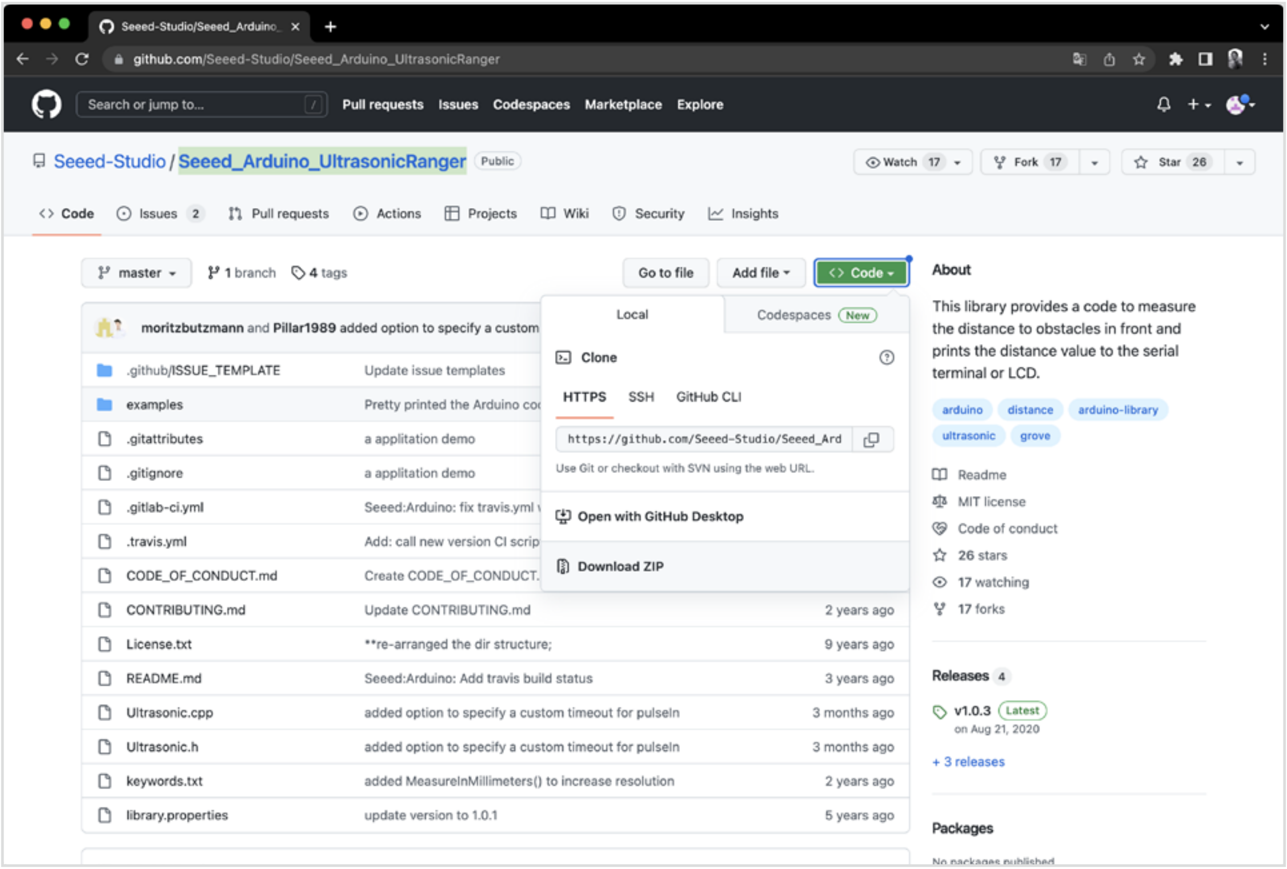
- Attention -

The Grove Ultrasonic Distance Sensor module is not included in the Seeed Studio XIAO Starter Kit!

3.3.2 Task 1: Reading the Grove Ultrasonic Distance Sensor Value

Adding the Seeed_Arduino_UltrasonicRanger Library

Before starting to program the Grove Ultrasonic Distance Sensor with Arduino IDE, it's necessary to add the essential library for the sensor. Type the library address https://github.com/Seeed-Studio/Seeed_Arduino_UltrasonicRanger into the browser address bar, enter the GitHub page, click `Code→Download ZIP` to download the resource package `Seeed_Arduino_UltrasonicRanger-master.zip` to your local drive, as shown below.



moritzbutzmann and Pillar1989 added option to specify a custom

.github/ISSUE_TEMPLATE	Update issue templates	
examples	Pretty printed the Arduino co	
.gitattributes	a applitation demo	
.gitignore	a applitation demo	
.gitlab-ci.yml	Seeed:Arduino: fix travis.yml	
.travis.yml	Add: call new version CI scrip	
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.	
CONTRIBUTING.md	Update CONTRIBUTING.md	2 years ago
License.txt	**re-arranged the dir structure;	9 years ago
README.md	Seeed:Arduino: Add travis build status	3 years ago
Ultrasonic.cpp	added option to specify a custom timeout for pulseIn	3 months ago
Ultrasonic.h	added option to specify a custom timeout for pulseIn	3 months ago
keywords.txt	added MeasureInMillimeters() to increase resolution	2 years ago
library.properties	update version to 1.0.1	5 years ago

```
https://github.com/Seeed-Studio/Seeed_Ard
```

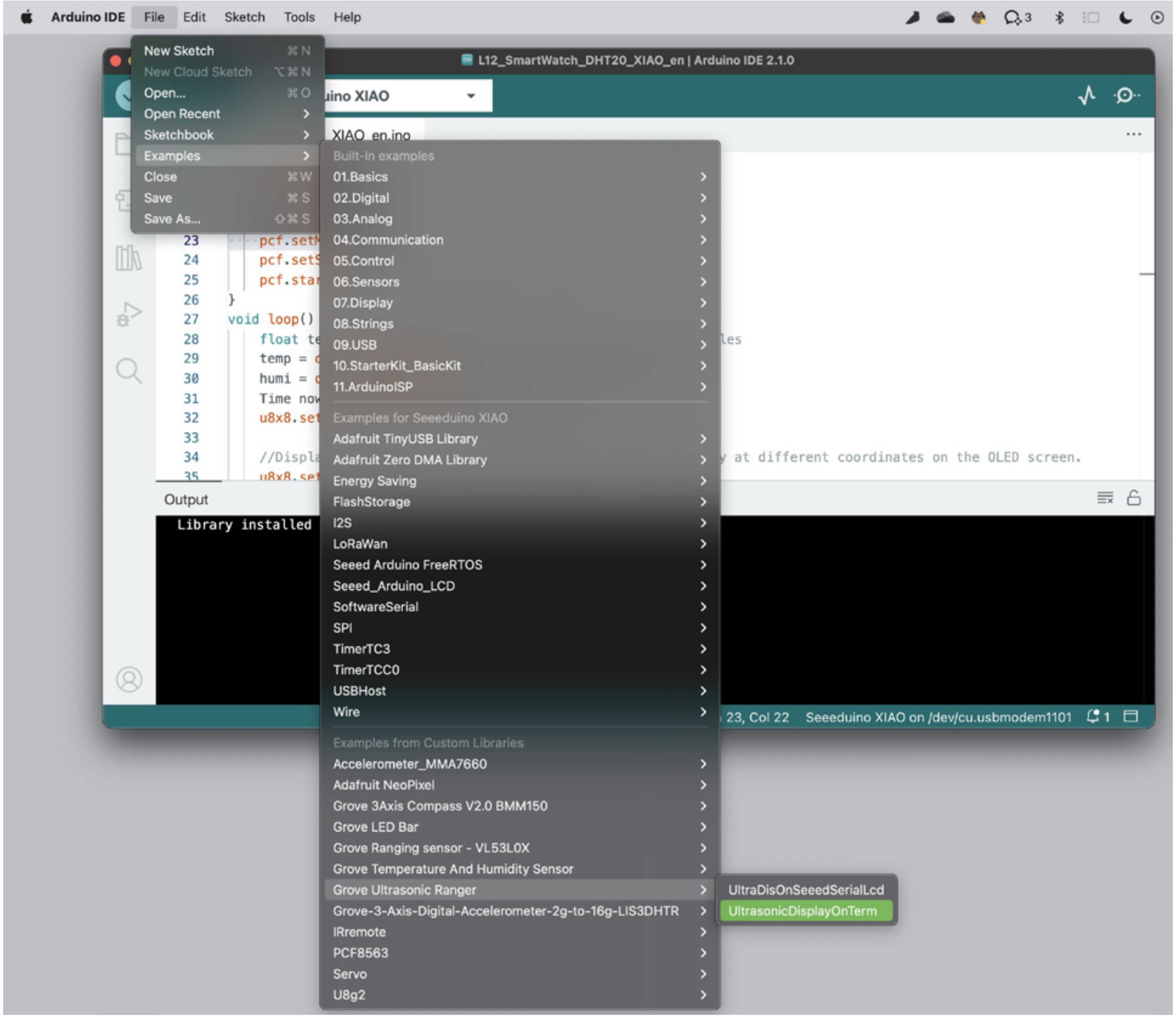
Use Git or checkout with SVN using the web URL.

This library provides a code to measure the distance to obstacles in front and prints the distance value to the serial terminal or LCD.

Add the downloaded resource package `Seeed_Arduino_UltrasonicRanger-master.zip` to the `Sketch→Include Library→Add .ZIP` Library from the menu bar until you see a successful library loading prompt.

Opening the Example File

After successfully installing the library, a new item `Grove Ultrasonic Ranger` will be added to the Arduino's `File→Examples` list. Open the `UltrasonicDisplayOnTerm` sample program from it. This program can display the value of the ultrasonic distance sensor on the Serial Monitor. Modify `Ultrasonic ultrasonic(7)`; in the sample program to `Ultrasonic ultrasonic(0)`; (the ultrasonic distance sensor will be connected to the `A0` port of the XIAO expansion board).

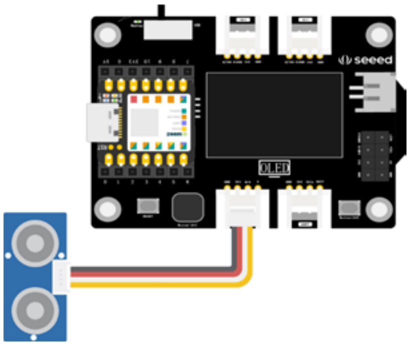


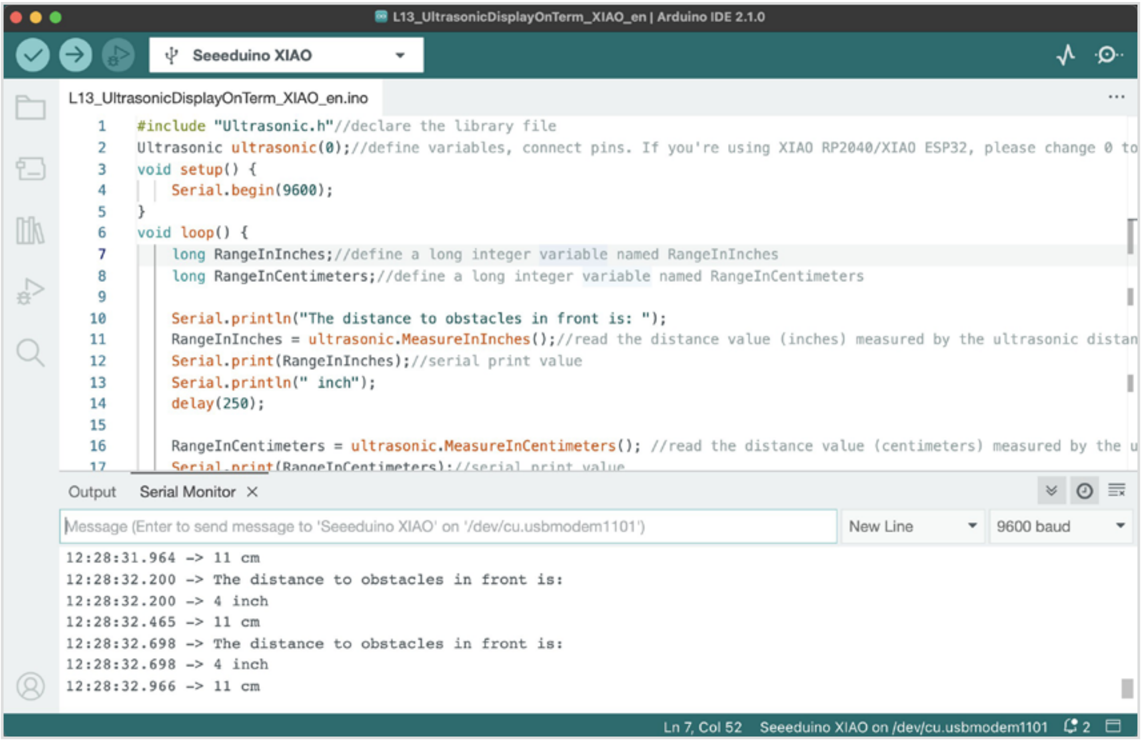
Open the modified sample file through the following path, https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L13_UltrasonicDisplayOnTerm_XIAO_en.

```
#include “Ultrasonic.h”//declare the library file
Ultrasonic ultrasonic(0);//define variables, connect pins. If you’re using XIAO
RP2040/XIAO ESP32, please change 0 to D0
void setup() {
    Serial.begin(9600);
}
void loop() {
    long RangeInInches;//define a long integer variable named RangeInInches
    long RangeInCentimeters;//define a long integer variable named RangeInCentimeters

    Serial.println(“The distance to obstacles in front is: “);
     RangeInInches = ultrasonic.MeasureInInches();//read the distance value (inches)
measured by the ultrasonic distance sensor and store it in the variable RangeInInch-
es
    Serial.print(RangeInInches);//serial print value
    Serial.println(“ inch”);
    delay(250);

     RangeInCentimeters = ultrasonic.MeasureInCentimeters(); //read the distance
value (centimeters) measured by the ultrasonic distance sensor and store it in the
variable RangeInCentimeters
    Serial.print(RangeInCentimeters);//serial print value
    Serial.println(“ cm”);
    delay(250);
}
```

The ultrasonic distance sensor is connected to the `A0` interface, as shown in the figure below:

After uploading the code, open the Serial Monitor. Place your hand or a card at any position in front of the ultrasonic distance sensor and observe the change in the values output by the Serial Monitor.




L13_UltrasonicDisplayOnTerm_XIAO_en | Arduino IDE 2.1.0
Seeeduino XIAO
L13_UltrasonicDisplayOnTerm_XIAO_en.ino
Output
Serial Monitor
New Line
9600 baud
Ln 7, Col 52
Seeeduino XIAO on /dev/cu.usbmodem1101
2

```
#include "Ultrasonic.h"//declare the library file
Ultrasonic ultrasonic(0);//define variables, connect pins. If you're using XIAO RP2040/XIAO ESP32, please change 0 to
void setup() {
  Serial.begin(9600);
}
void loop() {
  long RangeInInches;//define a long integer variable named RangeInInches
  long RangeInCentimeters;//define a long integer variable named RangeInCentimeters

  Serial.println("The distance to obstacles in front is: ");
  RangeInInches = ultrasonic.MeasureInInches();//read the distance value (inches) measured by the ultrasonic distan
  Serial.print(RangeInInches);//serial print value
  Serial.println(" inch");
  delay(250);

  RangeInCentimeters = ultrasonic.MeasureInCentimeters(); //read the distance value (centimeters) measured by the u
  Serial.print(RangeInCentimeters);//serial print value
```

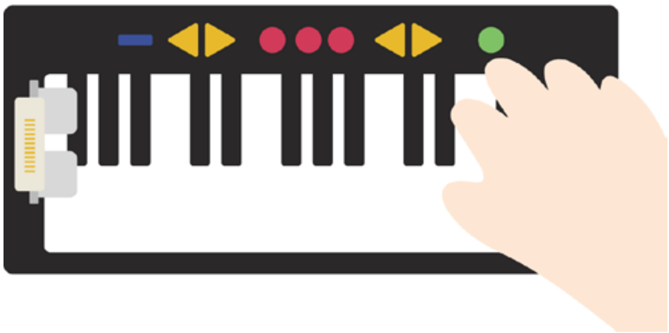
Message (Enter to send message to 'Seeeduino XIAO' on '/dev/cu.usbmodem1101')

```
12:28:31.964 -> 11 cm
12:28:32.200 -> The distance to obstacles in front is:
12:28:32.200 -> 4 inch
12:28:32.465 -> 11 cm
12:28:32.698 -> The distance to obstacles in front is:
12:28:32.698 -> 4 inch
12:28:32.966 -> 11 cm
```

3.3.3 Project Production: Ultrasonic Air Harp

Project Description

The working principle of the air harp is to measure the distance from the module to the palm of your hand through the ultrasonic distance sensor. Depending on the distance, the buzzer emits different musical notes. We have already learned how to measure distance and read values through the ultrasonic distance sensor with the sample program. Next, we just need to define different musical notes for



the corresponding distances. As shown in the figure below: According to the width of the palm, one musical note corresponds to a unit of 2cm, and the performance starts from 4cm. “Do, Re, Mi, Fa, Sol, La, Xi, Do” respectively correspond to 4cm, 6cm, 8cm, 10cm, 12cm, 14cm, 16cm, 18cm... and so on.

Writing the Program

The implementation of the air harp program requires the following steps:

- Declare the library file, define different notes and buzzer pins.
- Initialization, setting the status of the buzzer pin.
- Read the distance (cm) measured by the ultrasonic distance sensor, and make a condition judgment to set different distances to emit different notes.

Using the tone() Function to Play Melody

When we want to control the buzzer to play notes or songs through the program, we need to set the frequency value of each note ourselves. If a song has many notes, it’s too troublesome to adjust one by one, and it tests our music theory knowledge and pitch. Is there a simpler method? Of course! When defining notes, we can refer to the `tone()` function written on the Arduino website https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody, this function defines the corresponding frequency of different notes through pitches.h, which is convenient for us to use the `tone()` function to set the notes emitted by the buzzer. The code of `pitches.h` is shown below:

```
/*
* pitches.h
*/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Task 2: Ultrasonic Air Harp

Step 1: Declare the library file, define different notes and buzzer pins. The main notes we use are “Do Re Mi Fa Sol La Xi Do”, corresponding to “C5 D5 E5 F5 G5 A5 B5 C6”. You can only define the notes you need to avoid the program looking too lengthy.

```
#include “Ultrasonic.h”//declare the library file
Ultrasonic ultrasonic(0);//define the ultrasonic object and connect the ultrasonic
wave to the A0 interface. If you’re using XIAO RP2040, please change 0 to D0
int buzzerPin = 3;//The buzzer is connected to the A3 interface, if you’re using
XIAO RP2040, please change 3 to A3

#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
```

Step 2: Initialize the baud rate and set the buzzer pin status.

```
void setup()
{
    Serial.begin(9600);
    pinMode(buzzerPin,OUTPUT);
}
```

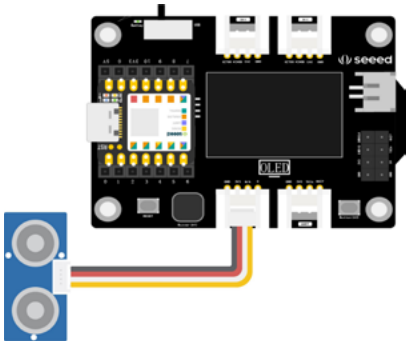
Step 3: Read the distance (cm) measured by the ultrasonic distance sensor and make a condition judgment to set different distances to emit different notes. Since the setting of the air harp is that different distances trigger different notes, and this distance is a long integer value, so we need to use the `long()` function to define the value returned by the ultrasonic wave. For example, `(long)RangeInCentimeters== 4`, that is, the distance value returned by the ultrasonic wave is 4. Corresponding to the buzzer emitting different notes, use the `tone()` function, for example, `tone(3,NOTE_C5,100)`, that is, the buzzer on pin 3, emits `NOTE_C5 (Do)` note, lasts for 100 milliseconds.

```
void loop()
{
    // Read the distance value detected by the ultrasonic distance sensor, in centi-
meters, and print it on the serial monitor
    long RangeInCentimeters;
    RangeInCentimeters = ultrasonic.MeasureInCentimeters();
    Serial.print(RangeInCentimeters);
    Serial.println(“ cm”);
    delay(250);
    // Using an if statement for conditional judgment, when the distance is 4, 6, 8,
10, 12, 14, 16, 18, it corresponds to C5, D5, E5, F5, G5, A5, B5, C6
    if (((long)RangeInCentimeters== 4)) {  //Do
```

```
            tone(3,NOTE_C5,100);
        }
        if (((long) RangeInCentimeters== 6)) { //Re
            tone(3,NOTE_D5,100);
        }
        if (((long) RangeInCentimeters== 8)) { //Mi
            tone(3,NOTE_E5,100);
        }
        if (((long) RangeInCentimeters== 10)) {  //Fa
            tone(3,NOTE_F5,100);
        }
        if (((long) RangeInCentimeters== 12)) {  //Sol
            tone(3,NOTE_G5,100);
        }
        if (((long) RangeInCentimeters== 14)) {  //La
            tone(3,NOTE_A5,100);
        }
        if (((long) RangeInCentimeters== 16)) { //Xi
            tone(3,NOTE_B5,100);
        }
        if (((long) RangeInCentimeters== 18)) {  //Do
            tone(3,NOTE_C6,100);
        }
    }
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L13_UltrasonicPiano_XIAO_en

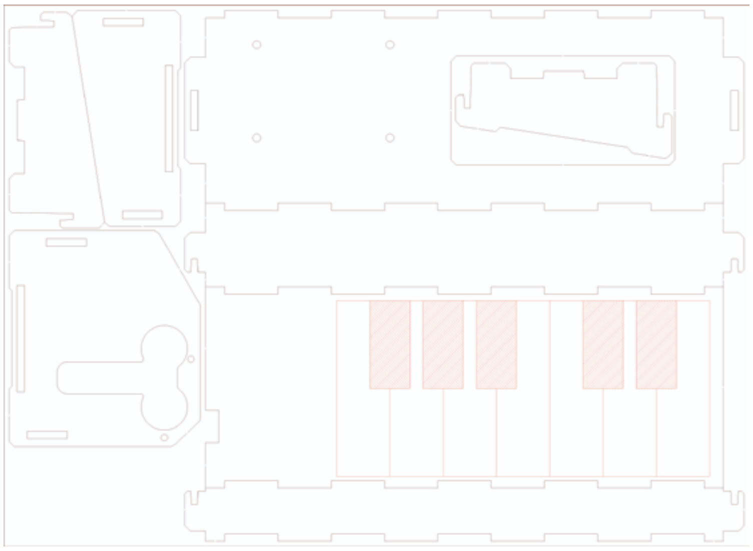
Step 4: Connect the hardware and upload the program. Connect the ultrasonic distance sensor to the `A0` interface of the XIAO expansion board as shown below:



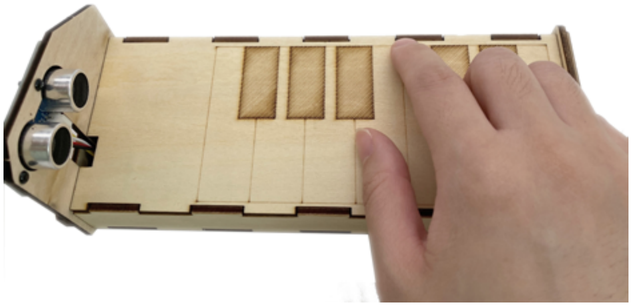
Use the data cable to connect XIAO to the computer, click the "Upload" button, upload the program to the hardware, when the debugging area shows "Upload Successful", open the serial monitor, and start playing with your palm.

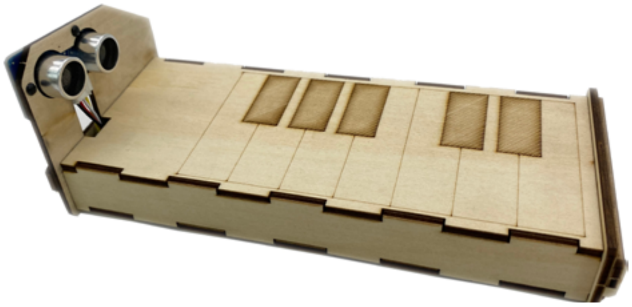
3.3.4 Exterior Design

The inspiration for the air harp comes from the piano, with a note every 2 cm also designed according to the style of the piano keys. In the process of creating the appearance, we can cut a harp surface from a basswood board, and fix the ultrasonic range sensor at the left end of the harp. We also provide laser cutting files for reference, which can be easily assembled, as shown in the picture:



Download the files suitable for the laser cutting machine https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/blob/main/dxf/XIAO_Air_Piano.dxf.





3.4
Implementing Wi-Fi Connection and Applications with XIAO ESP32C3

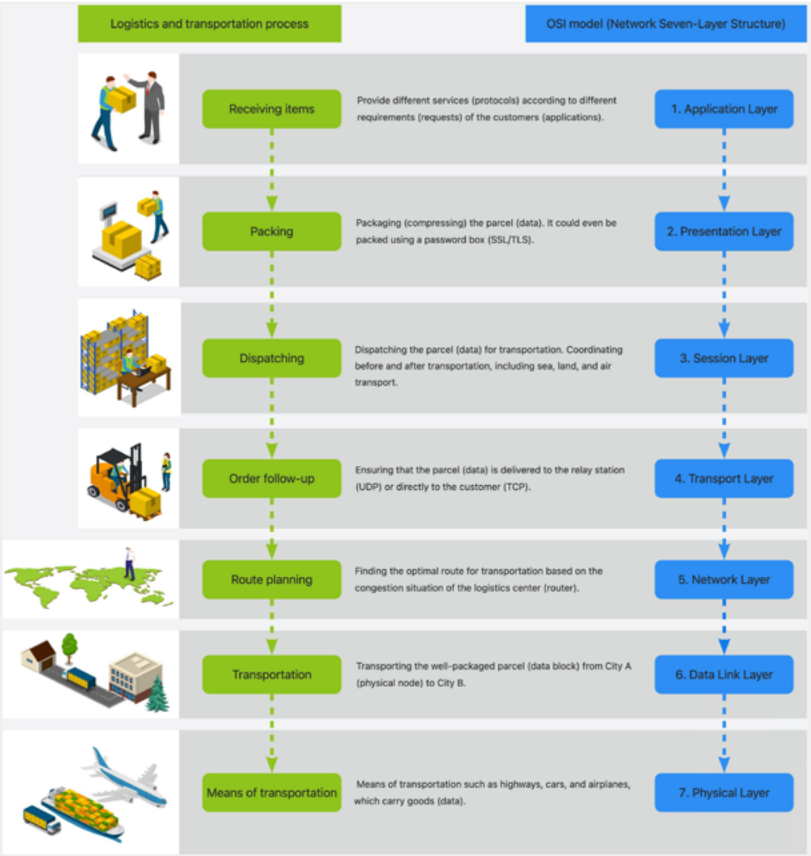
Rather than saying computers have changed the world, it would be more accurate to say that computer networks have. The emergence of networks has truly made computers different from previous tools. The sharing and exchange of information have made computers an epoch-making product. In this section, we will learn how to implement network requests using XIAO ESP32C3, which has Wi-Fi and Bluetooth (BLE) capabilities. This includes connecting XIAO to a Wi-Fi network, pinging specified websites, and issuing GET/POST requests using the HTTP protocol.

Seeed Studio XIAO ESP32C3

3.4.1 Background Knowledge

OSI Reference Model (Network Seven-Layer Model)

OSI (Open System Interconnect) is commonly known as the OSI reference model or network seven-layer structure, which is the network interconnect model researched by the ISO organization in 1985. This architectural standard defines the seven-layer framework (physical layer, data link layer, network layer, transport layer, session layer, presentation layer, and application layer) for network interconnection. For ease of understanding, the following diagram uses a logistics transportation process to correspond to each layer of the OSI model.



Logistics and transportation process		OSI model (Network Seven-Layer Structure)
Receiving items	Provide different services (protocols) according to different requirements (requests) of the customers (applications).	1. Application Layer
Packing	Packaging (compressing) the parcel (data). It could even be packed using a password box (SSL/TLS).	2. Presentation Layer
Dispatching	Dispatching the parcel (data) for transportation. Coordinating before and after transportation, including sea, land, and air transport.	3. Session Layer
Order follow-up	Ensuring that the parcel (data) is delivered to the relay station (UDP) or directly to the customer (TCP).	4. Transport Layer
Route planning	Finding the optimal route for transportation based on the congestion situation of the logistics center (router).	5. Network Layer
Transportation	Transporting the well-packaged parcel (data block) from City A (physical node) to City B.	6. Data Link Layer
Means of transportation	Means of transportation such as highways, cars, and airplanes, which carry goods (data).	7. Physical Layer

The following knowledge will use the concepts of these layers.

ICMP (Internet Control Message Protocol) and ping Command

ICMP (Internet Control Message Protocol) is a sub-protocol of the TCP/IP protocol suite. It is used to transmit control messages between IP hosts and routers. Control messages refer to messages about the network itself, such as whether the network is available, whether the host is reachable, whether the route is available, etc. Although these control messages do not transmit user data, they play an important role in the transmission of user data.

- Learn more -

> *Visit the Wikipedia entry on ICMP.*

We often use the ICMP protocol in the network, such as the `Ping` command (available in both Linux and Windows) we often use to check whether the network is available. This `Ping` process is actually the working process of the ICMP protocol. `ping` can test the connection speed between two devices and accurately report the time it takes for a packet to reach its destination and return to the sender's device. Although `ping` does not provide data about routing or hops, it is still a useful metric for measuring latency between two devices. Below we will learn how to implement `ping` requests on XIAO ESP32C3.

Before starting this attempt, we need to learn how to connect XIAO ESP32C3 with your Wi-Fi.

3.4.2 Task 1: Using Wi-Fi Network on XIAO ESP32C3

XIAO ESP32C3 supports Wi-Fi connections with IEEE 802.11b/g/n. The following will introduce the basic knowledge of using Wi-Fi on this board.

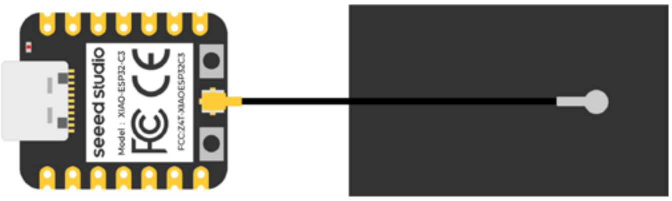
- Attention -

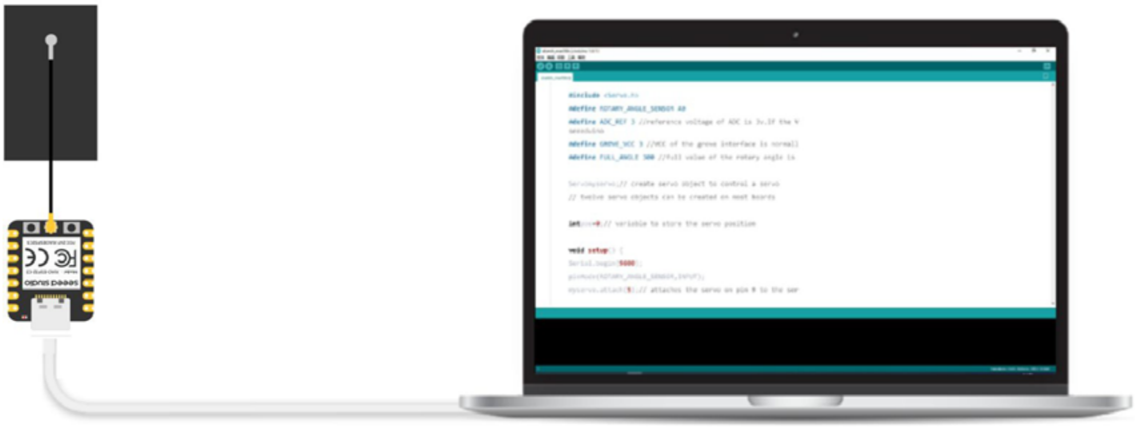
> *Be careful when attempting to use the XIAO ESP32C3 development board as a hotspot (access point). Overheating issues may occur and lead to burns.*

Hardware Setup: Connect an Antenna to the XIAO ESP32C3 and Connect it to Your Computer

Step 1: Connect the provided Wi-Fi/Bluetooth antenna to the IPEX connector on the development board.

Step 2: Connect the XIAO ESP32C3 to your computer via a USB Type-C data cable.



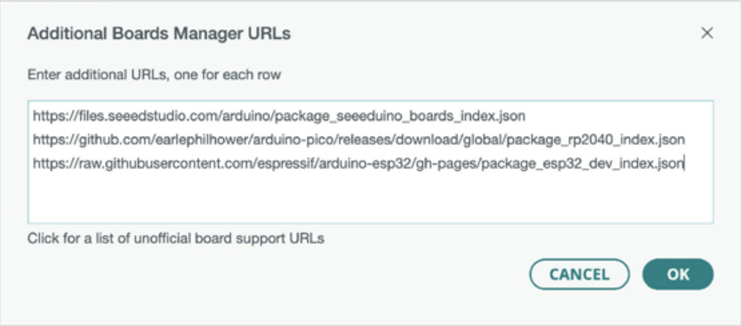


Software Setup: Add the ESP32 Board Package to the Arduino IDE

Step 1: Open the Arduino IDE preferences to add the Board Manager URL.

- For Windows users, first open your Arduino IDE, click on “File→Preferences” in the top menu bar, and copy the following URL into “Additional Board Manager URLs”.
- For Mac users, first open your Arduino IDE, click on “Arduino IDE→Preferences” in the top menu bar, and copy the following URL into “Additional Board Manager URLs”.

For Seeed Studio XIAO ESP32C3, copy the link below: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json to the Board Manager URL bar and confirm, as shown in the figure below.



Additional Boards Manager URLs

Enter additional URLs, one for each row

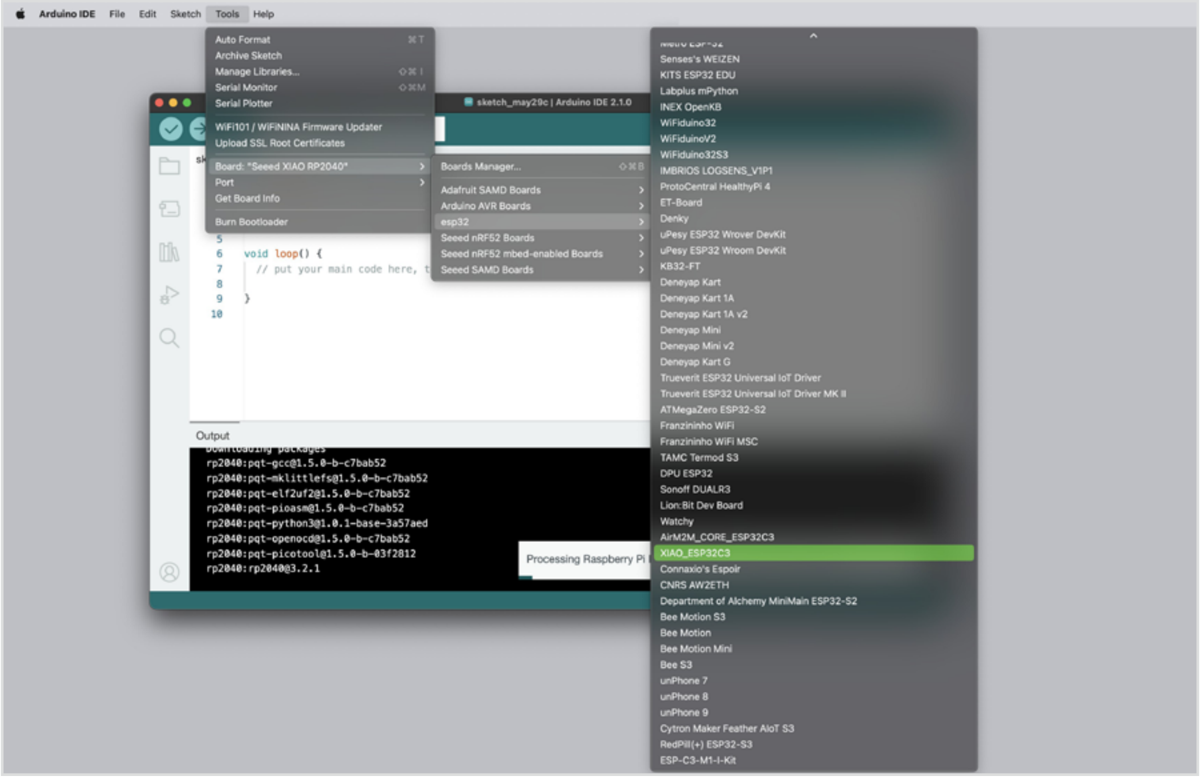
https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json
https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

Click for a list of unofficial board support URLs

Step 2: In the Arduino IDE menu, click “Tools→Board→Board Manager”, type “esp32” into the search bar, find the latest version of ESP32 Arduino in the resulting entries, and click “Install”. When the installation starts, you will see an output pop-up. Once the installation is complete, the “Installed” option will appear.

Step 3: Select the Board.

Navigate to “Tools > Board > ESP32 Arduino” and select “XIAO_ESP32C3”. The list will be a bit long, and you will need to scroll down to find it, as shown in the figure below.



```
void loop() {
  // put your main code here, t
}
```

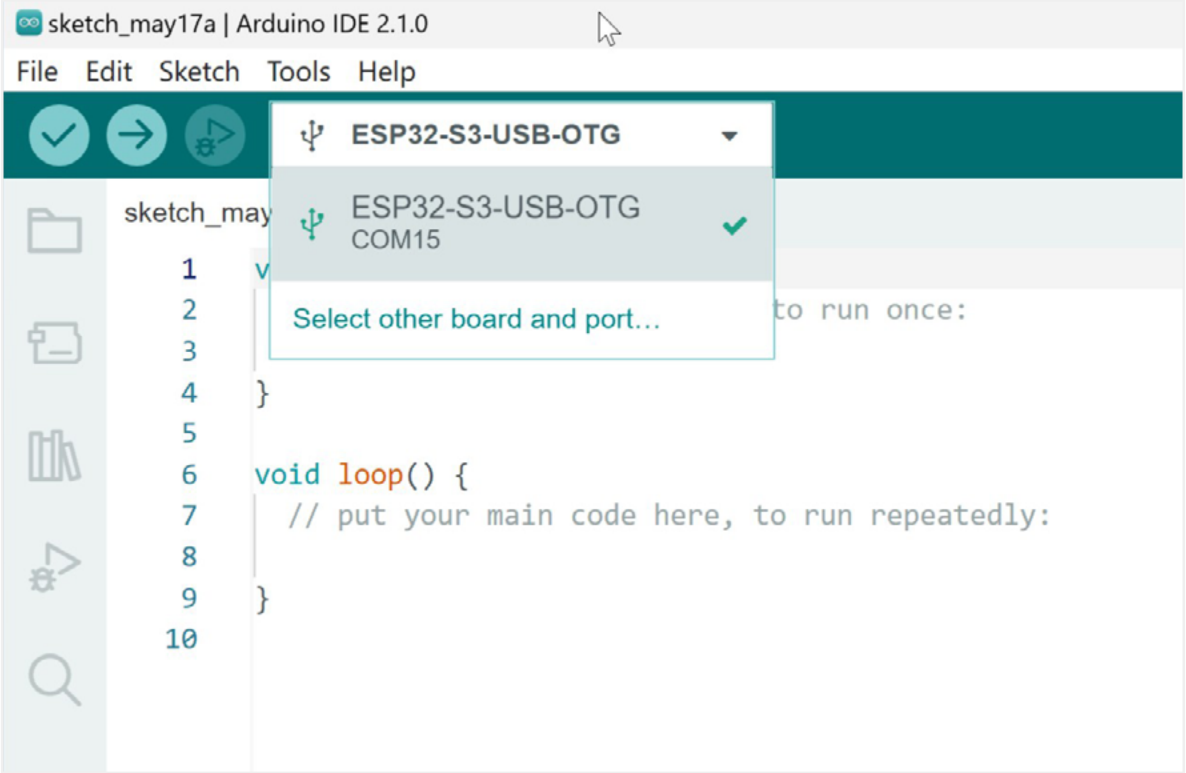
```
Downloading packages
rp2040:pqt-gcc@1.5.0-b-c7bab52
rp2040:pqt-mklittlefs@1.5.0-b-c7bab52
rp2040:pqt-elf2uf2@1.5.0-b-c7bab52
rp2040:pqt-pioasm@1.5.0-b-c7bab52
rp2040:pqt-python3@1.0.1-base-3a57aed
rp2040:pqt-openocd@1.5.0-b-c7bab52
rp2040:pqt-picotool@1.5.0-b-03f2812
rp2040:rp2040@3.2.1
```

Step 4: Add Port.

Check if the port connection is correct in the Arudino IDE. If not, you need to manually select.

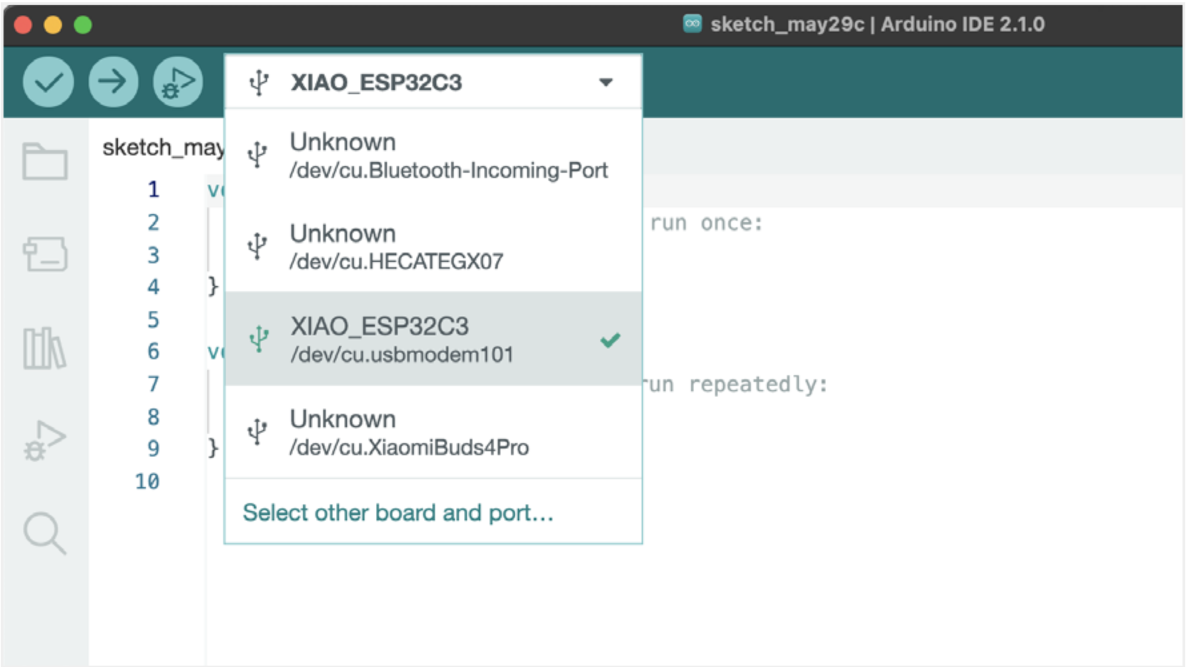
For Windows systems, the serial port is displayed as "COM+number", as shown in the figure below.

- For Windows systems, the serial port is displayed as "COM+number", as shown in the figure below.



```
v
                  to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

- On Mac or Linux systems, the port name is typically `/dev/tty.usbmodem+number` or `/dev/cu.usbmodem+number`, as shown in the figure below.



```
vo
             run once:
}

vo
             un repeatedly:

}
```

Scanning Nearby Wi-Fi Networks (STA Mode)

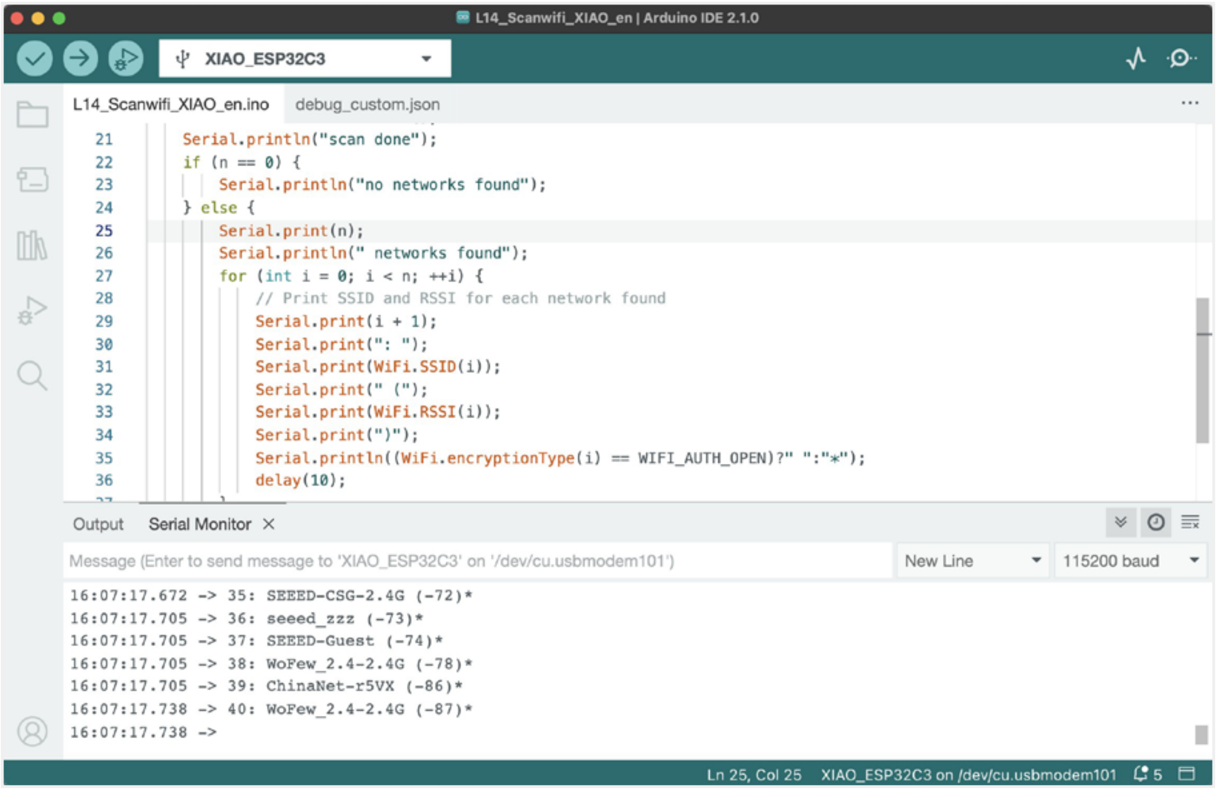
In this example, we will use the XIAO ESP32C3 to scan for available Wi-Fi networks in the area. The development board in this example will be configured in STA mode.

Step 1: Copy and paste the code below into the Arduino IDE.

```
#include “WiFi.h”

void setup()
{
    Serial.begin(115200);

    // Set WiFi to station mode and disconnect from an AP if it was previously con-
nected
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

    Serial.println(“Setup done”);
}

void loop()
{
    Serial.println(“scan start”);

    // WiFi.scanNetworks will return the number of networks found
    int n = WiFi.scanNetworks();
    Serial.println(“scan done”);
    if (n == 0) {
        Serial.println(“no networks found”);
    } else {
        Serial.print(n);
        Serial.println(“ networks found”);
        for (int i = 0; i < n; ++i) {
            // Print SSID and RSSI for each network found
            Serial.print(i + 1);
            Serial.print(“: “);
            Serial.print(WiFi.SSID(i));
            Serial.print(“ (“);
            Serial.print(WiFi.RSSI(i));
            Serial.print(“)”);
            Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN)?” “:”*”);
            delay(10);
        }
    }
    Serial.println(“”);

    // Wait a bit before scanning again
    delay(5000);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L14_Scanwifi_XIAO_en

Step 2: Upload the code and open the serial monitor to start scanning for Wi-Fi networks, as shown in the figure below.



L14_Scanwifi_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L14_Scanwifi_XIAO_en.ino
debug_custom.json
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 25, Col 25
XIAO_ESP32C3 on /dev/cu.usbmodem101
5

```
    Serial.println("scan done");
    if (n == 0) {
        Serial.println("no networks found");
    } else {
        Serial.print(n);
        Serial.println(" networks found");
        for (int i = 0; i < n; ++i) {
            // Print SSID and RSSI for each network found
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.print(WiFi.SSID(i));
            Serial.print(" (");
            Serial.print(WiFi.RSSI(i));
            Serial.print(")");
            Serial.println((WiFi.encryptionType(i) == WIFI_AUTH_OPEN)?" ":"*");
            delay(10);
```

```
16:07:17.672 -> 35: SEEED-CSG-2.4G (-72)*
16:07:17.705 -> 36: seeed_zzz (-73)*
16:07:17.705 -> 37: SEEED-Guest (-74)*
16:07:17.705 -> 38: WoFew_2.4-2.4G (-78)*
16:07:17.705 -> 39: ChinaNet-r5VX (-86)*
16:07:17.738 -> 40: WoFew_2.4-2.4G (-87)*
16:07:17.738 ->
```

Connecting to a Wi-Fi Network

In this example, we will use the XIAO ESP32C3 to connect to your Wi-Fi network.

Step 1: Copy and paste the code below into the Arduino IDE.

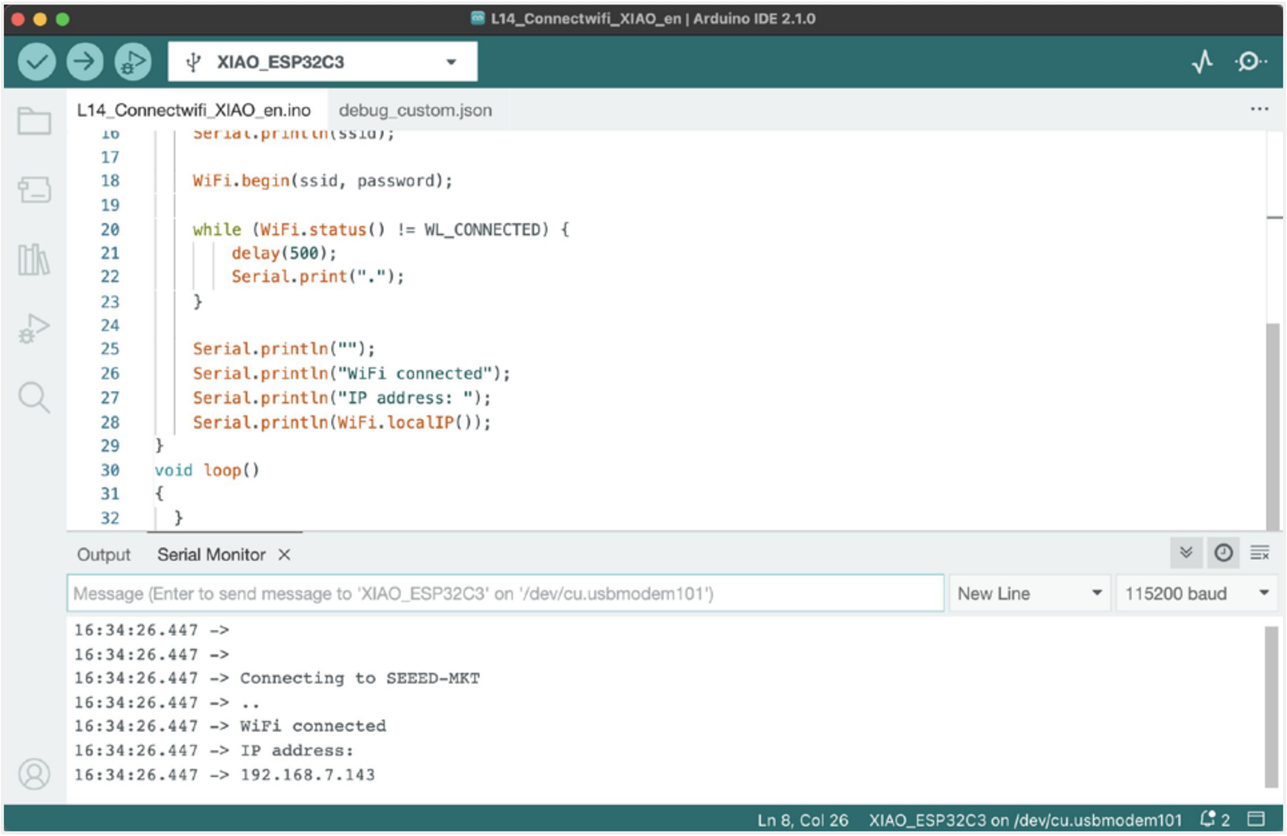
```
#include <WiFi.h>

const char* ssid     = “your-ssid”;
const char* password = “your-password”;

void setup()
{
    Serial.begin(115200);
    delay(10);

    // We start by connecting to a WiFi network

    Serial.println();
    Serial.println();
    Serial.print(“Connecting to “);
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(“.”);
    }

    Serial.println(“”);
    Serial.println(“WiFi connected”);
```

```
        Serial.println(“IP address: “);
        Serial.println(WiFi.localIP());
    }
    void loop()
    {
      }
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L14_Connectwifi_XIAO_en

Then, replace `your-ssid` in the code with the name of your Wi-Fi network, and replace `your-password` in the code with the password for your Wi-Fi network.

Step 2: Upload the code and open the serial monitor to check whether the development board is connected to the Wi-Fi network, as shown in the figure below.


L14_Connectwifi_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L14_Connectwifi_XIAO_en.ino
debug_custom.json
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 8, Col 26
XIAO_ESP32C3 on /dev/cu.usbmodem101

```
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void loop()
{
  }
```

```
16:34:26.447 ->
16:34:26.447 ->
16:34:26.447 -> Connecting to SEEED-MKT
16:34:26.447 -> ..
16:34:26.447 -> WiFi connected
16:34:26.447 -> IP address:
16:34:26.447 -> 192.168.7.143
```

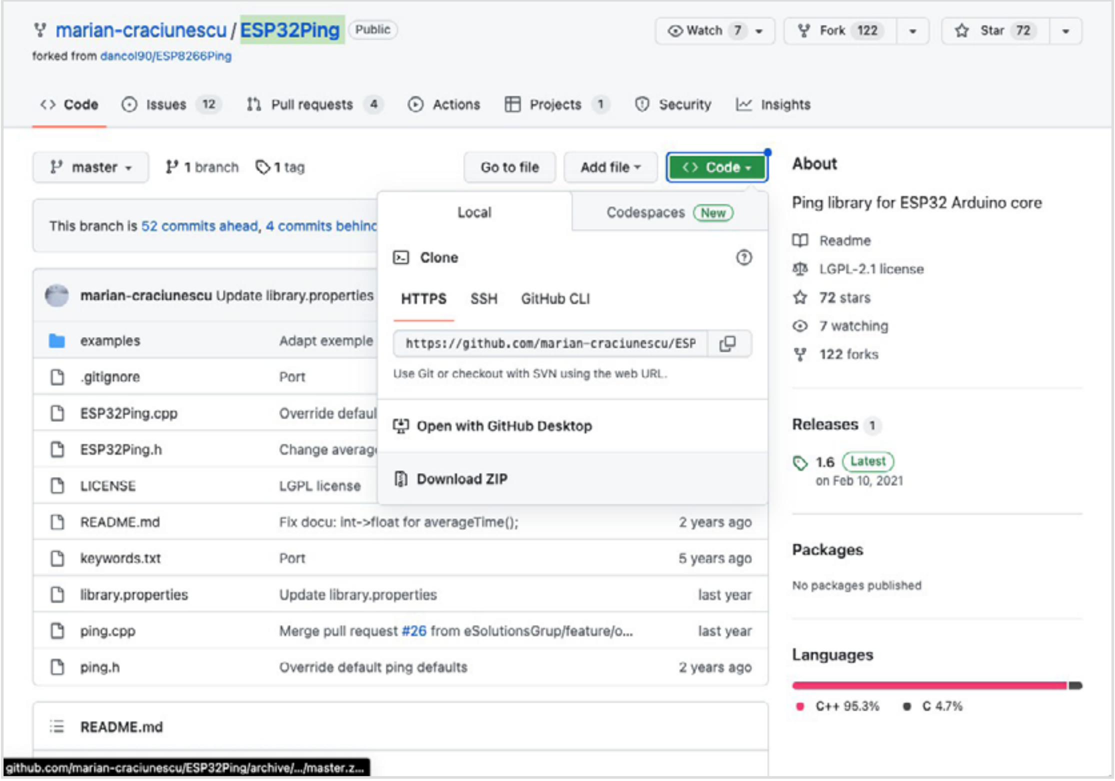
Learn more

You can read the Wiki documentation for more about using the XIAO ESP32C3.

3.4.3 Task 2: Ping a Specified Website

With the knowledge above, we can now learn how to use the XIAO ESP32C3 to ping a specified website.

Step 1: Download and install the ESP32Ping library. Enter the URL https://github.com/marian-craciunescu/ESP32Ping to go to the GitHub page, click on Code→Download ZIP to download the resource pack to your local machine, as shown in the figure below.

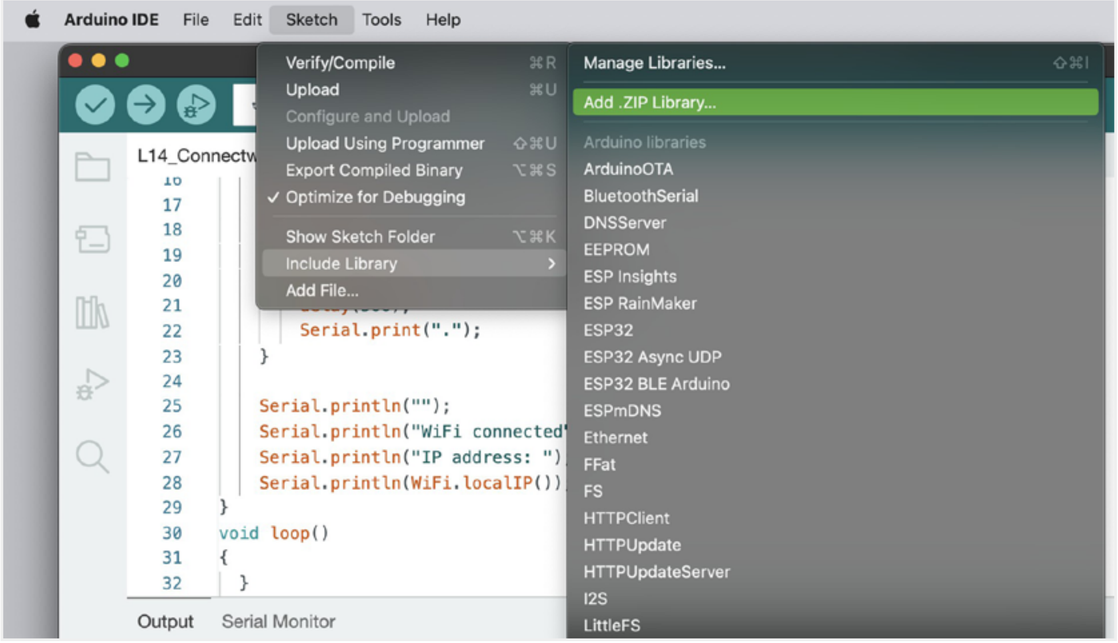


This branch is 52 commits ahead, 4 commits behind

marian-craciunescu Update library.properties

examples	Adapt exemple	
.gitignore	Port	
ESP32Ping.cpp	Override defaul	
ESP32Ping.h	Change averag	
LICENSE	LGPL license	
README.md	Fix docu: int->float for averageTime();	2 years ago
keywords.txt	Port	5 years ago
library.properties	Update library.properties	last year
ping.cpp	Merge pull request #26 from eSolutionsGrup/feature/o...	last year
ping.h	Override default ping defaults	2 years ago

After downloading, open the Arduino IDE, click on Sketch→Include Library→Add .ZIP Library, and choose the ZIP file you just downloaded.



```
      Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected'
    Serial.println("IP address: ")
    Serial.println(WiFi.localIP())
}
void loop()
{
  }
```

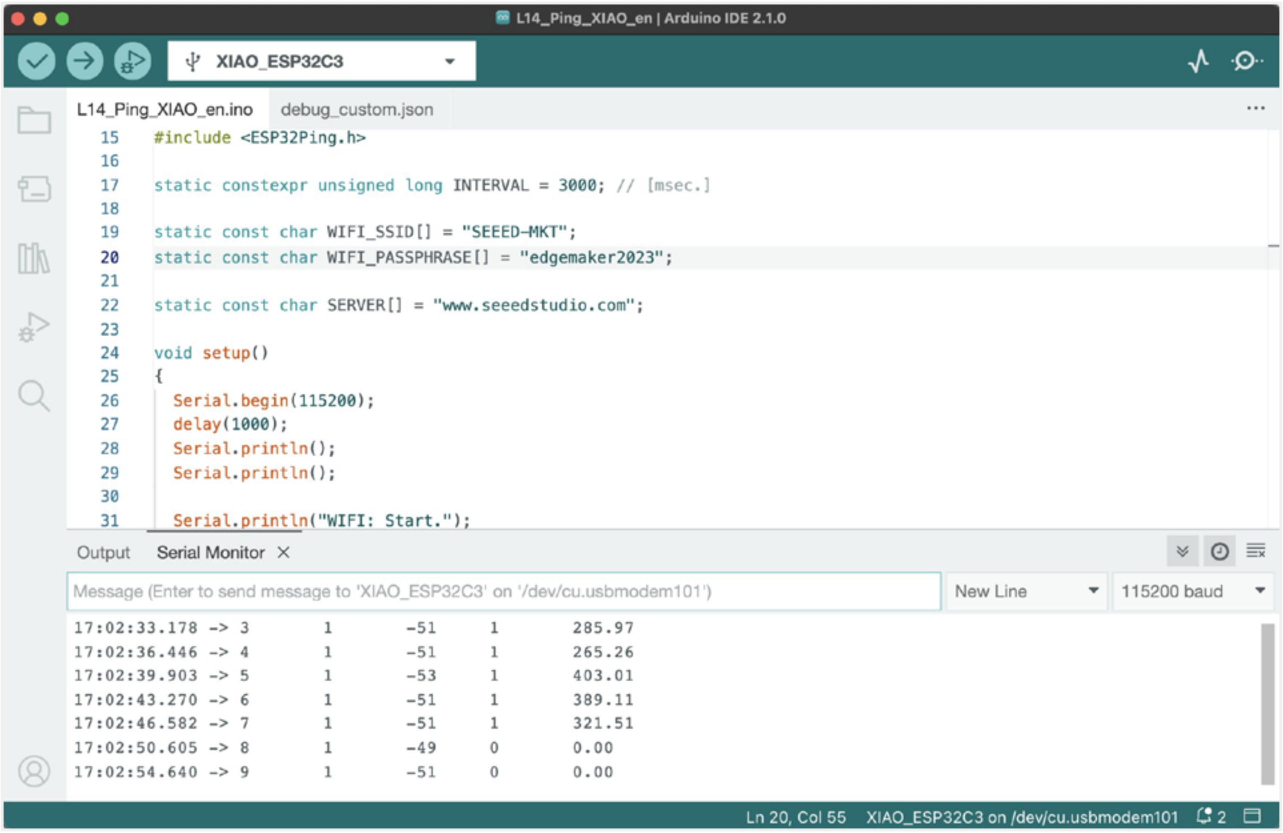
Step 2: Copy and paste the code below into the Arduino IDE. This code sets the test website to `www.seeedstudio.com`. Remember to replace `your-ssid` in the code with your Wi-Fi network name and `your-password` in the code with your Wi-Fi password.

```
////////////////////////////////////////////////////////////////////////////////
// IDE:
//   Arduino 2.0.3
// Platform:
```

```
//   esp32 2.0.4 - https://github.com/espressif/arduino-esp32
// Board:
//   XIAO_ESP32C3
// Libraries:
//   ESP32Ping 1.6 - https://github.com/marian-craciunescu/ESP32Ping

////////////////////////////////////////////////////////////////////////////////
// Includes

#include <WiFi.h>
#include <ESP32Ping.h>

static constexpr unsigned long INTERVAL = 3000; // [msec.]

static const char WIFI_SSID[] = “your-ssid”;
static const char WIFI_PASSPHRASE[] = “your-password”;

static const char SERVER[] = “www.google.com”;

void setup()
{
    Serial.begin(115200);
    delay(1000);
    Serial.println();
    Serial.println();

    Serial.println(“WIFI: Start.”);
    WiFi.mode(WIFI_STA);
    if (WIFI_SSID[0] != ‘\0’)
    {
        WiFi.begin(WIFI_SSID, WIFI_PASSPHRASE);
    }
    else
    {
        WiFi.begin();
    }
}

void loop()
{
    static int count = 0;

    const bool wifiStatus = WiFi.status() == WL_CONNECTED;
    const int wifiRssi = WiFi.RSSI();

    const bool pingResult = !wifiStatus ? false : Ping.ping(SERVER, 1);
    const float pingTime = !pingResult ? 0.f : Ping.averageTime();

    Serial.print(count);
    Serial.print(‘\t’);
    Serial.print(wifiStatus ? 1 : 0);
    Serial.print(‘\t’);
    Serial.print(wifiRssi);
    Serial.print(‘\t’);
    Serial.print(pingResult ? 1 : 0);
    Serial.print(‘\t’);
    Serial.println(pingTime);
    count++;
    delay(INTERVAL);
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L14_Ping_XIAO_en

Step 3: Upload the code and open the serial monitor to check the ping results, as shown in the figure below.



```
#include <ESP32Ping.h>

static constexpr unsigned long INTERVAL = 3000; // [msec.]

static const char WIFI_SSID[] = "SEEED-MKT";
static const char WIFI_PASSPHRASE[] = "edgemaker2023";

static const char SERVER[] = "www.seeedstudio.com";

void setup()
{
  Serial.begin(115200);
  delay(1000);
  Serial.println();
  Serial.println();

  Serial.println("WIFI: Start.");
```

Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')

```
17:02:33.178 -> 3       1       -51     1       285.97
17:02:36.446 -> 4       1       -51     1       265.26
17:02:39.903 -> 5       1       -53     1       403.01
17:02:43.270 -> 6       1       -51     1       389.11
17:02:46.582 -> 7       1       -51     1       321.51
17:02:50.605 -> 8       1       -49     0       0.00
17:02:54.640 -> 9       1       -51     0       0.00
```

3.4.4 Project Creation: Using XIAO ESP32C3 to Make HTTP GET and HTTP POST Requests

Introduction to HTTP Protocol

HTTP stands for HyperText Transfer Protocol. It's an application-layer protocol for distributed, collaborative, and hypermedia information systems. HTTP is the most widely used network transmission protocol on the Internet, and all WWW files must comply with this standard.

HTTP is designed for communication between Web browsers and Web servers, but it can also be used for other purposes. HTTP is a protocol that uses TCP/IP to transmit data (such as HTML files, image files, query results, etc.).

Despite its wide use, HTTP has significant security flaws, mainly its plain text data transmission and lack of message integrity checks. These are exactly the two most critical security aspects in emerging applications like online payment, online trading, Internet of Things, etc. Browsers like Google Chrome, Internet Explorer, and Firefox issue warnings about insecure connections when visiting websites with mixed content composed of encrypted and unencrypted content using HTTP.

Introduction to HTTPS Protocol

HTTPS stands for HyperText Transfer Protocol Secure. It's a protocol for secure communication over a computer network. HTTPS communicates via HTTP but uses SSL/TLS to encrypt packets. The main purpose of HTTPS is to authenticate the website server's identity and protect the privacy and integrity of the exchanged data.

HTTP
+
SSL
=
HTTPS

HTTP Request Methods

According to the HTTP standard, HTTP requests can use multiple request methods.

HTTP1.0 defined three request methods: GET, POST, and HEAD.

HTTP1.1 added six new request methods: OPTIONS, PUT, PATCH, DELETE, TRACE, and CONNECT.

No.	Method	Description
1	GET	Requests specified page information and returns the entity body.
2	HEAD	Similar to a GET request, but the response returned doesn't contain specific content, used to obtain headers.
3	POST	Submits data for processing to a specified resource (e.g., submits a form or uploads a file). The data is included in the request body. POST requests may result in the creation of a new resource and/or the modification of an existing resource.
4	PUT	The data sent from the client to the server replaces the content of a specified document.
5	DELETE	Requests the server to delete a specified page.
6	CONNECT	Reserved in HTTP/1.1 for proxy servers that can switch the connection to a pipe mode.
7	OPTIONS	Allows the client to view the server's capabilities.
8	TRACE	Echoes the request received by the server, mainly used for testing or diagnosis.
9	PATCH	It's a supplement to the PUT method, used to partially update a known resource.

We've already learned how to connect to a Wi-Fi network using XIAO ESP32C3. Now, let's try some more complex operations based on the network. The following sections will introduce how to use XIAO ESP32C3 to send HTTP GET and HTTP POST requests.

Task 3: Using XIAO ESP32C3 to Send an HTTP GET Request

To send an HTTP GET request, a corresponding backend server that supports the request is required. For convenient testing, we can set up a backend server on our own PC, allowing XIAO ESP32C3 to send an HTTP GET request to the PC through the local Wi-Fi connection.

There are many ways to set up a backend service. In this case, we'll use the popular Python web framework — FastAPI to set up the backend server. To learn more about this tool, visit its official documentation.

Setting Up a Backend Server with FastAPI

Here is the Python server code.

```
from typing import Union
from pydantic import BaseModel
from fastapi import FastAPI
import datetime

app = FastAPI()
items = {}

class Sensor_Item(BaseModel):
    name: str
    value: float

@app.on_event(“startup”)
async def startup_event():
    items[“sensor”] = {“name”: “Sensor”,”Value”:0}

@app.get(“/items/{item_id}”)
async def read_items(item_id: str):

    return items[item_id],datetime.datetime.now()

@app.post(“/sensor/”)
async def update_sensor(si: Sensor_Item):
    items[“sensor”][“Value”] = si.value
    return si

@app.get(“/”)
def read_root():
    return {“Hello”: “World”}
```

This code snippet, implemented using the Python FastAPI framework, can return the latest information of the Sensor stored on the backend server when we use a `get` request on `http://domain/items/sensor`. When we use `post` to send data to `http://domain/sensor/`, it can modify and record the latest Sensor value. The operation steps are as follows:

Step 1: Create a python file named `main.py` locally, copy and paste the code above into `main.py`. Then, on your PC, open the terminal and execute the following commands to install FastAPI.

```
pip install fastapi
pip install “uvicorn[standard]”
```

Step 2: Execute the following command to start the backend service and local monitoring.

```
uvicorn main:app –-reload –-host 0.0.0.0
```

- Attention -

When running the command above, make sure the terminal is currently in the directory where `main:app` *resides. If there is a prompt during running:*

```
ERROR:    [Errno 48] Address already in use
```

This means the current address is already occupied and there is an address conflict. You can specify a specific port as shown in the command below.

```
uvicorn main:app --reload --host 0.0.0.0 --port 1234
```

If the [Errno 48] error still appears, you can modify the port number after port.

The prompt information after the command is successfully run is as follows

```
INFO:     Will watch for changes in these directories: [‘’]
INFO:     Uvicorn running on http://0.0.0.0:1234 (Press CTRL+C to quit)
INFO:     Started reloader process [53850] using WatchFiles
INFO:     Started server process [53852]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

The backend server for testing is now running normally.

Using XIAO ESP32C3 to Send an HTTP GET Request

Next, we'll perform a request test on XIAO ESP32C3.

The GET method should only be used for reading data, and should not be used in operations that generate “side effects”. GET requests directly issued by browsers can only be triggered by a URL. If you want to carry some parameters outside of the URL in GET, you can only rely on the querystring (query string) attached to the URL.

Step 1: Copy and paste the following code into the Arduino IDE. This code sets the tested `serverName` to `http://192.168.1.2/items/sensor`. The `192.168.1.2` needs to be replaced with the IP address of your PC acting as the backend server. To get the IP address of your PC, Windows users can enter the `ipconfig` command in the command line window, and Mac users can enter the `ifconfig` command in the terminal window. Remember to change `your-ssid` in the code to your Wi-Fi network name and `your-password` to the corresponding Wi-Fi password.

```
#include “WiFi.h”
#include <HTTPClient.h>

const char* ssid = “your-ssid”;
const char* password = “your-password”;
String serverName = “http://192.168.1.2/items/sensor”;
unsigned long lastTime = 0;
unsigned long timerDelay = 5000;

void setup()
{
    Serial.begin(115200);

    WiFi.begin(ssid, password);
    Serial.println(“Connecting”);
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(“.”);
    }
    Serial.println(“”);
    Serial.print(“Connected to WiFi network with IP Address: “);
    Serial.println(WiFi.localIP());

     Serial.println(“Timer set to 5 seconds (timerDelay variable), it will take 5
seconds before publishing the first reading.”);

    Serial.println(“Setup done”);
}

void loop()
{
    if ((millis() – lastTime) > timerDelay) {
        //Check WiFi connection status
        if(WiFi.status()== WL_CONNECTED){
            HTTPClient http;

            String serverPath = serverName ;

            http.begin(serverPath.c_str());

            int httpResponseCode = http.GET();

            if (httpResponseCode>0) {
                Serial.print(“HTTP Response code: “);
                Serial.println(httpResponseCode);
                String payload = http.getString();
                Serial.println(payload);
            }
            else {
                Serial.print(“Error code: “);
                Serial.println(httpResponseCode);
            }

            http.end();
        }
        else {
            Serial.println(“WiFi Disconnected”);
        }
        lastTime = millis();
    }
}
```

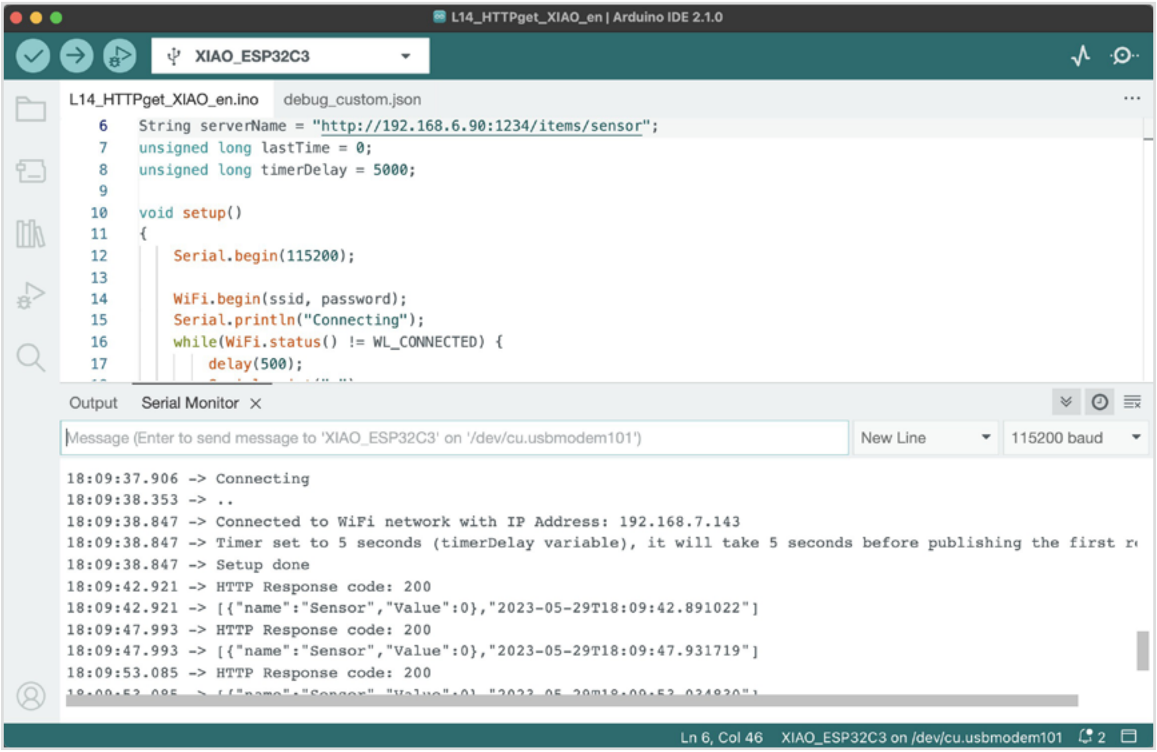
Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L14_HTTPget_XIAO_en

- Attention -

We need to change the `serverName` *in the Arduino code to the IP address of the host running the backend service. The XIAO ESP32C3 needs to be on the same local area network as it. If the local area network IP of the backend server (in this example, your PC) is* `192.168.1.2`*, then the GET request interface is* `http://192.168.1.2/items/sensor`*, and other interfaces are similar. If you specified a port when running the backend service and*

> *local monitoring, the GET request interface would be* `http://192.168.1.2:1234/items/sensor`.

Step 2: Upload the code to XIAO ESP32C3 in the Arduino IDE. After the upload is successful, open the serial monitor to check the result returned by our backend server after the GET is issued, as shown in the figure below.


L14_HTTPget_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L14_HTTPget_XIAO_en.ino
debug_custom.json
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 6, Col 46
XIAO_ESP32C3 on /dev/cu.usbmodem101
2

```
String serverName = "http://192.168.6.90:1234/items/sensor";
unsigned long lastTime = 0;
unsigned long timerDelay = 5000;

void setup()
{
    Serial.begin(115200);

    WiFi.begin(ssid, password);
    Serial.println("Connecting");
    while(WiFi.status() != WL_CONNECTED) {
        delay(500);
```

```
18:09:37.906 -> Connecting
18:09:38.353 -> ..
18:09:38.847 -> Connected to WiFi network with IP Address: 192.168.7.143
18:09:38.847 -> Timer set to 5 seconds (timerDelay variable), it will take 5 seconds before publishing the first r
18:09:38.847 -> Setup done
18:09:42.921 -> HTTP Response code: 200
18:09:42.921 -> [{"name":"Sensor","Value":0},"2023-05-29T18:09:42.891022"]
18:09:47.993 -> HTTP Response code: 200
18:09:47.993 -> [{"name":"Sensor","Value":0},"2023-05-29T18:09:47.931719"]
18:09:53.085 -> HTTP Response code: 200
18:09:53.085 -> [{"name":"Sensor","Value":0},"2023-05-29T18:09:53.034830"]
```

The prompt `HTTP Response code: 200` means the request has been successful, and our XIAO ESP32C3 has successfully gotten data from the server.

Task 4: Using XIAO ESP32C3 to Send an HTTP POST Request

Submit data to a specified resource and request the server to process it (for example, submit a form or upload a file). The data is included in the request body. This request may create new resources or modify existing resources, or both. Each time it is submitted, the form data is encoded into the body of the HTTP request by the browser. The body of a POST request issued by a browser mainly has two formats, one is `application/x-www-form-urlencoded` used to transmit simple data, roughly in the format of `key1=value1&key2=value2`. The other is for transmitting files and will use the `multipart/form-data` format. The latter is adopted because the encoding method of `application/x-www-form-urlencoded` is very inefficient for binary data like files.

Next, we will submit experimental data to the backend server built on our machine in a manner similar to submitting a form, and verify whether the backend server has received the data.

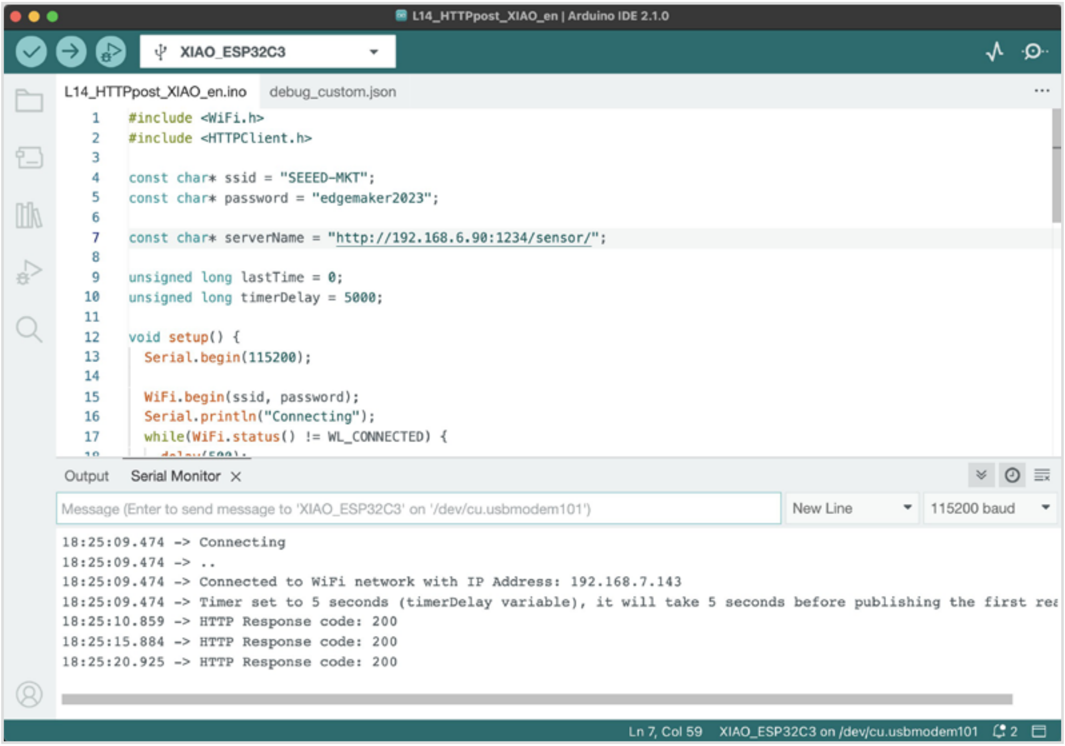
Step 1: Before starting this example, make sure that the backend server built with FastAPI in the previous step is running normally. If not, please refer to the above instructions to start the server program.

Step 2: Copy and paste the following code into the Arduino IDE. This code sets the tested `serverName` to `http://192.168.1.2/sensor/`. The `192.168.1.2` needs to be replaced with the IP address of your PC acting as the backend server. Remember to change `your-ssid` in the code to your Wi-Fi network name and `your-password` to the corresponding Wi-Fi password.

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = “your-ssid”;
const char* password = “your-password”;

const char* serverName = “https://192.168.1.2/sensor/”;

unsigned long lastTime = 0;
unsigned long timerDelay = 5000;

void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);
  Serial.println(“Connecting”);
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(“.”);
  }
  Serial.println(“”);
  Serial.print(“Connected to WiFi network with IP Address: “);
  Serial.println(WiFi.localIP());

  Serial.println(“Timer set to 5 seconds (timerDelay variable), it will take 5 sec-
onds before publishing the first reading.”);
}

void loop() {
  //Send an HTTP POST request every 10 minutes
  if ((millis() - lastTime) > timerDelay) {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
      WiFiClient client;
      HTTPClient http;


      http.begin(client, serverName);

      http.addHeader(“Content-Type”, “application/json”);
      int httpResponseCode = http.POST(“{\”name\”:\”sensor\”,\”value\”:\”123\”}”);

      Serial.print(“HTTP Response code: “);
      Serial.println(httpResponseCode);

      // Free resources
      http.end();
    }
    else {
      Serial.println(“WiFi Disconnected”);
    }
    lastTime = millis();
  }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L14_HTTPpost_XIAO_en

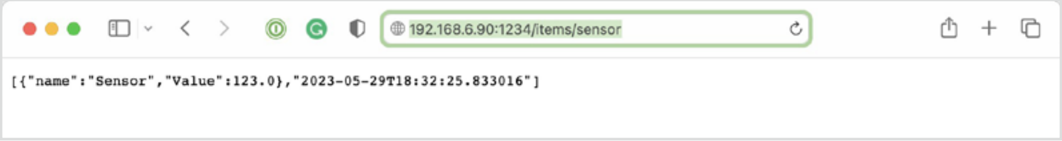
Step 2: Upload the code to XIAO ESP32C3 using the Arduino IDE. After a successful upload, open the Serial Monitor to examine the result returned by our backend server in response to the GET request. The image below illustrates the process.


L14_HTTPpost_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L14_HTTPpost_XIAO_en.ino
debug_custom.json
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 7, Col 59
XIAO_ESP32C3 on /dev/cu.usbmodem101
2

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "SEEED-MKT";
const char* password = "edgemaker2023";

const char* serverName = "http://192.168.6.90:1234/sensor/";

unsigned long lastTime = 0;
unsigned long timerDelay = 5000;

void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
```

```
18:25:09.474 -> Connecting
18:25:09.474 -> ..
18:25:09.474 -> Connected to WiFi network with IP Address: 192.168.7.143
18:25:09.474 -> Timer set to 5 seconds (timerDelay variable), it will take 5 seconds before publishing the first rea
18:25:10.859 -> HTTP Response code: 200
18:25:15.884 -> HTTP Response code: 200
18:25:20.925 -> HTTP Response code: 200
```

The message `**HTTP Response code: 200**` signifies a successful request. On your local PC, open a browser and navigate to `**http://192.168.1.2/items/sensor**` (please replace the IP address according to your actual PC's IP address, and if a port has been set, append a colon followed by the designated port number after the IP address). You should now see the most recent data sent by the XIAO ESP32C3. Since XIAO sends data every 5 seconds, you can always view the most recent data received by the backend server by refreshing the current page (the timestamp of the data will change).



```
[{"name":"Sensor","Value":123.0},"2023-05-29T18:32:25.833016"]
```

We have now successfully sent data from XIAO ESP32C3 to the local backend server.

3.5
Telemetry and Commands using the MQTT protocol with XIAO ESP32C3

In the previous section, we learned how to send HTTP GET or POST requests from the XIAO ESP32C3 to a local machine on a local area network via Wi-Fi. In this section, we'll step through: communication protocols, Message Queuing Telemetry Transport (MQTT), telemetry (data gathered from sensors and sent to the cloud), and commands (messages sent from the cloud to a device instructing it to do something).

3.5.1 Background Knowledge

IoT (Internet of Things)

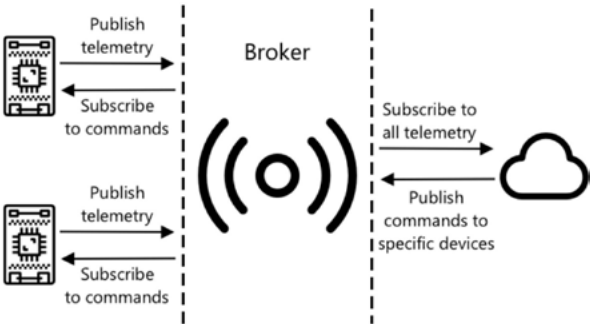
The "I" in IoT stands for Internet—cloud connectivity and services that enable many of the functions of IoT devices, from gathering sensor measurements linked to devices, to sending messages to control actuators. IoT devices typically connect to a single IoT cloud service using standard communication protocols, and this service is tightly integrated with the rest of your IoT application, from AI services making intelligent decisions around data, to web applications for control or reporting.

> *Data collected from sensors and sent to the cloud is called telemetry.*

IoT devices can also receive information from the cloud. This information typically consists of commands—instructions to perform internal actions (such as rebooting or updating firmware) or to actuate (e.g., turning on a light).

Communication Protocols

There are many popular communication protocols that IoT devices use to communicate with the internet. The most popular are based around the publishing/subscribing of messages through some agent: IoT devices connect to the agent, publish telemetry data and subscribe to commands. Cloud services also connect to the agent, subscribe to all telemetry information, and publish commands to specific devices or groups of devices, as shown in the figure below.



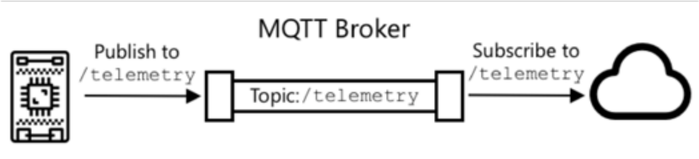
MQTT is the most popular communication protocol for IoT devices and will be covered in this section. Other protocols include AMQP and HTTP/HTTPS, which we introduced in the previous section.

Message Queuing Telemetry Transport (MQTT)

MQTT is short for Message Queuing Telemetry Transport. It is a messaging protocol based on the publish/subscribe paradigm under the ISO standard: ISO/IEC PRF 20922. It can be seen as a "bridge for data delivery". It operates on top of the TCP/IP protocol stack and is a publish/subscribe type messaging protocol designed for remote devices with poor hardware performance and poor network conditions. It is a lightweight, open standard messaging transport protocol that can send messages between devices. Originally designed in 1999 for monitoring oil pipelines, it was published as an open standard by IBM 15 years later.

The biggest advantage of MQTT is that it provides a real-time and reliable messaging service for connecting remote devices with minimal code and limited bandwidth. As a low-overhead, low-bandwidth consumption instant communication protocol, it is widely used in IoT, small devices, mobile applications, and so on.

MQTT has one broker and multiple clients. All clients connect to the broker, which then routes messages to the relevant clients. Messages are routed using named topics, not sent directly to a single client. Clients can publish to a topic, and any client subscribed to that topic will receive the message.



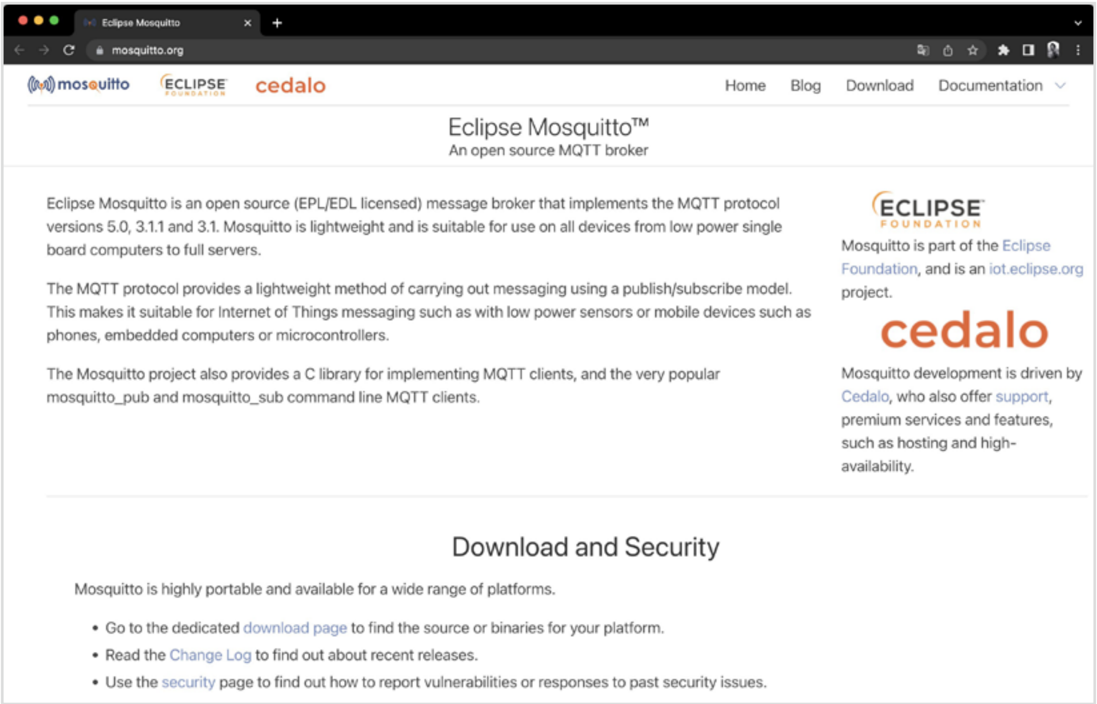
Do some research. If you have a large number of IoT devices, how can you ensure that your MQTT broker can handle all messages?

Some Open Source MQTT Brokers

While we can set up our own MQTT broker if circumstances allow, you might not be ready to delve into server and application setup yet. If you're just learning, you can start with some open source MQTT brokers.

Eclipse Mosquitto https://www.mosquitto.org/

This is an open source MQTT broker. Instead of dealing with the complexities of setting up an MQTT broker as part of this task, this test broker is publicly available at test.mosquitto.org and doesn't require account setup. It's a great tool for testing MQTT clients and servers.



Eclipse Mosquitto™
An open source MQTT broker

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers.

The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers.

The Mosquitto project also provides a C library for implementing MQTT clients, and the very popular mosquitto_pub and mosquitto_sub command line MQTT clients.

ECLIPSE FOUNDATION

Mosquitto is part of the Eclipse Foundation, and is an iot.eclipse.org project.

cedalo

Mosquitto development is driven by Cedalo, who also offer support, premium services and features, such as hosting and high-availability.

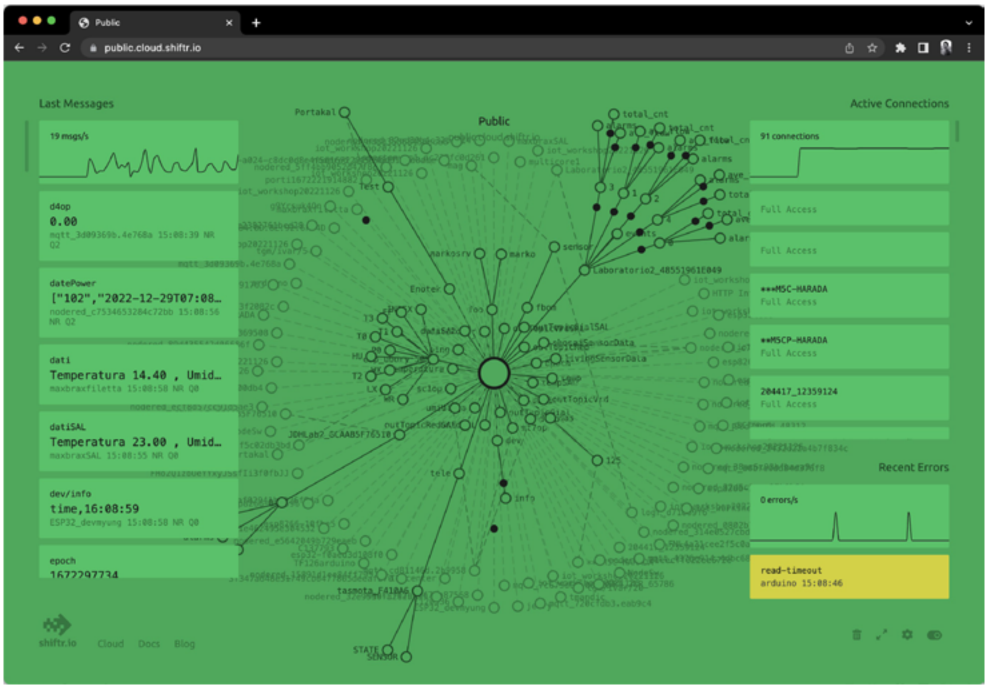
Download and Security

Mosquitto is highly portable and available for a wide range of platforms.

- Go to the dedicated download page to find the source or binaries for your platform.
- Read the Change Log to find out about recent releases.
- Use the security page to find out how to report vulnerabilities or responses to past security issues.

shiftr.io

An IoT platform for interconnected projects, quickly connect hardware and software with its cloud service and desktop applications. The platform also provides a clear view of all connections, topics, and messages in the network through real-time charts. The shiftr.io broker supports MQTT and HTTP for publishing, subscribing, and retrieving messages, and the platform supports free accounts, enough for us to learn and use. They also provide a public server at public.cloud.shiftr.io with username `public` on ports 1883 (MQTT) and 8883 (MQTTS). The animated view of connected services and data being exchanged on the public server is very cool, as shown in the image below.



19 msgs/s

d4op
0.00
mqtt_3d09369b.4e768a 15:08:39 NR Q2

datePower
["102","2022-12-29T07:08…
nodered_c7534653284c72bb 15:08:56 NR Q2

dati
Temperatura 14.40 , Umid…
maxbraxfiletta 15:08:58 NR Q0

datiSAL
Temperatura 23.00 , Umid…
maxbraxSAL 15:08:55 NR Q0

dev/info
time,16:08:59
ESP32_devmyung 15:08:58 NR Q0

epoch
1672297734

91 connections

Full Access

Full Access

***M5C-HARADA
Full Access

***M5CP-HARADA
Full Access

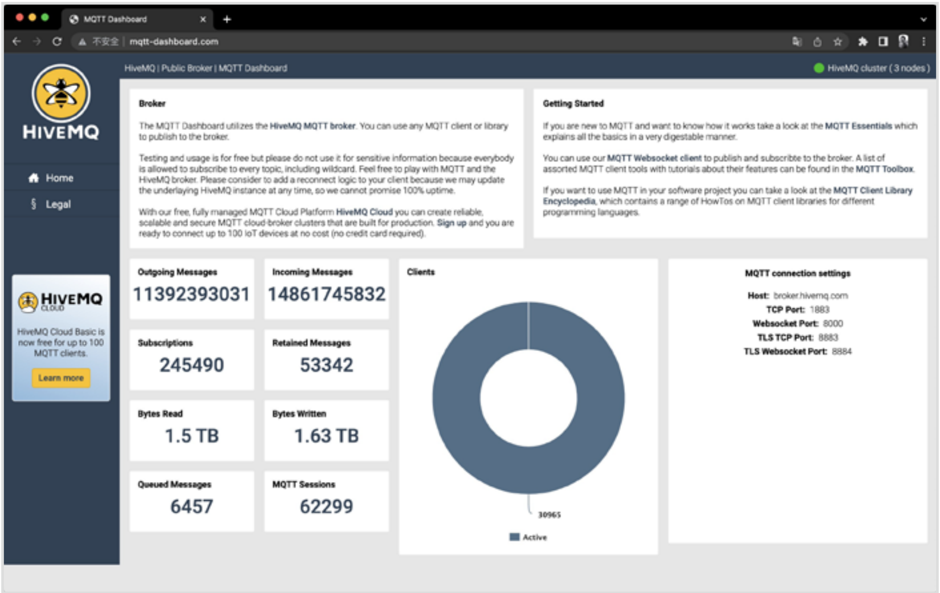
204417_12359124
Full Access

0 errors/s

read-timeout
arduino 15:08:46

HiveMQ

HiveMQ is a cloud-based MQTT platform, offering scalable, secure, and reliable IoT communication services. HiveMQ can help enterprises and developers quickly build and manage IoT applications, supporting millions of devices and messages.



HiveMQ | Public Broker | MQTT Dashboard

HiveMQ cluster (3 nodes)

Broker

The MQTT Dashboard utilizes the **HiveMQ MQTT broker**. You can use any MQTT client or library to publish to the broker.

Testing and usage is for free but please do not use it for sensitive information because everybody is allowed to subscribe to every topic, including wildcard. Feel free to play with MQTT and the HiveMQ broker. Please consider to add a reconnect logic to your client because we may update the underlaying HiveMQ instance at any time, so we cannot promise 100% uptime.

With our free, fully managed MQTT Cloud Platform **HiveMQ Cloud** you can create reliable, scalable and secure MQTT cloud-broker clusters that are built for production. **Sign up** and you are ready to connect up to 100 IoT devices at no cost (no credit card required).

Getting Started

If you are new to MQTT and want to know how it works take a look at the **MQTT Essentials** which explains all the basics in a very digestable manner.

You can use our **MQTT Websocket client** to publish and subscribte to the broker. A list of assorted MQTT client tools with tutorials about their features can be found in the **MQTT Toolbox**.

If you want to use MQTT in your software project you can take a look at the **MQTT Client Library Encyclopedia**, which contains a range of HowTos on MQTT client libraries for different programming languages.

HiveMQ Cloud Basic is now free for up to 100 MQTT clients.

Outgoing Messages 11392393031	Incoming Messages 14861745832
Subscriptions 245490	Retained Messages 53342
Bytes Read 1.5 TB	Bytes Written 1.63 TB
Queued Messages 6457	MQTT Sessions 62299

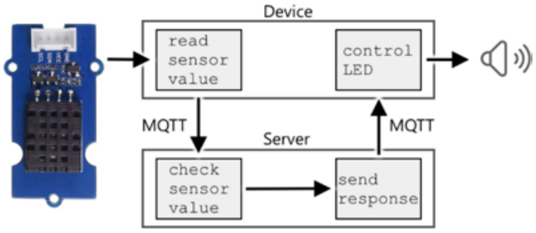
MQTT connection settings

Host: broker.hivemq.com
TCP Port: 1883
Websocket Port: 8000
TLS TCP Port: 8883
TLS Websocket Port: 8884

3.5.2 Task 1: Connect the XIAO ESP32C3 to the MQTT Broker

The first step to adding internet control to your smart temperature and humidity meter is to connect the XIAO ESP32C3 to an MQTT broker.

In this part of the section, you'll connect your smart temperature and humidity meter from Section 2.2 to the internet, enabling it to provide telemetry and be remotely controlled. Later in this section, your device will send a telemetry message via MQTT to a public MQTT broker, which will be received by some server code you'll write. This code will check the temperature and humidity values, and send a command message to the device, telling it to turn a buzzer on or off.



One real-world use of this setup would be in a large indoor space with many temperature and humidity sensors, such as a farm. Before deciding to turn on air conditioning, data can be gathered from multiple temperature and humidity sensors. If only one sensor reading exceeds the threshold, but other sensor readings are normal, this can prevent the entire air conditioning system from being turned on.

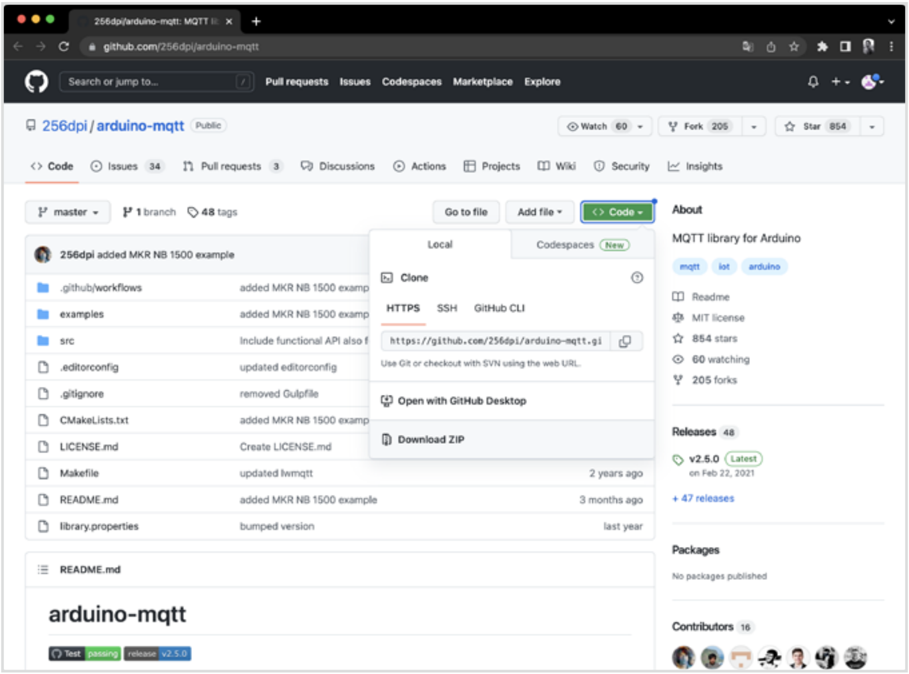
Can you think of other situations where an evaluation of data from multiple sensors is required before issuing a command?

> *Remember, this test broker is public and unsecure, and anyone can listen in on what you're publishing, so it should not be used for any data that needs to be kept confidential.*

Follow the related steps below to connect your device to the MQTT broker we introduced earlier: public.cloud.shiftr.io.

Add the arduino-mqtt library

Before you start programming the XIAO ESP32C3 with the Arduino IDE, you need to add the necessary libraries. Type the library URL https://github.com/256dpi/arduino-mqtt into your browser's address bar to go to the GitHub page. Click on `Code→Download ZIP` to download the resource pack `arduino-mqtt-master.zip` to your local machine, as shown in the image below.



MQTT library for Arduino

https://github.com/256dpi/arduino-mqtt.gi

Use Git or checkout with SVN using the web URL.

256dpi added MKR NB 1500 example

.github/workflows	added MKR NB 1500 examp	
examples	added MKR NB 1500 examp	
src	Include functional API also f	
.editorconfig	updated editorconfig	
.gitignore	removed Gulpfile	
CMakeLists.txt	added MKR NB 1500 examp	
LICENSE.md	Create LICENSE.md	
Makefile	updated lwmqtt	2 years ago
README.md	added MKR NB 1500 example	3 months ago
library.properties	bumped version	last year

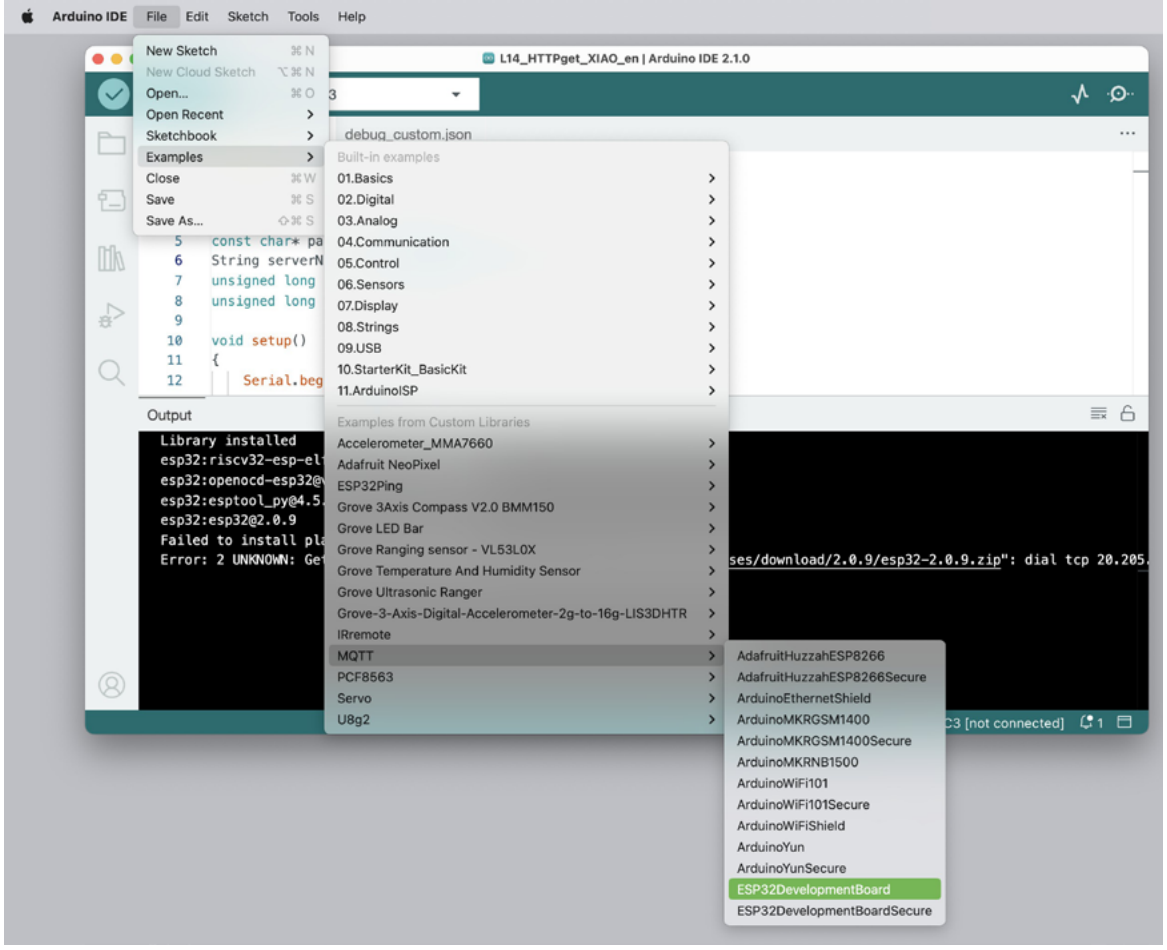
README.md

arduino-mqtt

From the menu bar, select `Sketch→Include Library→Add .ZIP` Library to add the resource pack `arduino-mqtt-master.zip` you downloaded in the previous step. Continue until you see a message indicating successful library loading.

Run the ESP32 MQTT example

After the library is loaded successfully, open the "ESP32DevelopmentBoard" example in the Arduino IDE through the following path: `File→Examples→MQTT→ESP32DevelopmentBoard`, as shown in the image below.



```
const char* pa
String serverN
unsigned long
unsigned long

void setup()
{
    Serial.beg
```

```
Library installed
esp32:riscv32-esp-el
esp32:openocd-esp32@
esp32:esptool_py@4.5.
esp32:esp32@2.0.9
Failed to install pl
Error: 2 UNKNOWN: Ge                    ses/download/2.0.9/esp32-2.0.9.zip": dial tcp 20.205.
```

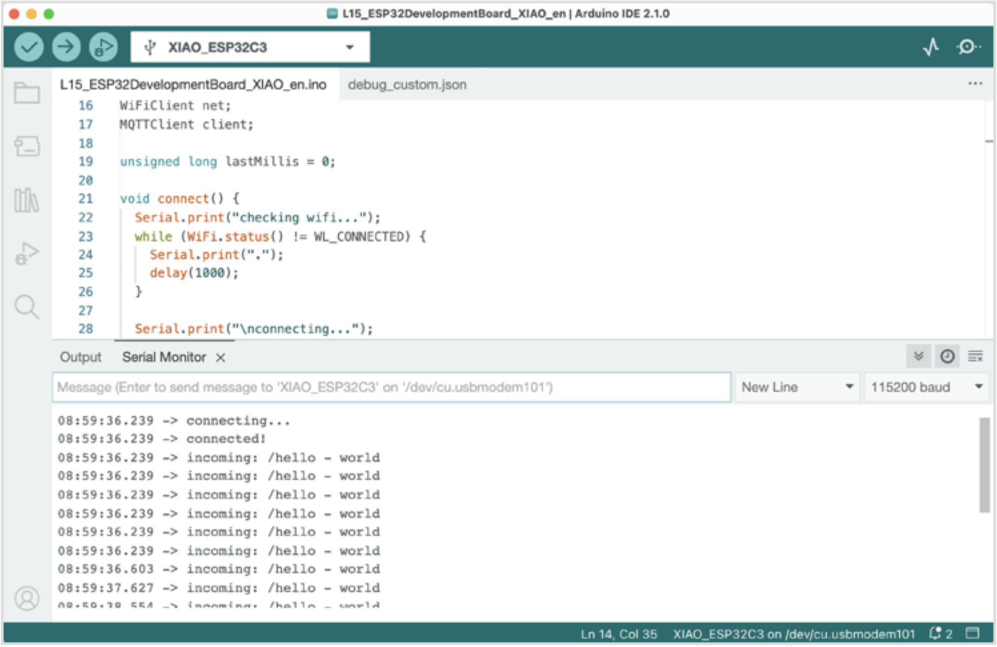
After the example program is opened, you can see the program as shown below. Then change the ssid in the code to your Wi-Fi network name, and change the pass in the code to the corresponding Wi-Fi password for your Wi-Fi network.

```
// This example uses an ESP32 Development Board
// to connect to shiftr.io.
//
// You can check on your device after a successful
// connection here: https://www.shiftr.io/try.
//
// by Joël Gähwiler
// https://github.com/256dpi/arduino-mqtt

#include <WiFi.h>
#include <MQTT.h>
```

```
const char ssid[] = “ssid”;
const char pass[] = “pass”;

WiFiClient net;
MQTTClient client;

unsigned long lastMillis = 0;

void connect() {
  Serial.print(“checking wifi...”);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(“.”);
    delay(1000);
  }

  Serial.print(“\nconnecting...”);
  while (!client.connect(“arduino”, “public”, “public”)) {
    Serial.print(“.”);
    delay(1000);
  }

  Serial.println(“\nconnected!”);

  client.subscribe(“/hello”);
  // client.unsubscribe(“/hello”);
}

void messageReceived(String &topic, String &payload) {
  Serial.println(“incoming: “ + topic + “ – “ + payload);

  // Note: Do not use the client in the callback to publish, subscribe or
  // unsubscribe as it may cause deadlocks when other things arrive while
  // sending and receiving acknowledgments. Instead, change a global variable,
  // or push to a queue and handle it in the loop after calling `client.loop()`.
}

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, pass);

  // Note: Local domain names (e.g. “Computer.local” on OSX) are not supported
  // by Arduino. You need to set the IP address directly.
  client.begin(“public.cloud.shiftr.io”, net);
  client.onMessage(messageReceived);

  connect();
}

void loop() {
  client.loop();
  delay(10);  // <– fixes some issues with WiFi stability

  if (!client.connected()) {
    connect();
  }

  // publish a message roughly every second.
  if (millis() – lastMillis > 1000) {
    lastMillis = millis();
    client.publish("/hello", "world");
  }
}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L15_ESP32DevelopmentBoard_XIAO_en

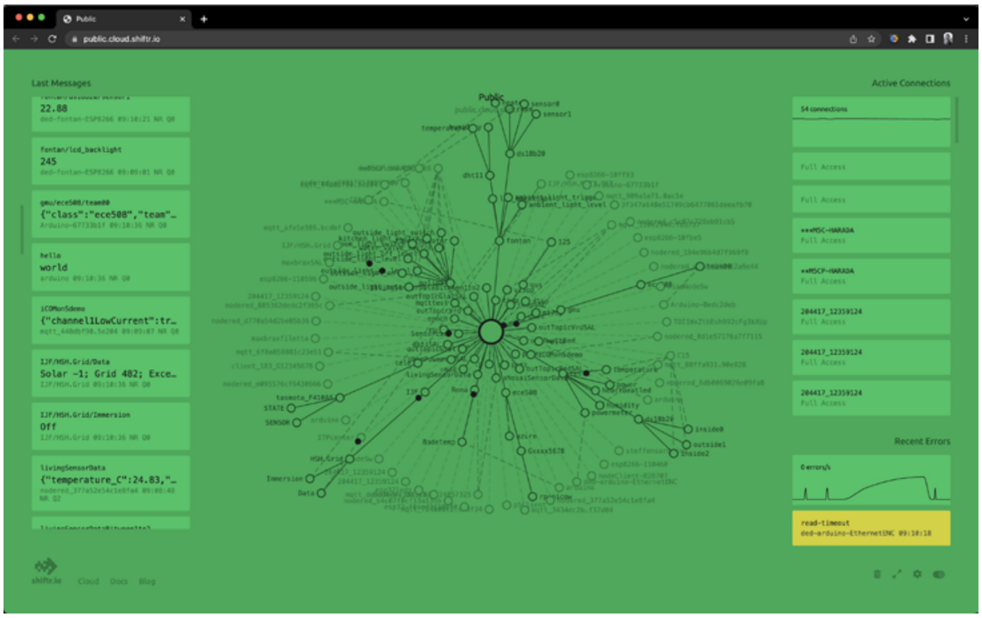
Run the example and check the serial monitor for: `connected!`. If you see a connected client and flowing messages in the live chart, your XIAO is continuously sending data to this public MQTT broker!


L15_ESP32DevelopmentBoard_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L15_ESP32DevelopmentBoard_XIAO_en.ino
debug_custom.json
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')
New Line
115200 baud
Ln 14, Col 35
XIAO_ESP32C3 on /dev/cu.usbmodem101
2

```
WiFiClient net;
MQTTClient client;

unsigned long lastMillis = 0;

void connect() {
  Serial.print("checking wifi...");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }

  Serial.print("\nconnecting...");
```

```
08:59:36.239 -> connecting...
08:59:36.239 -> connected!
08:59:36.239 -> incoming: /hello - world
08:59:36.239 -> incoming: /hello - world
08:59:36.239 -> incoming: /hello - world
08:59:36.239 -> incoming: /hello - world
08:59:36.239 -> incoming: /hello - world
08:59:36.239 -> incoming: /hello - world
08:59:36.603 -> incoming: /hello - world
08:59:37.627 -> incoming: /hello - world
08:59:38.554 -> incoming: /hello - world
```

You can see the messages you sent by accessing public.cloud.shiftr.io in your browser. However, because this is a public broker, your device will quickly get lost in the crowd.



22.88
ded-fontan-ESP8266 09:10:21 NR Q0

fontan/lcd_backlight
245
ded-fontan-ESP8266 09:09:01 NR Q0

gmu/ece508/team00
{"class":"ece508","team"…
Arduino-67733b1f 09:10:36 NR Q0

hello
world
arduino 09:10:36 NR Q0

iCOMonSdemo
{"channel1LowCurrent":tr…
mqtt_448dbf90.5e204 09:09:07 NR Q0

IJF/HSH.Grid/Data
Solar -1; Grid 482; Exce…
IJF/HSH.Grid 09:10:36 NR Q0

IJF/HSH.Grid/Immersion
Off
IJF/HSH.Grid 09:10:36 NR Q0

livingSensorData
{"temperature_C":24.83,"…
nodered_377a52e54c1e8fa4 09:08:48 NR Q2

54 connections

Full Access

Full Access

***MSC-HARADA
Full Access

**MSCP-HARADA
Full Access

204417_12359124
Full Access

204417_12359124
Full Access

204417_12359124
Full Access

0 errors/s

read-timeout
ded-arduino-EthernetENC 09:10:18

> *Keep in mind, this test broker is public and insecure. Anyone can listen to what you're publishing, so it should not be used for anything requiring confidentiality.*

3.5.3 Deep Dive into MQTT

Topics can have a hierarchy, and clients can use wildcards to subscribe to different levels of different hierarchies. For example: you can send temperature telemetry to the `/telemetry/temperature` topic, humidity data to the `/telemetry/humidity` topic, and then subscribe to the `/telemetry/*` topic in your cloud application to receive both temperature and humidity telemetry. When messages are sent, a Quality of Service (QoS) can be specified which determines the guarantee of message delivery.

- At most once: The message is sent only once, and no additional steps are taken by the client and the broker to confirm delivery (Fire and Forget).
- At least once: The message is retried by the sender until it receives an acknowledgment (Acknowledged delivery).
- Exactly once: A two-level handshake is performed by the sender and receiver to ensure that only one copy of the message is received (Assured delivery).

In what scenarios might you need to deliver messages on a Fire and Forget basis?

Although MQTT (Message Queuing Telemetry Transport) has "Message Queuing" in its name (the first two letters of MQTT), it does not actually support message queues. This means that if a client disconnects and then reconnects, it will not receive messages that were sent while it was disconnected, except for those messages that it had already begun processing using the QoS process. A retain flag can be set on a message. If this flag is set, the MQTT broker will store the last message sent on a topic with this flag, and will send it to any clients who subsequently subscribe to that topic. This way, clients always receive the latest message.

MQTT also supports a keep-alive feature to check if the connection is still online during long intervals between messages.

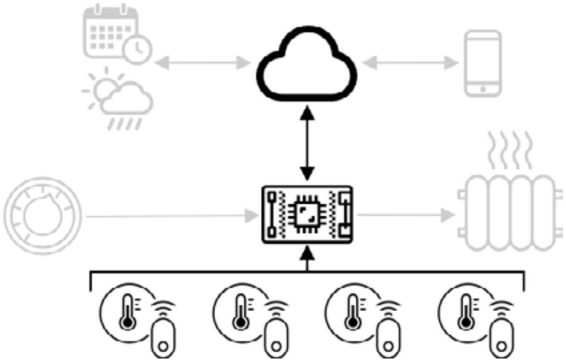
MQTT connections can be public, or encrypted and protected using usernames, passwords, or certificates.

MQTT communicates over TCP/IP, the same underlying network protocol as HTTP, but on a different port. You can also communicate with web applications running in a browser over MQTT on websockets, or in situations where firewalls or other network rules block standard MQTT connections.

3.5.4 Telemetry

The word "telemetry" comes from Greek roots meaning "remote measurement". Telemetry refers to the act of collecting data from sensors and sending it to the cloud.

One of the earliest telemetry devices was invented in France in 1874, sending real-time weather and snow depth data from Mont Blanc to Paris. As there was no wireless technology at the time, it used a physical wire.



Let's go back to the smart thermostat example from Section 1.1.

The thermostat has temperature sensors to collect telemetry data. It likely has a built-in temperature sensor and may connect to multiple external temperature sensors via wireless protocols such as Low Energy Bluetooth (BLE).

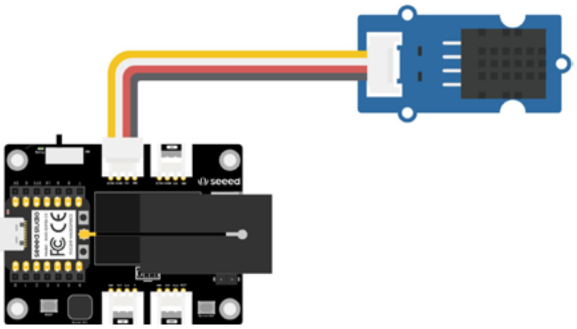
An example of the telemetry data it sends could be:

Name	Value	Description
AC_Temperature	18°C	The temperature measured by the thermostat's built-in temperature sensor
Living_Room_ Temperature	19°C	The temperature measured by a remote temperature sensor named livingroom , indicating the room it is in
Bedroom_Temperature	21°C	The temperature measured by a remote temperature sensor named bedroom , indicating the room it is in

Then, the cloud service can use this telemetry data to decide what commands to send to control cooling or heating.

3.5.5 Task 2: Sending Telemetry Information from XIAO to MQTT Broker

The next part of adding internet control to your smart hygrothermograph is sending the temperature and humidity telemetry data to the telemetry topic of the MQTT broker. Replace the XIAO of your smart hygrothermograph device from Section 2.2 with the XIAO ESP32C3, as shown in the image below.



Load the following program into the Arduino IDE to test sending telemetry data from your device to the MQTT broker. Note that in this example, we're trying a different MQTT broker than in Task 1: `broker.hivemq.com`, and we've set `XIAO_ESP32C3_Telemetry/` as the subscription name.

```
////////////////////////////////////////////////////////////////////////////////
// IDE:
//   Arduino 2.0.0
// Platform:
//   esp32 2.0.5 - https://github.com/espressif/arduino-esp32
// Board:
//   XIAO_ESP32C3
// Libraries:
//   MQTT 2.5.0 - https://github.com/knolleary/pubsubclient
//   ArduinoJson 6.19.4 - https://github.com/bblanchon/ArduinoJson

////////////////////////////////////////////////////////////////////////////////
```

```
// Includes

#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include “DHT.h”
#define DHTTYPE DHT20
DHT dht(DHTTYPE);

const char* ssid = “ssid”;
const char* password = “pass”;

const char* mqtt_server = “broker.
hivemq.com”;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

float temperature = 0;
float humidity = 0;

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  Wire.begin();
  dht.begin();
}

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi
network
  Serial.println();
  Serial.print(“Connecting to “);
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECT-
ED) {
    delay(500);
    Serial.print(“.”);
  }

  Serial.println(“”);
  Serial.println(“WiFi connected”);
  Serial.println(“IP address: “);
  Serial.println(WiFi.localIP());
}

void reconnect() {
  // Loop until we’re reconnected
  while (!client.connected()) {
    Serial.print(“Attempting MQTT con-
nection...”);
    // Attempt to connect
    if (client.connect(“XIAO_ESP32”)) {
      Serial.println(“connected”);
      // Subscribe
        client.subscribe(“XIAO_ESP32/
LEDOUTPUT”);
    } else {
      Serial.print(“failed, rc=”);
      Serial.print(client.state());
       Serial.println(“ try again in 5
seconds”);
       // Wait 5 seconds before retry-
ing
      delay(5000);
    }
  }
}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  float temp_hum_val[2] = {0};
  if (now - lastMsg > 5000) {
    lastMsg = now;

     dht.readTempAndHumidity(temp_hum_
val);
    temperature = temp_hum_val[1];

    char tempString[8];
     dtostrf(temperature, 1, 2, temp-
String);
    Serial.print(“Temperature: “);
    Serial.println(tempString);
     client.publish(“XIAO_ESP32C3_Te-
lemetry/Temperaturedataread”, temp-
String);

    humidity = temp_hum_val[0];

    char humString[8];
     dtostrf(humidity, 1, 2, hum-
String);
    Serial.print(“Humidity: “);
    Serial.println(humString);
     client.publish(“XIAO_ESP32_Teleme-
try/Humiditydataread”, humString);
  }

}
```

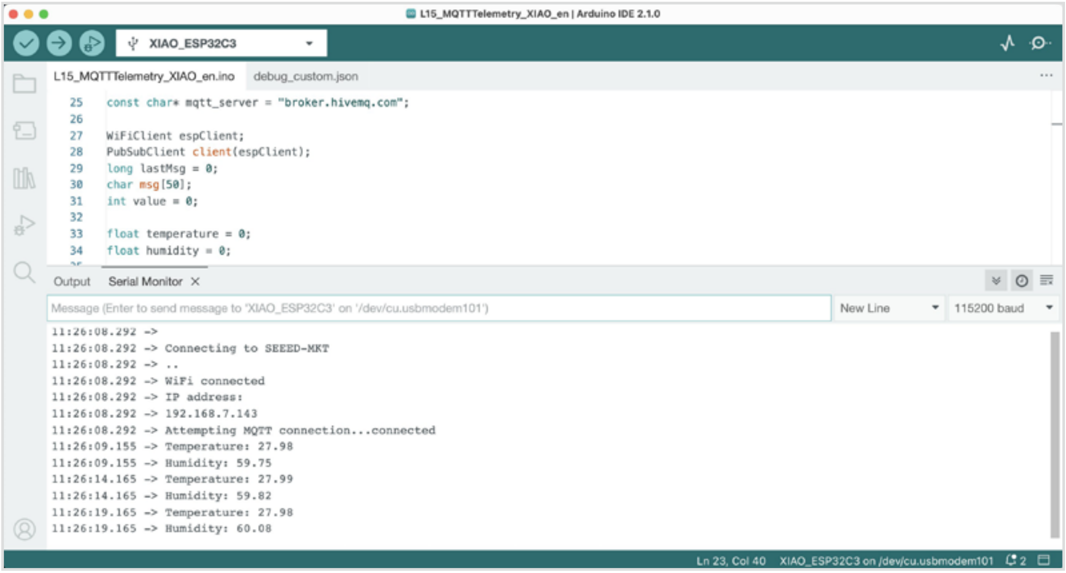
Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L15_MQTTTelemetry_XIAO_en

Because this example relies on the `PubSubClient.h` library, if you try to compile it directly, you will encounter the error "**PubSubClient.h: No such file or directory**". To resolve this issue, follow the steps below to install the library.

1. Open the Arduino IDE.
2. Go to "Sketch" -> "Include Library" -> "Manage Libraries".
3. In the Library Manager, type "PubSubClient" in the search bar.
4. Look for the "PubSubClient" library by Nick O'Leary and click on it.
5. Click the "Install" button to install the library.

Then modify the `ssid` in the code to your Wi-Fi network name and `pass` to your Wi-Fi password corresponding to your Wi-Fi network name.

After successfully uploading the program, open the serial monitor. If all goes well, you will see the device start sending temperature and humidity data, as shown in the image below.

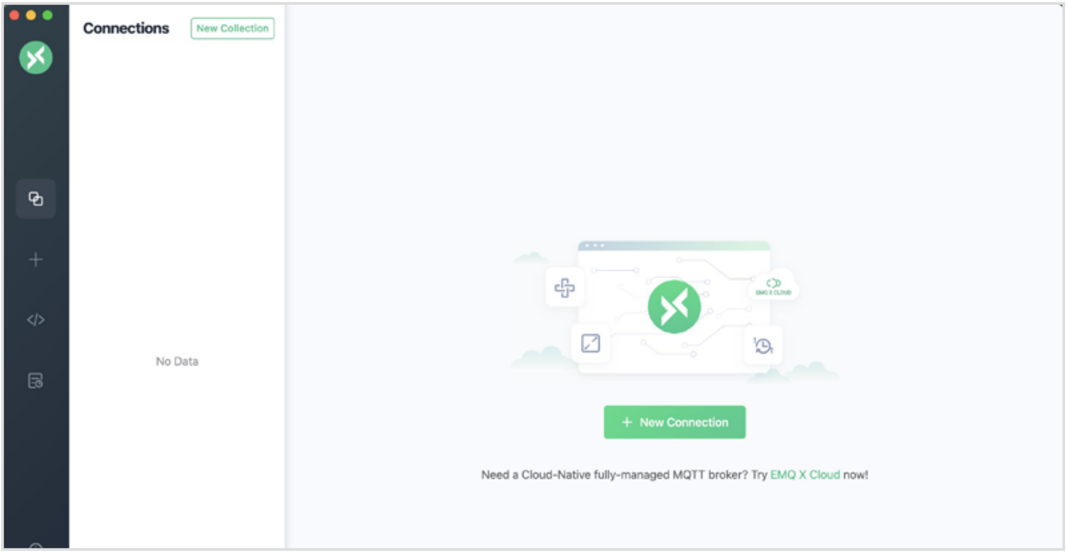

L15_MQTTTelemetry_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L15_MQTTTelemetry_XIAO_en.ino
debug_custom.json

```
const char* mqtt_server = "broker.hivemq.com";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

float temperature = 0;
float humidity = 0;
```

Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem101')
New Line
115200 baud

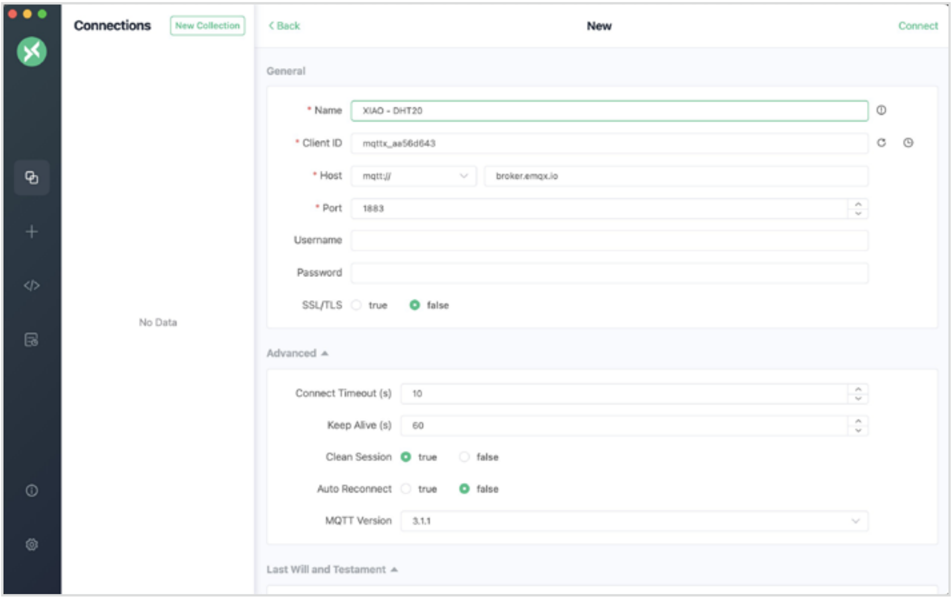
```
11:26:08.292 ->
11:26:08.292 -> Connecting to SEEED-MKT
11:26:08.292 -> ..
11:26:08.292 -> WiFi connected
11:26:08.292 -> IP address:
11:26:08.292 -> 192.168.7.143
11:26:08.292 -> Attempting MQTT connection...connected
11:26:09.155 -> Temperature: 27.98
11:26:09.155 -> Humidity: 59.75
11:26:14.165 -> Temperature: 27.99
11:26:14.165 -> Humidity: 59.82
11:26:19.165 -> Temperature: 27.98
11:26:19.165 -> Humidity: 60.08
```

How can you see the sensor data from another platform? There are many ways, such as MQTT X. After downloading and installing the software suitable for your PC system, the interface is as shown in the image below.



Need a Cloud-Native fully-managed MQTT broker? Try EMQ X Cloud now!

Clicking the `+ New Connection` button will bring you to the connection creation window, as shown in the image below. Fill in `XIAO-DHT20` in the Name box as the connection name. The Host is `broker.hivemq.com` that we set in the program, no other settings are needed, click `Connect` in the upper right corner.



General

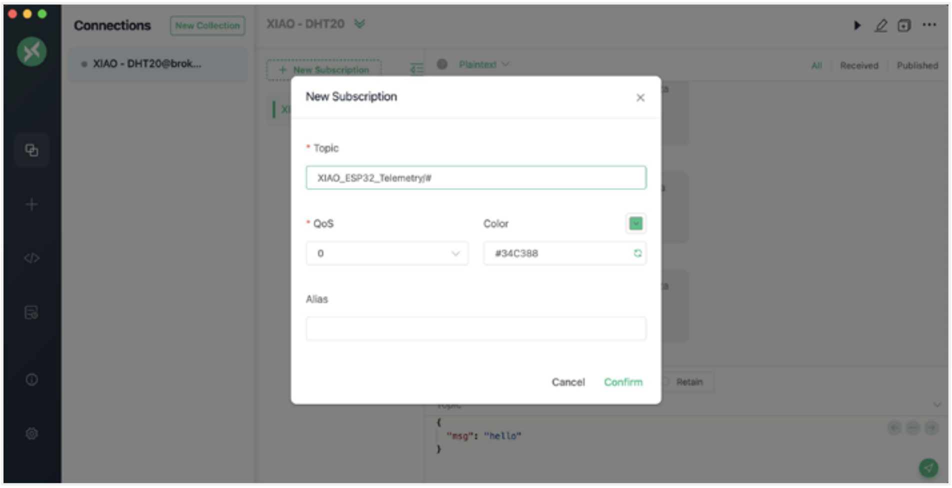
- * Name: XIAO - DHT20
- * Client ID: mqttx_aa56d643
- * Host: mqtt:// broker.emqx.io
- * Port: 1883
- Username:
- Password:
- SSL/TLS: ○ true ● false

Advanced ▲

- Connect Timeout (s): 10
- Keep Alive (s): 60
- Clean Session: ● true ○ false
- Auto Reconnect: ○ true ● false
- MQTT Version: 3.1.1

Last Will and Testament ▲

Create a new subscription, showing all the information under `XIAO_ESP32C3_Telemetry/`, as shown in the image below.



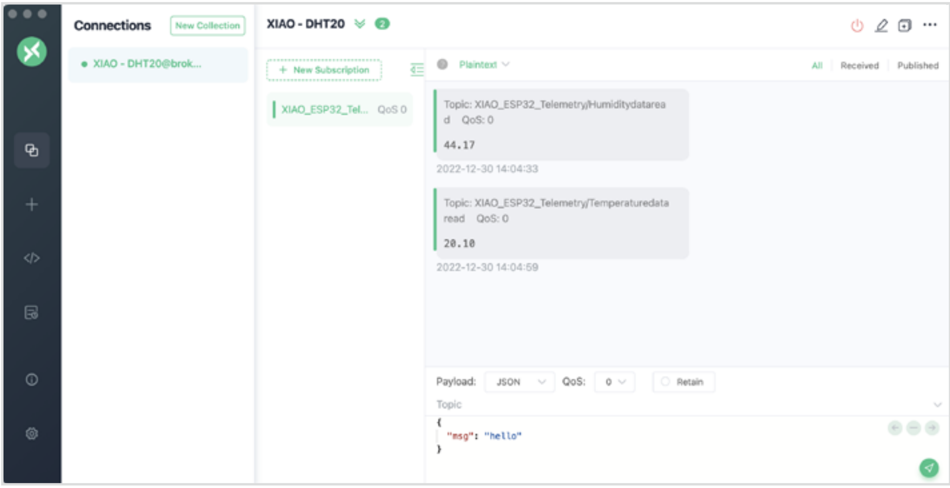
New Subscription

- * Topic: XIAO_ESP32_Telemetry/#
- * QoS: 0
- Color: #34C388
- Alias:

Cancel Confirm

```
{
  "msg": "hello"
}
```

Now, we can see the telemetry data sent from XIAO ESP32C3, as shown in the image below.



Topic: XIAO_ESP32_Telemetry/Humiditydataread QoS: 0

44.17

2022-12-30 14:04:33

Topic: XIAO_ESP32_Telemetry/Temperaturedataread QoS: 0

20.10

2022-12-30 14:04:59

Payload: JSON QoS: 0 ○ Retain

Topic

```
{
  "msg": "hello"
}
```

How often should telemetry be sent?

One question that needs careful consideration with telemetry is: how often should you measure and send data? The answer is — it depends on the needs of the device being monitored and the task at hand. If you measure frequently, you can indeed respond to changes in the measurements more quickly, but this would cause your device to consume more power, more bandwidth, generate more data, and require more cloud resources to handle. You need to strike a balance between measuring often enough but not too often.

For a thermostat, measuring every few minutes might be enough because the temperature isn't likely to change frequently. If you only measure once a day, then you might be heating your house for nighttime temperatures on a sunny day, and if you measure every second, you'd have thousands of unnecessary repeated temperature measurements which will eat up users' internet speed and bandwidth (which is a problem for people with limited bandwidth plans), and also consume more power, which is a problem for devices like remote sensors that rely on battery power, and further increase the cost of cloud computing resources to process and store them.

If you're monitoring data around a machine in a factory that might cause catastrophic damage and millions in lost revenue if it fails, then measuring multiple times a second may be necessary. Wasting bandwidth is better than missing telemetry data that could signal the need to stop and repair before a machine fails.

> *In this situation, you could consider first using an edge device to handle the telemetry data to reduce dependence on the internet.*

Losing connection

Internet connections can be unreliable, and it's common to lose signal. In this case, what should an IoT device do? Should it lose data, or should it store data until the connection is restored? Again, the answer is — it depends on the device being monitored.

For a thermostat, data is likely lost once a new temperature measurement has been made. If the current temperature is 19°C, the heating system doesn't care that the temperature 20 minutes ago was 20.5°C; it's the current temperature that dictates whether the heat should be turned on or off.

For some machines, you may want to retain this data, especially if it's being used to look for trends. Some machine learning models can identify anomalies in data streams by looking at a defined time period (e.g., the last hour). This is often used for predictive maintenance, looking for signs that something might be about to fail so you can repair or replace it before disaster strikes. You may want every point of telemetry from a machine sent so it can be used for anomaly detection, so once an IoT device can reconnect, it will send all the telemetry data generated during the internet outage.

IoT device designers should also consider whether an IoT device can operate during an internet outage or if it loses signal due to location. If a smart thermostat is unable to send telemetry data to the cloud due to an internet outage, it should be able to make some limited decisions to control heating.

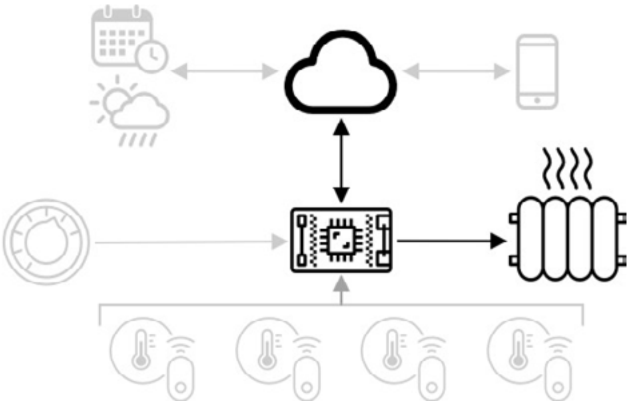
> *This Ferrari became a brick when someone tried to update it in an underground car park... but there was no cell signal there.*

For MQTT handling connection interruptions, if necessary, the device and server code will need to be responsible for ensuring message delivery, for example, requiring all sent messages to be replied to by an additional message on the reply topic, and if not, to manually queue them for later resending.

3.5.6 Commands

Commands are messages sent by the cloud to a device instructing it to do something. Most often, this involves providing some output via an actuator, but it could be an instruction to the device itself, such as to reboot, or to collect additional telemetry data and send it as a response to the command.

A thermostat could receive a command from the cloud to turn on the heat. Based on the telemetry data from all sensors, if the cloud service has decided that the heat should be turned on, then it sends the appropriate command.



3.5.7 Task 3: Send Commands to XIAO via MQTT Broker

Having mastered telemetry, the next step is to send commands to IoT devices via an MQTT broker. In this task, we will try to use a computer with MQTT broker, often called a host computer, to send specific characters and let the Wi-Fi connected XIAO ESP32C3 control a buzzer attached to an expansion board to emit a warning sound.

In the Arduino IDE, load the following program to test sending specific characters (first character is '0') from the MQTT broker to activate the buzzer. We use the MQTT broker: `broker.hivemq.com` in this example.

```
////////////////////////////////////////////////////////////////////////////////
// IDE:
//   Arduino 2.0.0
// Platform:
//   esp32 2.0.5 - https://github.com/espressif/arduino-esp32
// Board:
//   XIAO_ESP32C3
// Libraries:
//   MQTT 2.5.0 - https://github.com/knolleary/pubsubclient
//   ArduinoJson 6.19.4 - https://github.com/bblanchon/ArduinoJson
//  https://github.com/Seeed-Studio/Seeed_Arduino_MultiGas


////////////////////////////////////////////////////////////////////////////////
// Includes


#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
```

```
const char* ssid = “ssid”;
const char* password = “pass”;

const char* mqtt_server = “broker.hivemq.com”;

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

int speakerPin = A3;

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print(“Connecting to “);
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(“.”);
  }

  Serial.println(“”);
  Serial.println(“WiFi connected”);
  Serial.println(“IP address: “);
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print(“Message arrived [“);
  Serial.print(topic);
  Serial.print(“] “);
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  if((char)payload[0]==’0’){
    Serial.print(“  RUN”);
    digitalWrite(speakerPin, HIGH);
    delay(2000);
    digitalWrite(speakerPin, LOW);
    delay(100);
  }
  Serial.println();
}

void setup() {
  Serial.begin(115200);
  pinMode(speakerPin, OUTPUT);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.subscribe(“XIAO_ESP32/Recieve”);
  client.setCallback(callback);
}

void reconnect() {
  // Loop until we’re reconnected
  while (!client.connected()) {
    Serial.print(“Attempting MQTT connection...”);
    // Attempt to connect
    if (client.connect(“XIAO_ESP32”)) {
      Serial.println(“connected”);
      // Subscribe

    } else {
      Serial.print(“failed, rc=”);
      Serial.print(client.state());
      Serial.println(“ try again in 5 seconds”);
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void loop() {

  if (!client.connected()) {
    reconnect();
    client.subscribe(“XIAO_ESP32/Recieve”);
  }
  client.loop();

}
```

Get this program from Github https://github.com/mouseart/XIAO-Mastering-Arduino-and-TinyML/tree/main/code/L15_MQTTCommand_XIAO_en

Then modify the `ssid` in the code to your Wi-Fi network name, and modify the `pass` in the code to the Wi-Fi password corresponding to your Wi-Fi network name.

The logic of the program execution is explained as follows:

```
client.setServer(mqtt_server, 1883);
client.subscribe(“XIAO_ESP32/Recieve”);
client.setCallback(callback);
```

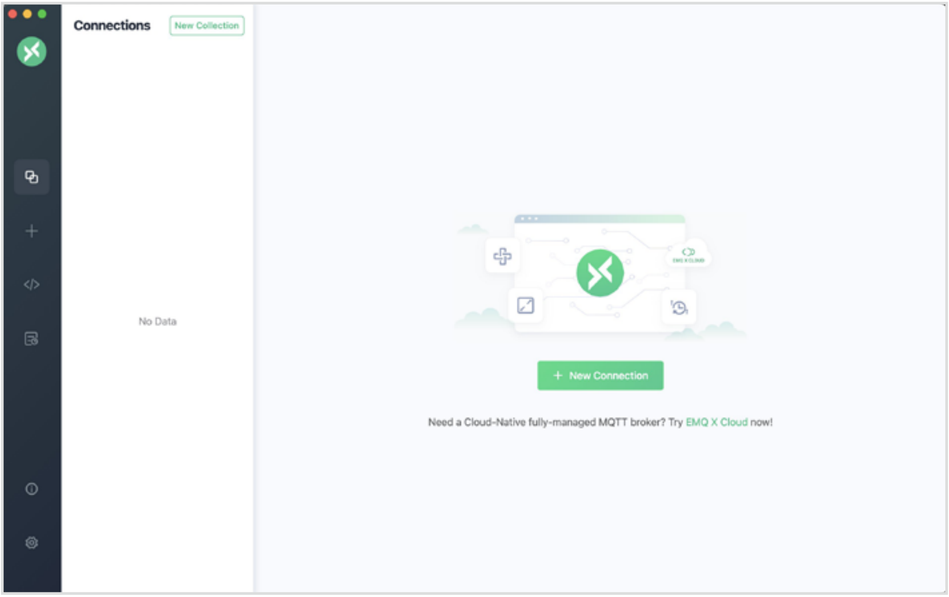
During the `setup` stage, the connection between XIAO and the MQTT server is initialized, and the topic subscription settings and callback functions are set. Here we subscribe to the topic `XIAO_ESP32/Recieve` as an example. When we send a message to this topic from the host computer, the corresponding callback function `callback` will be executed:

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print(“Message arrived [“);
  Serial.print(topic);
  Serial.print(“] “);
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  if((char)payload[0]==’0’){
    Serial.print(“  RUN”);
    digitalWrite(speakerPin, HIGH);
    delay(2000);
    digitalWrite(speakerPin, LOW);
    delay(100);
  }
  Serial.println();
}
```

Here it will first print out the received message, then extract the character at position `0`. When the character at position `0`, which is the first character, is `0`, it satisfies the condition for the `if` statement to perform an action. Here we connect the XIAO ESP32C3 and the expansion board together. When the condition is met, the buzzer on the expansion board will change its level briefly and beep for 2 seconds, while sending the prompt message `RUN` to the serial port.

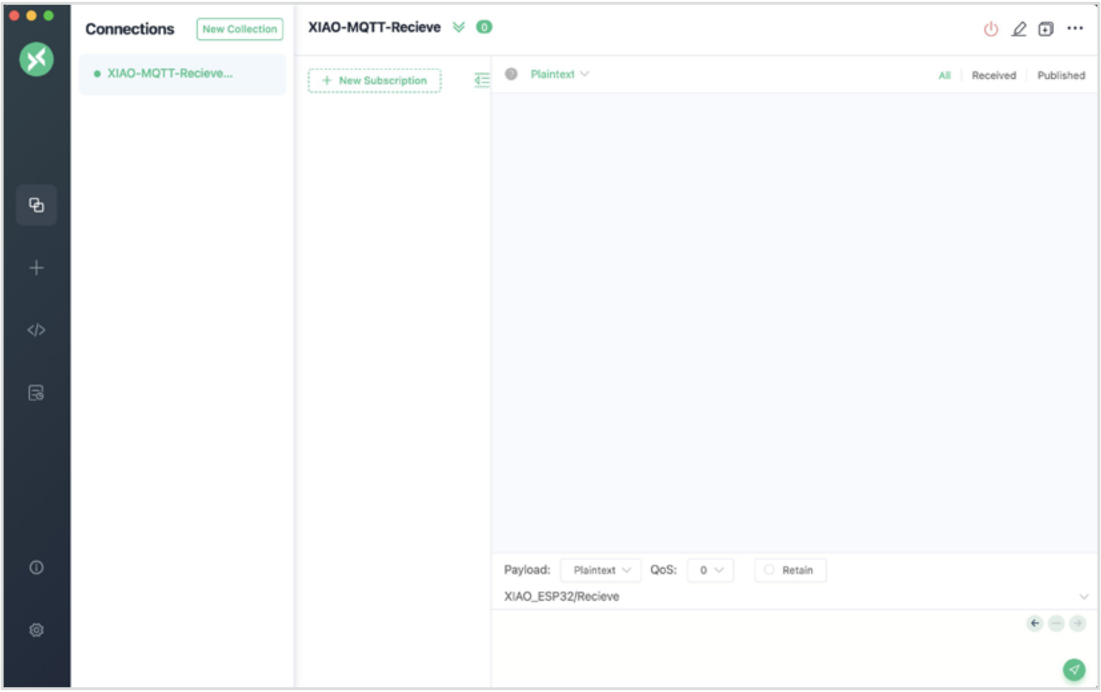
In the process of development and testing by readers, you can also try to integrate the receive and send functions of MQTT, and send messages to specific topics in the callback function, so that the sender can ensure that XIAO has received the message.

On the host computer, we use MQTT X to test. Open MQTT X, the interface is as shown in the following figure.



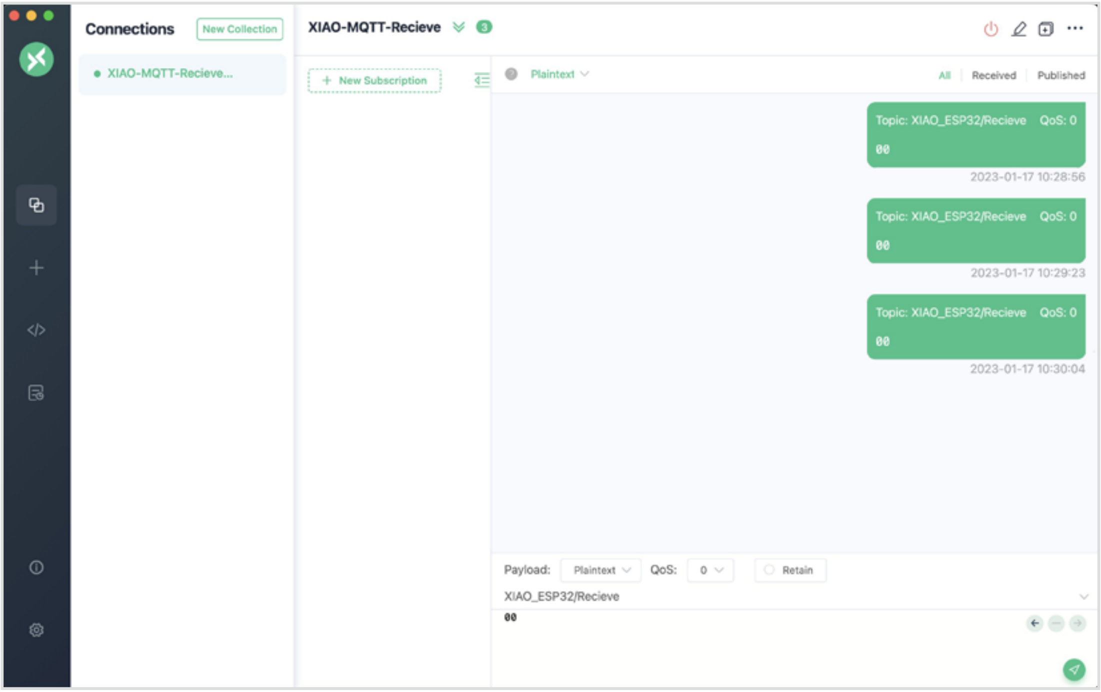
Need a Cloud-Native fully-managed MQTT broker? Try EMQ X Cloud now!

Click the `+ New Connection` button to enter the connection creation window, as shown in the following figure. Fill in the Name box with `XIAO-MQTT-Recieve` as the connection name. Host is the `broker.hivemq.com` we set in the program, and nothing else needs to be set. Click `Connect` at the top right corner. The interface after successful connection is as shown in the following figure.



XIAO_ESP32/Recieve

Now we can publish messages to the specified topic, which is the topic `XIAO_ESP32/Recieve` we subscribed to on XIAO. Then we enter `00` in the input box of `XIAO_ESP32/Recieve` at the lower right corner of the interface, and then click the send button in the lower right corner.



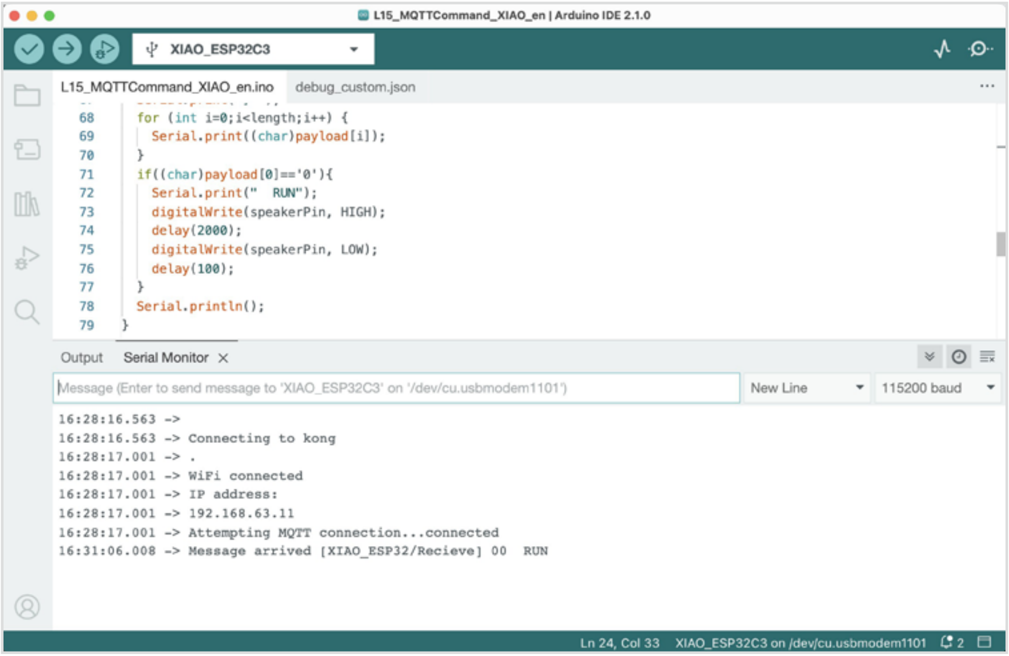
Topic: XIAO_ESP32/Recieve QoS: 0
00
2023-01-17 10:28:56

Topic: XIAO_ESP32/Recieve QoS: 0
00
2023-01-17 10:29:23

Topic: XIAO_ESP32/Recieve QoS: 0
00
2023-01-17 10:30:04

XIAO_ESP32/Recieve
00

At this time, in the serial monitor on the PC side, you can see the prompt message received from XIAO, as shown in the following figure, and prompt `RUN`, the buzzer will sound for 2 seconds, indicating that the message has been received.


L15_MQTTCommand_XIAO_en | Arduino IDE 2.1.0
XIAO_ESP32C3
L15_MQTTCommand_XIAO_en.ino
debug_custom.json
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32C3' on '/dev/cu.usbmodem1101')
New Line
115200 baud
Ln 24, Col 33
XIAO_ESP32C3 on /dev/cu.usbmodem1101
2

```
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  if((char)payload[0]=='0'){
    Serial.print("  RUN");
    digitalWrite(speakerPin, HIGH);
    delay(2000);
    digitalWrite(speakerPin, LOW);
    delay(100);
  }
  Serial.println();
}
```

```
16:28:16.563 ->
16:28:16.563 -> Connecting to kong
16:28:17.001 -> .
16:28:17.001 -> WiFi connected
16:28:17.001 -> IP address:
16:28:17.001 -> 192.168.63.11
16:28:17.001 -> Attempting MQTT connection...connected
16:31:06.008 -> Message arrived [XIAO_ESP32/Recieve] 00  RUN
```

Now, we have successfully driven the buzzer on the expansion board connected to the Wi-Fi connected XIAO ESP32C3 through the instruction sent by the PC side.

The action of the buzzer can be replaced with the control of any peripheral to achieve the desired function.

Lost connection

If a cloud service needs to send a command to an offline IoT device, what should it do? Again, the answer depends on the situation. If the latest command overwrites the previous one, the previous command may be ignored. If the cloud service sends a command to turn on the heating, and then sends another command to turn off the heating, then the turn-on command can be ignored and does not need to be resent.

If the commands need to be processed in order, such as first moving the robot arm up and then closing the gripper, then they need to be sent in order once the connection is restored.

How can device or server code ensure that commands are always sent and processed in order through MQTT if needed?

Using XIAO's Bluetooth function

XIAO nRF52840, XIAO nRF52840 Sense, XIAO ESP32C3 all support Bluetooth function, you can refer to the related Wiki documents to learn how to use the Bluetooth function.

- *Bluetooth Usage on Seeed Studio XIAO ESP32C3*
- *Bluetooth Usage (Seeed nRF52 Boards Library)*
- *Bluetooth Usage (Seeed nrf52 mbed-enabled Boards Library)*

Chapter 4:
Project Practice Advanced - TinyML Applications

Among the XIAO series products, the Seeed Studio XIAO nRF52840 Sense has Bluetooth 5.0 wireless connectivity, low power consumption, and onboard 6-axis IMU and PDM microphone sensors. Besides, the XIAO ESP32S3 Sense further integrates PSRAM, a camera, a digital microphone, and SD card support.

> *Those characteristics make those devices powerful tools for TinyML (Tiny Machine Learning) projects.*

TinyML solves problems in a completely different way from traditional programming methods. This chapter will introduce you to this cutting-edge field by walking through the entire TinyML workflow, from data collection, pre-processing, model definition, training, testing, and deployment to allow actual inference on the physical world.

4.1 Understanding TinyML and Edge Impulse Studio

This section will explain embedded machine learning, the differences between TinyML and other artificial intelligence, and some essential applications. This section will help you understand what TinyML is and why we need it. Edge Impulse Studio is one of the tools that allows developers to create next-generation intelligent device solutions through embedded machine learning. This section will introduce you to this tool and help you understand the basic steps to build an embedded machine-learning model.

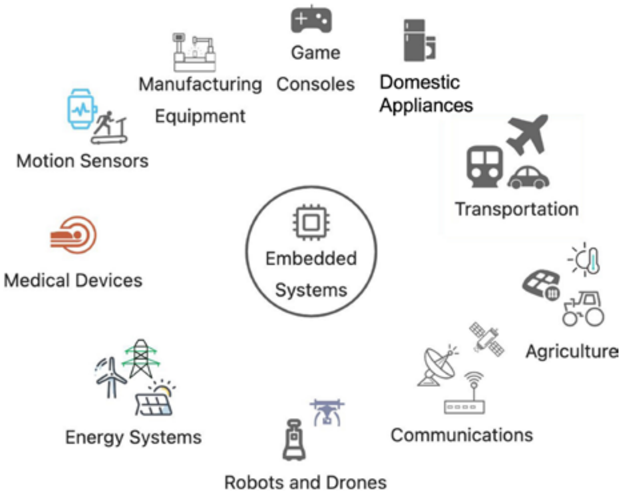
4.1.1 Common Terms

In addition to TinyML, we often hear conceptually similar terms such as edge computing, edge AI, embedded machine learning, etc. So, before learning TinyML, you need to understand these terms and their meanings.

Embedded Systems

An embedded system is a computer used only to solve a few particular problems and is challenging to change. The term "embedded" means that it is built into the system. It is a permanent part of the more extensive system. It usually doesn't look like a computer; in most cases, it doesn't have a keyboard, monitor, or mouse. But like any computer, it has a processor, software, input, and output.

Embedded systems are computers controlling various physical and electronic devices, and now they are almost everywhere. From the Bluetooth headphones you use, home audio-visual equipment, game consoles, air conditioners, sweeping robots, rice cookers, and washing machines to the control units of electric vehicles to communication equipment, factory equipment, medical equipment, office places, and even military equipment, almost any electrically driven device has the presence of embedded systems. Embedded software is the software running on them, and the following figures show some scenes where you can see embedded systems.



Embedded systems can be large or small, as small as the microcontroller controlling the digital watch and as large as the embedded computer in the autonomous car. Unlike general-purpose computers such as laptops or smartphones, embedded systems usually perform a specific specialized task.

- Do some research -

Look around you, what devices might have embedded systems in them?

The size and complexity of embedded systems vary, but they all contain three basic components:

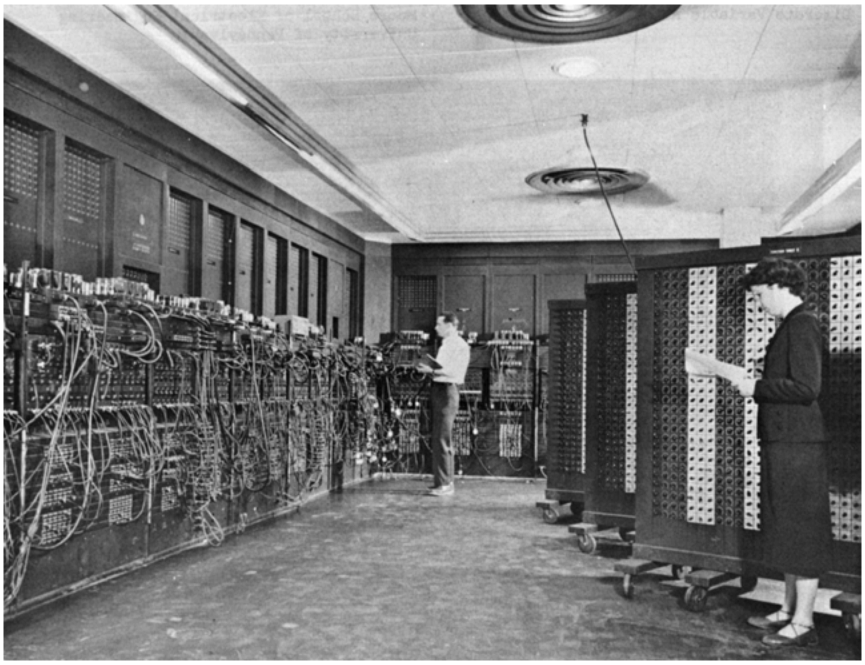
- **Hardware:** These systems use microprocessors and microcontrollers as their hardware. Microprocessors are similar to microcontrollers because they are both related to CPUs (central processing units), and CPUs are combined with other essential computer parts (such as storage chips and digital signal processors (DSPs)). Microcontrollers integrate all these parts into a single chip.
- **Firmware and Software:** The complexity of the system software differs from industrial-grade microcontrollers and embedded IoT systems, on the other hand, they usually run relatively basic software, using very little memory.
- **Real-Time Operating System (RTOS):** These systems, especially the more minor, often do not include these operation systems. By supervising software during program execution and establishing standards, RTOS determines how the system operates.

Embedded systems are often also constrained by their deployment environment. For example, many embedded systems need to run on battery power. Hence, their design needs to consider energy efficiency metrics - perhaps memory is limited, or the clock speed is extremely slow.

The challenge for engineers programming for embedded systems is often implementing functional requirements within these limited hardware and environmental resource constraints. You must consider hardware resource constraints when learning to build your TinyML project model for XIAO later.

Edge Computing and Internet of Things (IoT)

The concept of "Edge" is relative to the "Center." In the early days, computers like the ENIAC (Electronic Numerical Integrator and Computer), built in 1945, were massive, weighing nearly 30 tons and occupying 170 square meters.



Staff programming the ENIAC

At this stage, computational tasks were centralized on the core machine. These large computers evolved into "minicomputers," typically consisting of a central host and multiple terminals connected to the host. Multiple users could issue computational instructions through the terminals, but most of the computation still occurred on the central host. As time passed, the terminals became more complex and took over more and more functions of the central computer.

It wasn't until the advent of personal computers that computation truly expanded to the "edge." The rapid development of personal computers also led to the decline of those massive machines, and the scale of human computation quickly tilted towards the "edge."

The emergence and development of the Internet led to a large concentration of servers to provide a variety of data storage and computational services, search engines, streaming video, online games, social networks, etc. The highly centralized cloud computing era had arrived, and many internet service providers owned massive data center rooms.

1977-78
digital
pdp11/70
processor
handbook

The processor manual cover for the minicomputer pdp11/70, showing the host and connected terminals.

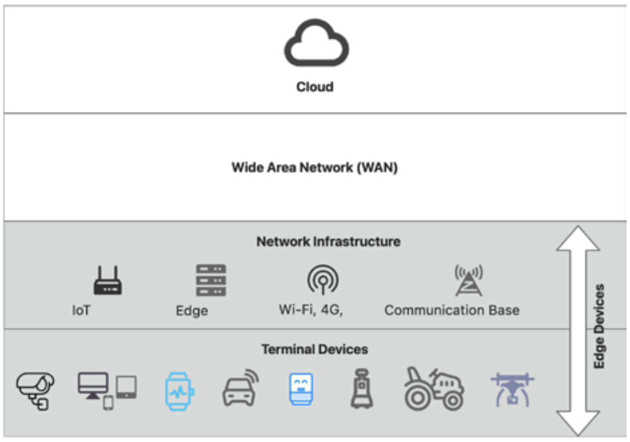
The 70s Wang computer, once a world leader in market share. (Personal computer Wang 2200 PCS II. It is located in the Belgrade Museum of Technology.)

Google's data center located on the outskirts of Pryor, Oklahoma, USA

In theory, all our computing services can be completed in the cloud. But these cloud-based services are useless in many areas without internet connections or when the internet goes down.

The computers we use for work and entertainment are just some devices connected to these cloud services. As of 2021, there were as many as 12.2 billion connected devices worldwide, and we call this network of devices **IoT (Internet of Things)**. It includes everything you can and can't think of, such as mobile phones, smart speakers, connected security cameras, cars, containers, pet trackers, industrial sensors, etc.

These devices are embedded systems containing microprocessors running software written by embedded software engineers. We can also call them edge devices because they are located at the edge of the network. Computation performed on edge devices is known as **Edge Computing**. The following illustration expresses the relationship between the cloud and edge devices.



Devices at the network edge can communicate with the cloud, edge infrastructure, and each other. Edge applications typically span multiple locations on this map. For example, data may be sent from an IoT device equipped with sensors to a local edge server for processing.

Artificial Intelligence (AI)

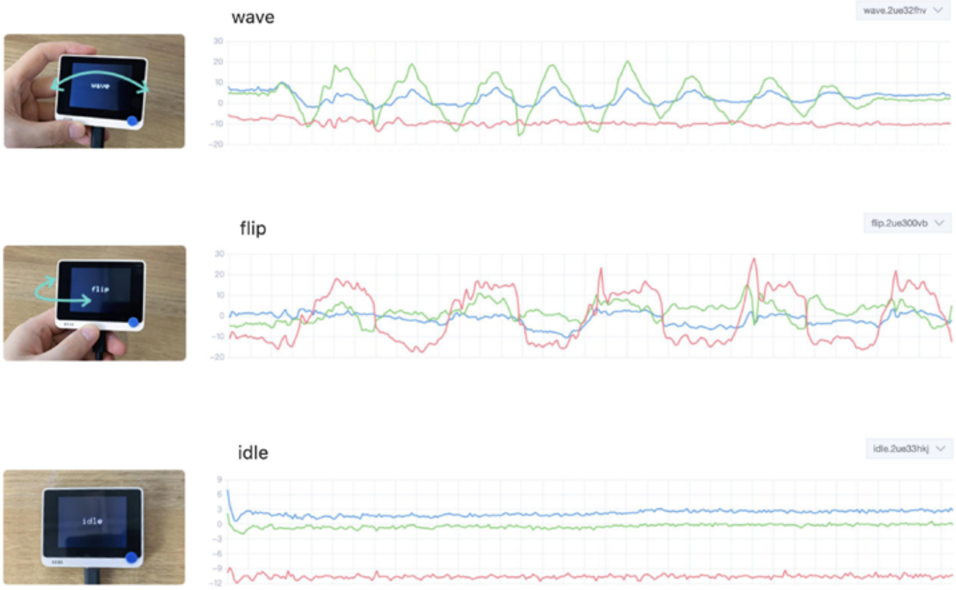
Artificial Intelligence (AI) is a vast concept and is difficult to define. In a vague sense, it's about making things possess human-like intelligence and thinking abilities. But even the definition of intelligence itself is disputed, and this is a cutting-edge field with many unknowns, which readers are welcome to explore on their own. For the convenience of understanding the following concepts, this book provides a relatively narrow definition of AI: an artificial system capable of making wise decisions based on specific inputs.

Machine Learning (ML)

Machine Learning (ML) primarily aims to design and analyze algorithms that allow computers to "learn" automatically. ML algorithms are a class of algorithms that automatically analyze and learn patterns from data and use these patterns to predict unknown data.

Take a typical example of machine learning—continuous motion recognition. In Section 2.4, we learned about the triaxial accelerometer. The image below shows a Wio Terminal with an embedded triaxial accelerometer and screen. We can use it to record accelerometer data for several different movements: waving (wave), flipping (flip), and idling (idle).

Looking at the data generated from these different movements, you can find a method for the



machine to recognize these motion patterns. The traditional way involves manually analyzing and checking the data, identifying specific logical rules for different movements through mathematical analysis, and then writing a recognition program to perform the desired action based on these rules. Sounds complicated, doesn't it?

Fortunately, we now have machine learning methods. Training and testing these data will yield an algorithm, and the device only needs to run this algorithm to automatically complete our desired "inference" process and deliver results. From the current state of machine learning development, this method is exceptionally proficient in handling complex data scenarios. We will learn more about this process in subsequent sections.

Edge AI

As the term suggests, Edge AI combines edge devices and artificial intelligence. The development of Edge AI stems from the pursuit of lower system power consumption and higher efficiency. For example, popular smartwatches or fitness bands often have built-in accelerometers that generate hundreds of readings per second—a large volume of data—and continuous data reading is required to recognize movement states. If the recognition of movements is performed in the cloud, the smartwatch would need to consume a lot of energy to send data to the cloud, and there would usually be a delay in receiving the result. This makes the entire computational process uneconomical—high energy consumption and latency. This latency can also prevent us from effectively using data for real-time feedback.

Edge AI solves this problem by recognizing movements on the smartwatch or band itself. This allows for quick results without relying on the cloud. If necessary data needs to be uploaded to the cloud, there's no need to send a large amount of sensor data; instead, just the essential motion recognition results are sent, significantly reducing communication volume and consuming less electric power.

Embedded Machine Learning

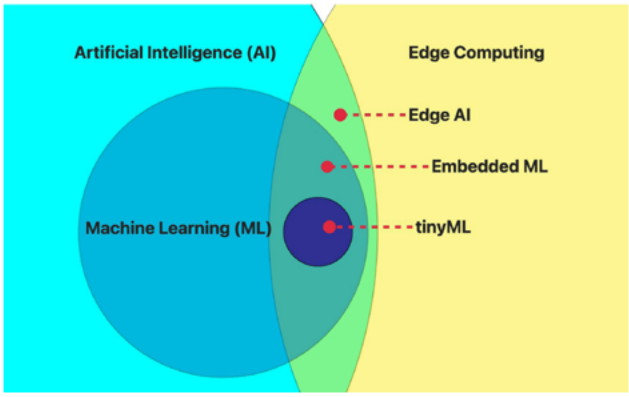
Embedded Machine Learning is the art and science of running machine learning models on embedded systems. When we talk about embedded machine learning, we typically refer to machine learning inference—the process of taking input and making a prediction (for example, guessing motion status based on accelerometer data). The training part is typically still performed on traditional computers.

Furthermore, embedded systems usually have limited memory. This challenges running many machine learning models, which often have high demands for read-only memory (storing the model) and RAM (handling intermediate results generated during inference). They often need more computing capability as well. Many machine learning models are quite compute-intensive, which can also pose problems.

Tiny Machine Learning (TinyML)

TinyML involves implementing the inference process of machine learning on the most restricted embedded hardware, such as Micro Processor Units (MCUs), Digital Signal Processors (DSPs), and Field Programmable Gate Arrays (FPGAs).

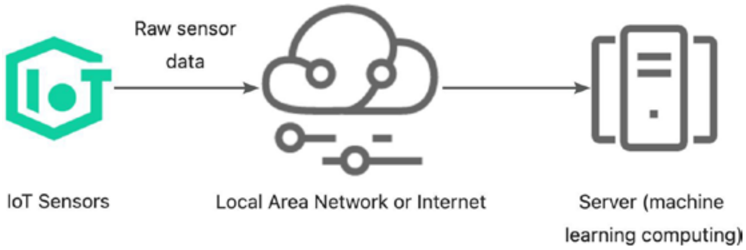
The image below helps to understand the relationship between these terms better.



Advantages and Operation of Edge AI

For many years, the Internet of Things (IoT) has been referred to as "machine-to-machine" (M2M). It involves connecting sensors and various computing devices for process automation control and has been widely adopted in industrial machinery and processes.

Machine learning offers the ability to make further progress in automation by introducing models that can make predictions or decisions without human intervention. Due to the complexity of many machine learning algorithms, the traditional integration of IoT and ML involves sending raw sensor data to a central server, which performs the necessary inference computations to generate predictions.


Raw sensor data
IoT Sensors
Local Area Network or Internet
Server (machine learning computing)

This configuration might be acceptable for low volumes of raw data and complex models. However, several potential problems have emerged:

- Transmitting extensive sensor data (like images) can consume a lot of network bandwidth.
- Data transmission also requires power.
- Sensors must continuously connect to the server to provide near real-time machine learning computation.

Given these challenges and the rapid development of machine learning, Edge AI has begun to emerge. Jeff Bier, founder of Edge AI and Vision Alliance, outlined five factors that push artificial intelligence to the edge in his article What's Driving AI and Vision to the Edge —**BLERP**, which stands for **Bandwidth**, **Latency**, **Economics**, **Reliability**, and **Privacy**.

- **Bandwidth:** If you have a commercial greenhouse, workshop, or mall with hundreds of cameras, it's impossible to send this information to the cloud for processing—the data would choke any type of internet connection you have. You simply need to process it locally.
- **Latency:** The latency here refers to the time between the system receiving sensor input and making a response. Consider autonomous vehicles: if a pedestrian suddenly appears at a crosswalk, the car's computer might only have a few hundred milliseconds to decide. There's not enough time to send the image to the cloud and await a response.
- **Economics:** Cloud computing and communication are getting better and cheaper, but they still cost money—possibly a lot of money, especially regarding video data. Edge computing reduces the amount of data that must be sent to the cloud and the computation workload once it arrives, significantly reducing costs.
- **Reliability:** Think of a home security system with facial recognition—even if there's an internet outage, you still want it to allow your family members to enter. Local processing makes this possible and gives the system more robust fault tolerance.
- **Privacy:** The rapid development of edge audio and video sensors has caused severe privacy issues, and sending this information to the cloud dramatically increases these concerns. The more information that can be processed locally, the less potential for abuse—what happens on the edge stays on the edge.

In most cases, training a machine learning model involves a three-step process of **Model → Training → Inference**, where obtaining the model requires more intensive computation than executing inference.

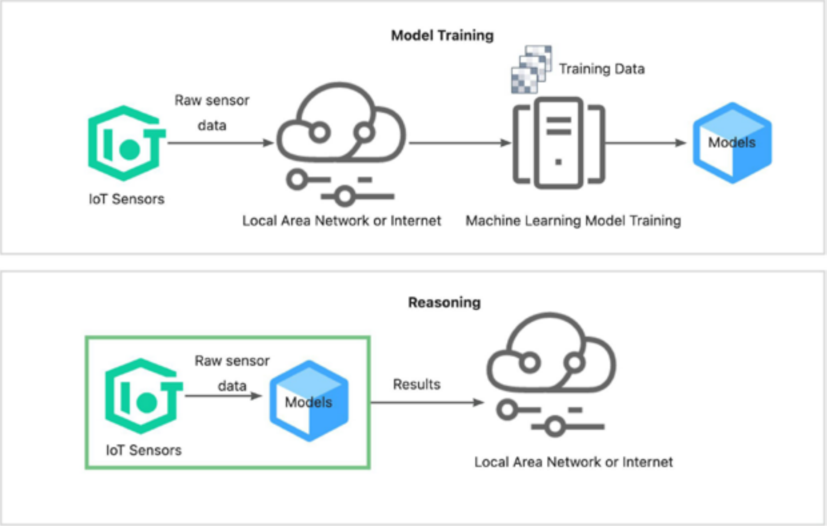
- **Model:** A mathematical formula trying to generalize information from a given dataset.
- **Training:** The process of automatically updating the parameters from data within a model. This model "learns" to make conclusions and generalize about the data.
- **Inference:** Providing new, unseen data to a trained model to make predictions, decisions, or classifications.

Thus, under normal circumstances, we would rely on powerful server clusters to train new models, constructing datasets from raw data collected on-site (images, sensor data, etc.) and using this dataset to train our machine learning models.

- Attention -

In some cases, we can train on the device side. However, this is generally unfeasible due to the memory and processing limitations of such edge devices.

Once we have a trained model, which is just a mathematical model (in the form of a software library), we can deploy it to our intelligent sensors or other edge devices. We can use this model to write firmware or software to collect new raw sensor readings, perform inferences, and take actions based on these inference results, as shown in the figure below. These actions could be self-driving cars, moving robotic arms, or sending notifications to users about engine failures. Since inference is performed locally on the edge device, the device does not need to maintain a network connection (the optional connection is shown as a dotted line in the chart).



4.1.3 Applications of Edge AI

Running machine learning models on edge devices without staying connected to a more powerful computer opens up possibilities for various automated tools and more intelligent IoT systems. Here are a few examples of edge AI enabling innovation in multiple industries.

Environmental Protection

- Smart grid monitoring to detect faults in power lines early
- Wildlife tracking and behavior research
- Forest fire detection and early warning

Agriculture

- Precision weeding, fertilization, pesticide application, or irrigation

Benjamin Cabé used TinyML technology to create an artificial nose thatdistinguishes between various distinct smells.

- Automatic recognition of irrigation needs
- Automatic recognition of crop status and disease/insect infestation conditions

Smart Buildings

- Monitoring of intrusions and recognition of abnormal states
- Air conditioning systems that adapt based on the number of people in a room

Health and Sports

- Wearable devices that track sleep and exercise conditions
- Portable medical devices
- Gesture recognition

Human-Machine Interaction

- Voice activation word detection
- Gesture and device motion recognition for auxiliary control

Industry

- Automatic safety helmet detection
- Machine, equipment, and facility condition monitoring
- Production line defect detection
- Position and motion state detection

The computational power typically required to perform machine learning inference at the edge is often more significant than simply polling sensors and transmitting raw data. However, locally, achieving such computations requires less power than sending raw data to a remote server.

The following table provides some suggestions on the type of hardware needed to perform machine learning inference at the edge, depending on the required application.

[Source: https://docs.edgeimpulse.com/docs/concepts/what-is-edge-machine-learning]

XIAO for this kind of task

	Low-End MCU	**High-End MCU**	**NPU** (Neural Network Processor)	**MPU** (Microprocessor)	**GPU** (Graphics Processor)
Task	Sensor Fusion Classification	Audio Classification	Image Classification	Complex Images or Sound and simple Videos	Video Classification
Memory	18KB	50KB	256KB	1MB+	1GB+
Sensors	✅	✅	✅	✅	✅
Audio		✅	✅	✅	✅
Images			✅	✅	✅
Videos				✅	✅
XIAO	**nRF52840 Sense**	**nRF52840 & ESP32S3 Sense**	**ESP32S3 Sense**	**ESP32S3 Sense**	**—**

> *Embedded hardware is also rapidly evolving, and it's expected that the contents of this table will change soon.*

4.1.4 Introduction to Edge Impulse Studio

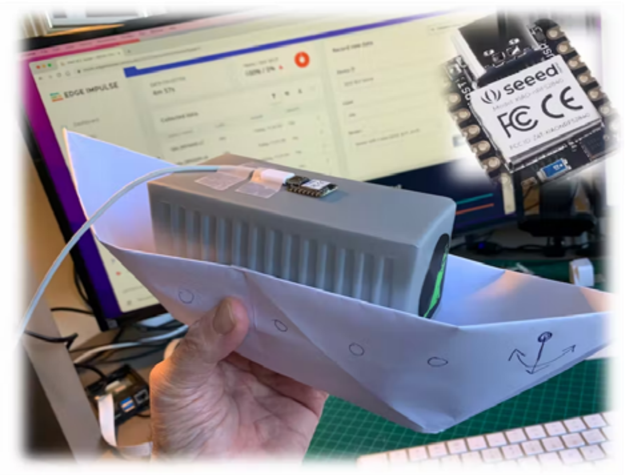
Edge Impulse was founded by Zach Shelby and Jan Jongboom in 2019. It is the leading edge device machine learning development platform. This platform allows developers to create and optimize solutions with real-world data, making the process of building, deploying, and scaling embedded ML applications more accessible and faster than ever before.

You can visit Edge Impulse's official website for more information about this tool and check the official documentation for a basic explanation.

In the following sections, we will learn to achieve continuous motion recognition with the on-board 6-axis accelerometer of the XIAO nRF52840 Sense shown below and voice keyword wake-up functionality using the on-board PDM microphone.

Computer Vision applications such as Image Classification and Object Detection will also be implemented using the camera of the XIAO ESP32S3 Sense shown below.

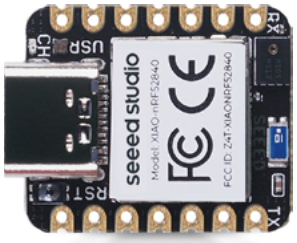
4.2
Anomaly Detection & Motion Classification



4.2.1 Things used in this project

Hardware components

Seeed Studio XIAO nRF52840 Sense × 1



Software apps and online services

- Arduino IDE
- Edge Impulse Studio

4.2.2 Introduction

As you learned in the previous section, microcontrollers (MCUs) are very cheap electronic components, usually with just a few kilobytes of RAM, designed to use tiny amounts of energy. They can be found in almost any consumer, medical, automotive, and industrial device. Over 40 billion microcontrollers will be sold this year, and there are probably hundreds of billions in service nowadays. However, these devices get little attention because they're often only used to replace the functionality of older electro-mechanical systems in cars, washing machines, or remote controls. More recently, with the Internet of Things (IoT) era, a significant part of those MCUs is generating "quintillions" of data that, in its majority, is not used due to the high cost and complexity (bandwidth and latency) of data transmission.

On the other hand, in recent decades, we have seen a lot of development in Machine Learning models trained with vast amounts of data in very powerful and power-hungry mainframes. And what is happening today is that due to those developments, it is now possible to take noisy signals like images, audio, or accelerometers and extract meaning from them by using Machine Learning algorithms such as Neural Networks.

And what is more important is that we can run these algorithms on microcontrollers and sensors themselves using very little power, interpreting much more of those sensor data that we are currently ignoring. This is TinyML, a new technology that enables machine intelligence right next to the physical world.

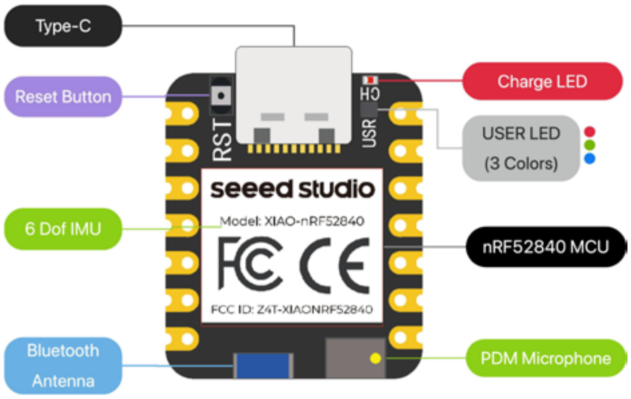
> *TinyML can have many exciting applications for the benefit of society at large.*

This section will explore TinyML, running on a robust and tiny device, the Seed XIAO nRF52840 Sense (also called XIAO BLE Sense).

4.2.3 XIAO nRF52840 Sense

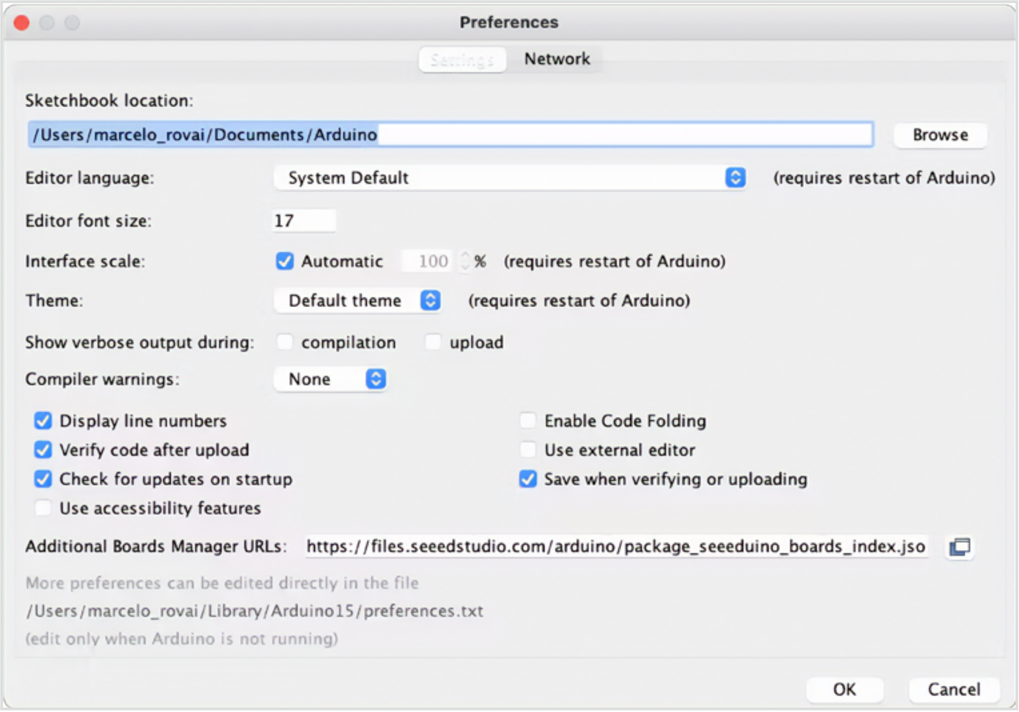
MainFeatures

- Bluetooth 5.0 with onboard antenna
- CPU: Nordic nRF52840, ARM® Cortex®-M4 32-bit processor with FPU, 64 MHz
- Ultra-Low Power: Standby power consumption is less than 5μA
- Battery charging chip: Supports lithium battery charge and discharge management
- 2 MB flash
- 256 KB RAM
- PDM microphone
- 6-axis LSM6DS3TR-C IMU
- Ultra Small Size: 20 x 17.5mm, XIAO series classic form-factor for wearable devices
- Rich interfaces: 1xUART, 1xI2C, 1xSPI, 1xNFC, 1xSWD, 11xGPIO(PWM), 6xADC
- Single-sided components, surface mounting design



Connecting the XIAO nRF52840 Sense with Arduino IDE

The simple way to test and use this device is using the Arduino IDE. Once you have the IDE installed on your machine, navigate to `File > Preferences`, and fill in "Additional Boards Manager URLs" with the URL below: `https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json`



Preferences

Settings | Network

Sketchbook location:
/Users/marcelo_rovai/Documents/Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 17

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

☑ Display line numbers
☑ Verify code after upload
☑ Check for updates on startup
□ Use accessibility features
□ Enable Code Folding
□ Use external editor
☑ Save when verifying or uploading

Additional Boards Manager URLs: https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.jso

More preferences can be edited directly in the file
/Users/marcelo_rovai/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

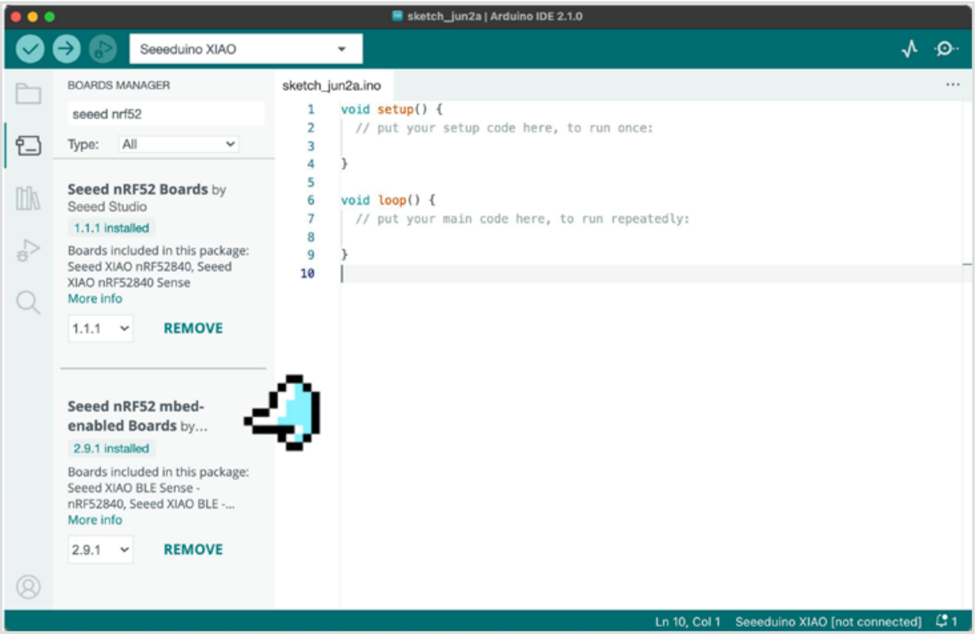
[OK] [Cancel]

Now, navigate to `Tools→Board→Board Manager` in the top menu, and type in the filter keyword `seeed nrf52` in the search box.

You will see two installation packages: `Seeed nRF52 Boards` and `Seeed nRF52 mbed-enabled Boards`, the differences between these two packages are as follows:

- `Seeed nRF52 Boards`: Friendly for Bluetooth and low-power compatibility, suitable for Bluetooth and low power applications.
- `Seeed nRF52 mbed-enabled Boards`: Friendly for TinyML support, suitable for making TinyML or Bluetooth-related projects, but not suitable for applications with high low-power requirements.

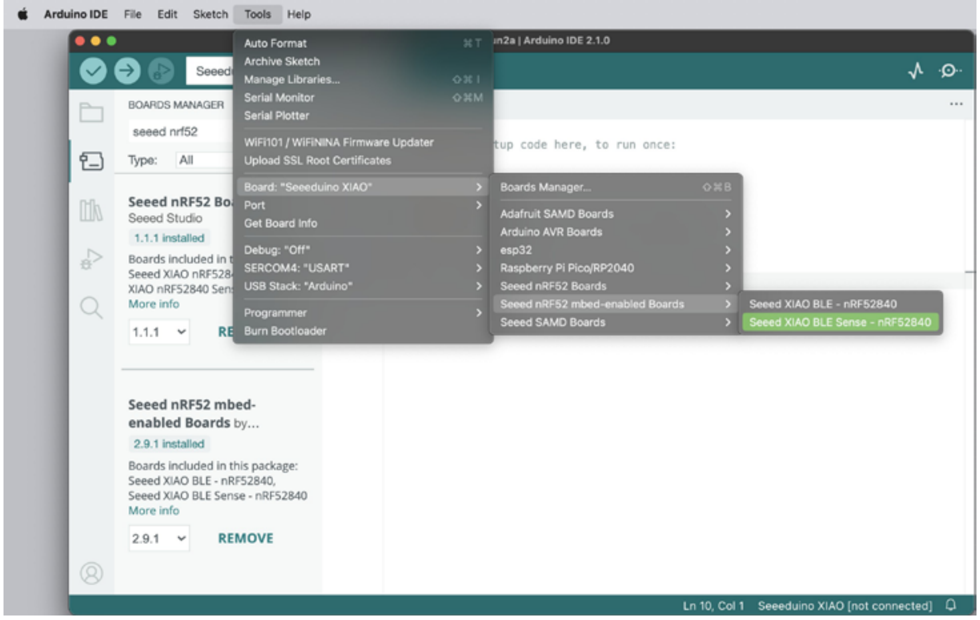
Because we will develop a TinyML project, we chose the latest version of the `Seeed nRF52 mbed-enabled Boards package`. Install it and wait until you see a successful installation prompt in the output window.



sketch_jun2a | Arduino IDE 2.1.0
Seeeduino XIAO
BOARDS MANAGER
seeed nrf52
Type: All
Seeed nRF52 Boards by Seeed Studio
1.1.1 installed
Boards included in this package: Seeed XIAO nRF52840, Seeed XIAO nRF52840 Sense
More info
1.1.1
REMOVE
Seeed nRF52 mbed-enabled Boards by...
2.9.1 installed
Boards included in this package: Seeed XIAO BLE Sense - nRF52840, Seeed XIAO BLE -...
More info
2.9.1
REMOVE
sketch_jun2a.ino
Ln 10, Col 1 Seeeduino XIAO [not connected]

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

```

Now, you can access this device from your Arduino IDE by selecting the development board and serial port, as shown in the figure below.

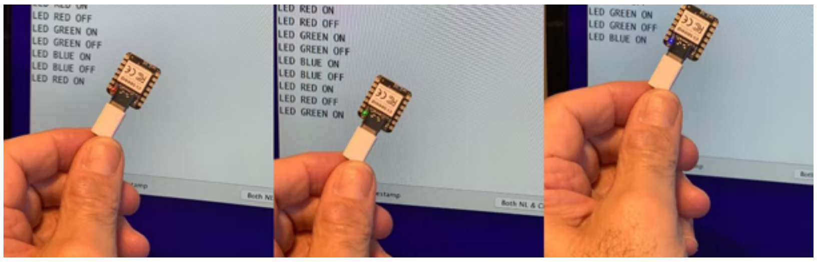


Your development board is now ready to run code on it. Let's start with Blink - lighting up the LED. Note that the board does not have a regular LED like most Arduino boards. Instead, you will find an RGB LED that can be activated with "reverse logic" (you should apply LOW to activate each of the three separate LEDs). Test your RGB LED with the following code:

```
void setup() {

  // initialize serial.
  Serial.begin(115200);
  while (!Serial);
  Serial.println(“Serial Started”);

  // Pins for the built-in RGB LEDs on the Arduino Nano 33 BLE Sense
  pinMode(LEDR, OUTPUT);
  pinMode(LEDG, OUTPUT);
  pinMode(LEDB, OUTPUT);

  // Note: The RGB LEDs are ON when the pin is LOW and off when HIGH.
  digitalWrite(LEDR, HIGH);
  digitalWrite(LEDG, HIGH);
  digitalWrite(LEDB, HIGH);

}

void loop() {
  digitalWrite(LEDR, LOW);
  Serial.println(“LED RED ON”);
  delay(1000);
  digitalWrite(LEDR, HIGH);
  Serial.println(“LED RED OFF”);
  delay(1000);

  digitalWrite(LEDG, LOW);
  Serial.println(“LED GREEN ON”);
  delay(1000);
  digitalWrite(LEDG, HIGH);
  Serial.println(“LED GREEN OFF”);
  delay(1000);

  digitalWrite(LEDB, LOW);
  Serial.println(“LED BLUE ON”);
  delay(1000);
  digitalWrite(LEDB, HIGH);
  Serial.println(“LED BLUE OFF”);
  delay(1000);
}
```

Get this code online https://github.com/Mjrovai/Seeed-XIAO-BLE-Sense/tree/main/Seeed_Xiao_Sense_bilnk_RGB

Here is the result:

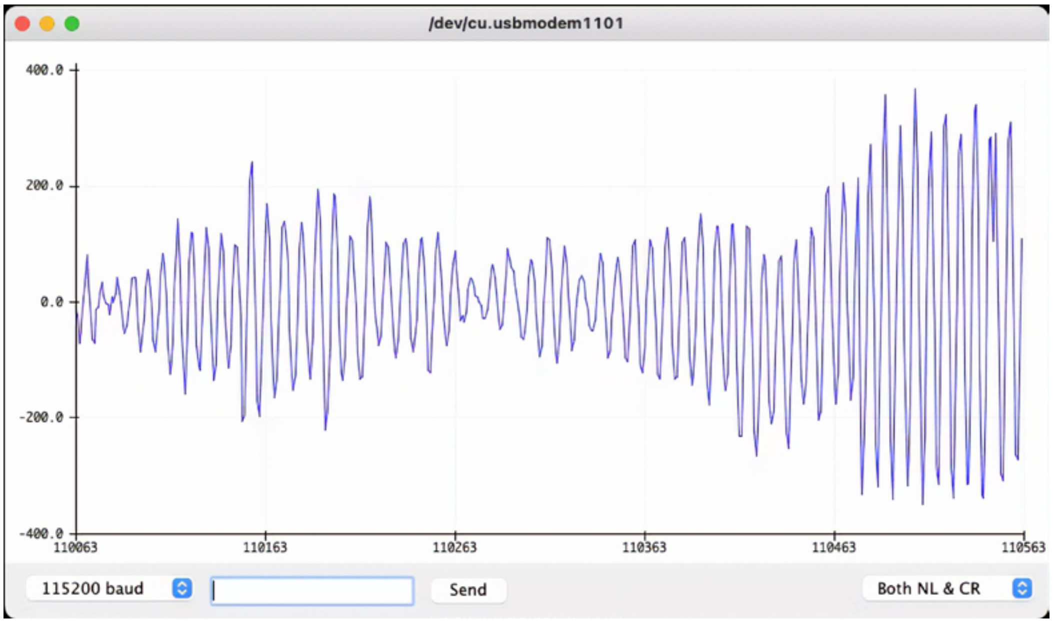


Testing the Microphone

The XIAO nRF52840 Sense has a PDM digital output MEMS microphone. Run the below code for testing it:

```
#include <PDM.h>

// buffer to read samples into, each sample is 16–bits
short sampleBuffer[256];

// number of samples read
volatile int samplesRead;

void setup() {
    Serial.begin(9600);
    while (!Serial);

    // configure the data receive callback
    PDM.onReceive(onPDMdata);

    // optionally set the gain, defaults to 20
    // PDM.setGain(30);

    // initialize PDM with:
    // – one channel (mono mode)
    // – a 16 kHz sample rate
    if (!PDM.begin(1, 16000)) {
        Serial.println(“Failed to start PDM!”);
        while (1);
    }
}

void loop() {
    // wait for samples to be read
    if (samplesRead) {

        // print samples to the serial monitor or plotter
        for (int i = 0; i < samplesRead; i++) {
            Serial.println(sampleBuffer[i]);
            // check if the sound value is higher than 500
            if (sampleBuffer[i]>=500){
                digitalWrite(LEDR,LOW);
                digitalWrite(LEDG,HIGH);
                digitalWrite(LEDB,HIGH);
            }
            // check if the sound value is higher than 250 and lower than 500
            if (sampleBuffer[i]>=250 && sampleBuffer[i] < 500){
                digitalWrite(LEDB,LOW);
                digitalWrite(LEDR,HIGH);
                digitalWrite(LEDG,HIGH);
            }
            //check if the sound value is higher than 0 and lower than 250
            if (sampleBuffer[i]>=0 && sampleBuffer[i] < 250){
                digitalWrite(LEDG,LOW);
                digitalWrite(LEDR,HIGH);
                digitalWrite(LEDB,HIGH);
            }
        }

        // clear the read count
        samplesRead = 0;
    }
}

void onPDMdata() {
    // query the number of bytes available
    int bytesAvailable = PDM.available();

    // read into the sample buffer
    PDM.read(sampleBuffer, bytesAvailable);

    // 16-bit, 2 bytes per sample
    samplesRead = bytesAvailable / 2;
}
```

The above code will continuously capture data to its buffer, displaying it in the Serial Monitor and Plotter:



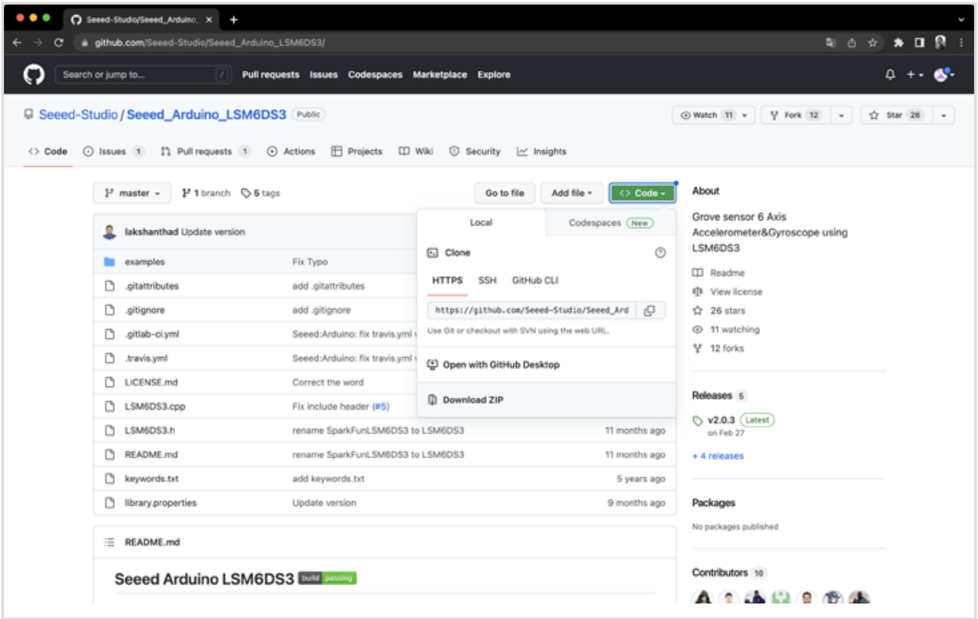
Also, note that the RGB LED will be set up depending on the intensity of sound.

> *The Micrphone will not be used on this project in particular, but it is good to have it tested if it is your first time using the XIAO nRF52840 Sense.*

Testing the IMU

Our tiny device also has integrated a 6-Axis IMU, the LSM6DS3TR-C, a system-in-package 3D digital accelerometer, and a 3D digital gyroscope. For testing, you should first install its library 'Seeed Arduino LSM6DS3'.

Before programming the accelerometer with the Arduino IDE, you must add the necessary library for the sensor. Enter the library address https://github.com/Seeed-Studio/Seeed_Arduino_LSM6DS3/ in the browser address bar, go to the GitHub page, click `Code→Download ZIP` to download the resource pack `Seeed_Arduino_LSM6DS3-master.zip` to the local area, as shown below.



lakshanthad Update version

examples	Fix Typo	
.gitattributes	add .gitattributes	
.gitignore	add .gitignore	
.gitlab-ci.yml	Seeed:Arduino: fix travis.yml	
.travis.yml	Seeed:Arduino: fix travis.yml	
LICENSE.md	Correct the word	
LSM6DS3.cpp	Fix include header (#5)	
LSM6DS3.h	rename SparkFunLSM6DS3 to LSM6DS3	11 months ago
README.md	rename SparkFunLSM6DS3 to LSM6DS3	11 months ago
keywords.txt	add keywords.txt	5 years ago
library.properties	Update version	9 months ago

https://github.com/Seeed-Studio/Seeed_Ard

Use Git or checkout with SVN using the web URL.

Grove sensor 6 Axis Accelerometer&Gyroscope using LSM6DS3

Seeed Arduino LSM6DS3

Add the resource pack `Seeed_Arduino_LSM6DS3-master.zip` downloaded in the previous step in the menu bar's `Sketch→Include Library→Add .ZIP Library` until you see a prompt that the library has been loaded successfully.

Run the test code based on Harvard University's tinymlx - Sensor Test

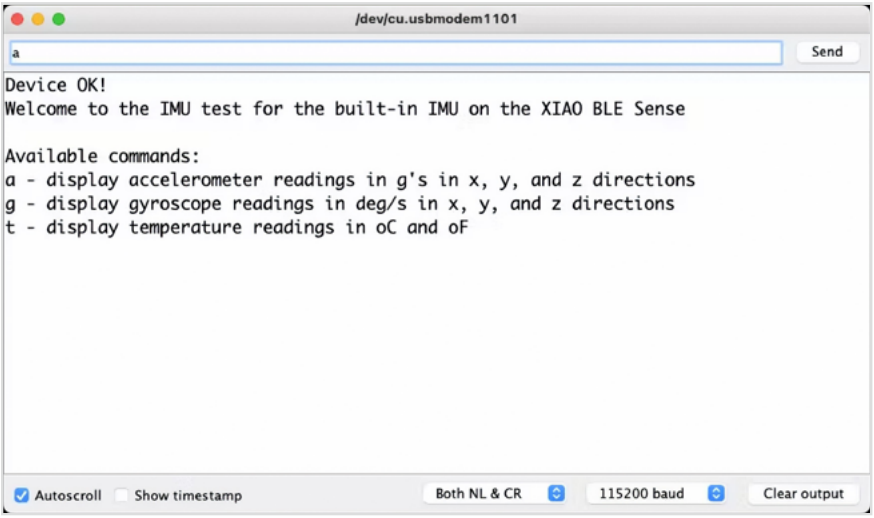
Now, run the following test code based on Harvard University's tinymlx - Sensor Test.

```
#include “LSM6DS3.h”
#include “Wire.h”

//Create an instance of class LSM6DS3
LSM6DS3 xIMU(I2C_MODE, 0x6A);    //I2C device address 0x6A

char c;
int sign = 0;

void setup() {
  Serial.begin(115200);
  while (!Serial);

  // configure the IMU
  if (xIMU.begin() != 0) {
      Serial.println(“Device error”);
  } else {
      Serial.println(“Device OK!”);
  }

   Serial.println(“Welcome to the IMU test for the built–in IMU on the XIAO BLE
Sense\n”);
  Serial.println(“Available commands:”);
   Serial.println(“a – display accelerometer readings in g’s in x, y, and z direc–
tions”);
   Serial.println(“g – display gyroscope readings in deg/s in x, y, and z direc–
tions”);
  Serial.println(“t – display temperature readings in oC and oF”);
}
```

```
void loop() {
  // Read incoming commands from serial monitor

  if (Serial.available()) {
    c = Serial.read();
    Serial.println(c);
  }

  if(c == ‘a’)sign=1;
  else if(c == ‘g’)sign=2;
  else if(c == ‘t’)sign=3;

  float x, y, z;
  if (sign ==1) { // testing accelerometer
      //Accelerometer
      x = xIMU.readFloatAccelX();
      y = xIMU.readFloatAccelY();
      z = xIMU.readFloatAccelZ();
      Serial.print(“\nAccelerometer:\n”);
      Serial.print(“Ax:”);
      Serial.print(x);
      Serial.print(‘ ‘);
      Serial.print(“Ay:”);
      Serial.print(y);
      Serial.print(‘ ‘);
      Serial.print(“Az:”);
      Serial.println(z);
    }
  else if (sign ==2) { // testing gyroscope
      //Gyroscope
      Serial.print(“\nGyroscope:\n”);
      x = xIMU.readFloatGyroX();
      y = xIMU.readFloatGyroY();
      z = xIMU.readFloatGyroZ();
      Serial.print(“wx:”);
      Serial.print(x);
      Serial.print(‘ ‘);
      Serial.print(“wy:”);
      Serial.print(y);
      Serial.print(‘ ‘);
      Serial.print(“wz:”);
      Serial.println(z);
    }
  else if (sign ==3) { // testing thermometer
       //Thermometer
      Serial.print(“\nThermometer:\n”);
      Serial.print(“ Degrees oC = “);
      Serial.println(xIMU.readTempC(), 0);
      Serial.print(“ Degrees oF = “);
      Serial.println(xIMU.readTempF(), 0);
      delay(1000);
    }
}
```

Get this code online https://github.com/Mjrovai/Seeed-XIAO-BLE-Sense/blob/main/xiao_test_IMU/xiao_test_IMU.ino

Once you run the above sketch, open the Serial Monitor:

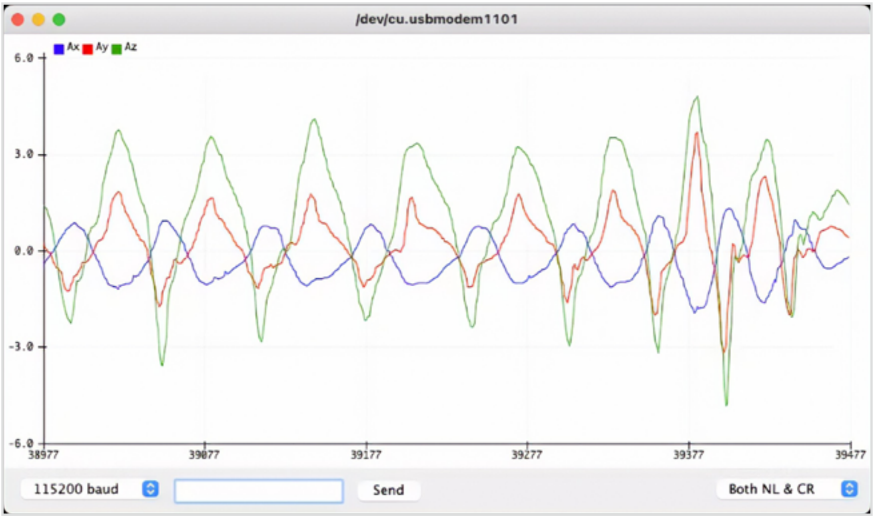


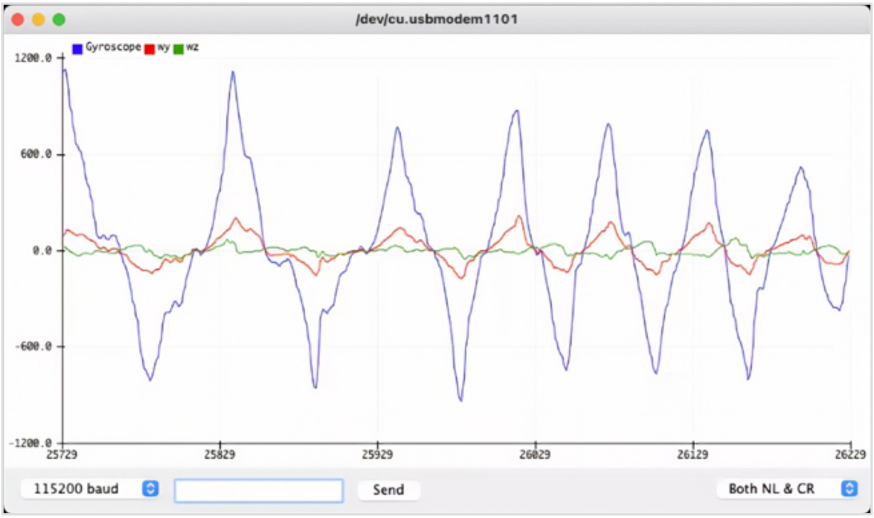
```
Device OK!
Welcome to the IMU test for the built-in IMU on the XIAO BLE Sense

Available commands:
a - display accelerometer readings in g's in x, y, and z directions
g - display gyroscope readings in deg/s in x, y, and z directions
t - display temperature readings in oC and oF
```

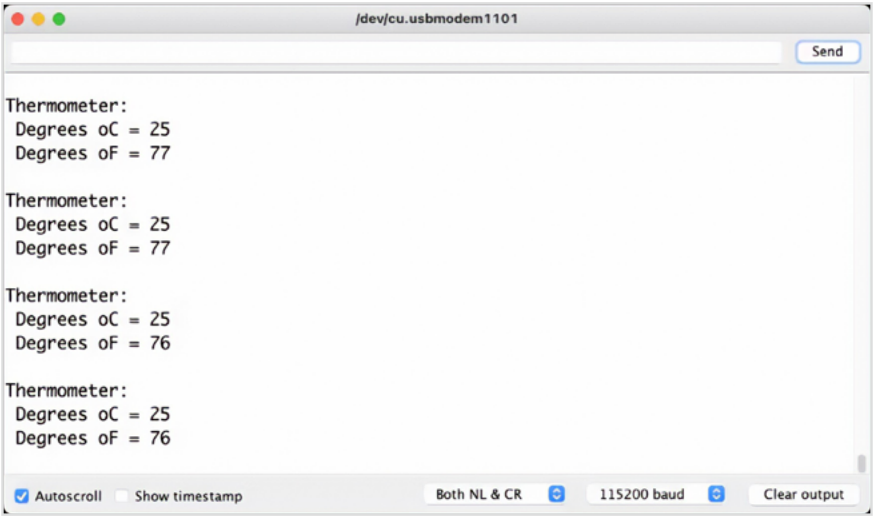
Choose one of the three options to test:

- **a:** Accelerometer (see the result on Plotter)
- **g:** Gyroscope (see the result on Plotter)
- **t:** Temperature (see the result on Serial Monitor)

The following images show the result:





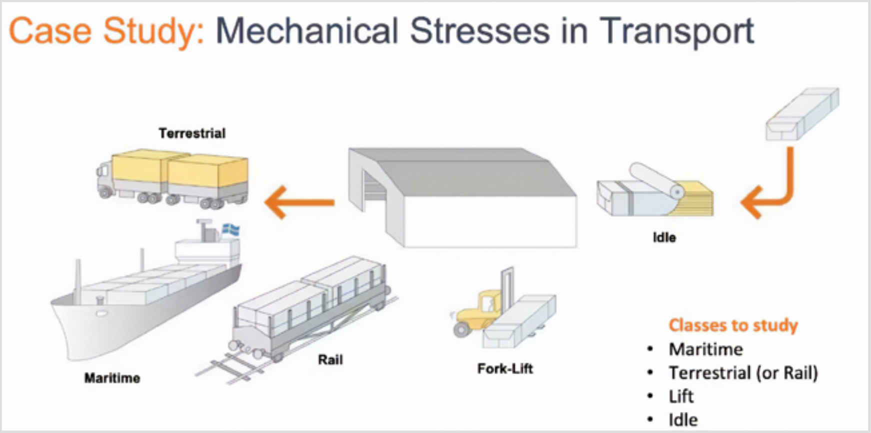


```
Thermometer:
 Degrees oC = 25
 Degrees oF = 77

Thermometer:
 Degrees oC = 25
 Degrees oF = 77

Thermometer:
 Degrees oC = 25
 Degrees oF = 76

Thermometer:
 Degrees oC = 25
 Degrees oF = 76
```

4.2.4 The TinyML Motion Classification Model

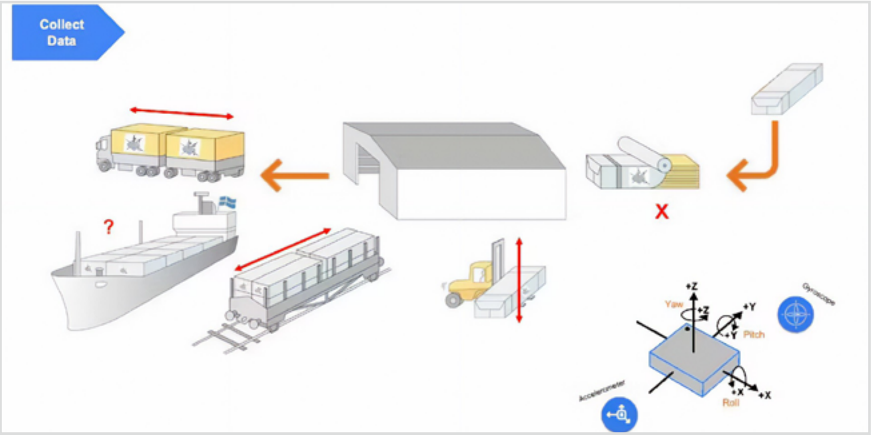
For our project, we will simulate mechanical stresses in transport. Our problem will be to classify four classes of movement:

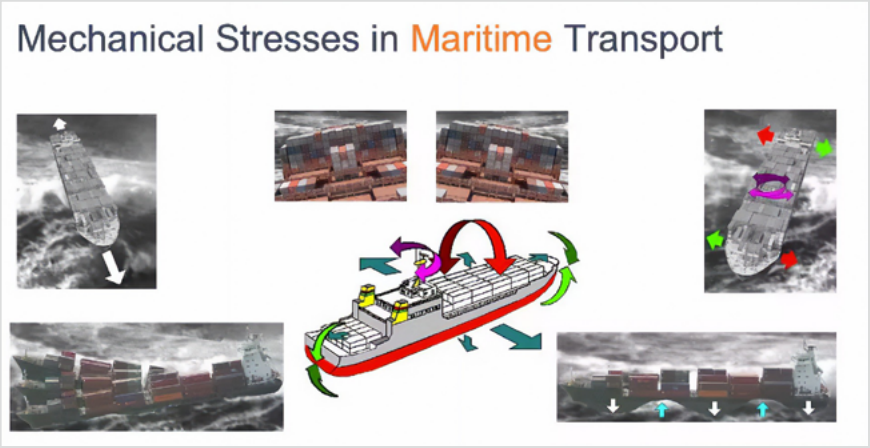
- **Maritime** (pallets in boats)
- **Terrestrial** (palettes in a Truck or Train)
- **Lift** (Palettes being handled by Fork-Lift)
- **Idle** (Palettes in Storage houses)


Case Study: Mechanical Stresses in Transport

Classes to study
- Maritime
- Terrestrial (or Rail)
- Lift
- Idle

So, to start, we should collect data. Then, accelerometers will provide the data on the palette (or container).





From the above images, we can see that primarily horizontal movements should be associated with "Terrestrial class," Vertical movements to "Lift Class," no activity to "Idle class," and movent on all three axes to Maritime class.

Connecting a Device to the Edge Impulse Studio

For data collection, we can have several options. In a real case, we can have our device, for example, connected directly to one container, and the data collected on a file (for example .CSV) and stored on an SD card (via SPI connection) or an offline repo in your computer. Data can also be sent remotely to a nearby repository, such as a mobile phone, using Bluetooth as done in this project: Sensor DataLogger. Once your dataset is collected and stored as a .CSV file, it can be uploaded to the Studio using the CSV Wizard tool.

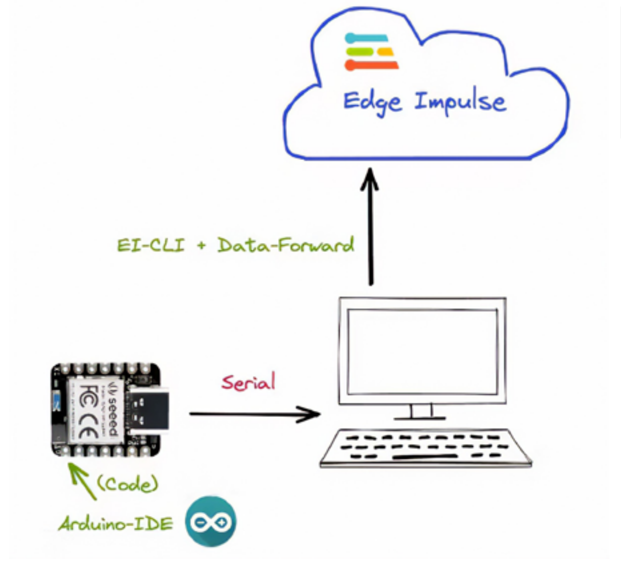
> *In this video, you can learn alternative ways to send data to the Edge Impulse Studio.*

In this project, we should first connect our device to the Edge Impulse Studio for data collection, which will also be used for data pre-processing, model training, testing, and deployment.

> *Follow the instructions here to install the Node.js and Edge Impulse CLI on your computer.*

Once the XIAO nRF52840 Sense is not a fully supported development board by Edge Impulse, we should use the CLI Data Forwarder to capture data from the accelerometer and send it to the Studio, as shown in this diagram:

Your device should be connected to the computer serial and running a code to capture IMU (Accelerometer) data and "print them" on the serial. Further, the Edge Impulse Studio will "capture" them. Run the code below:



```
#include "LSM6DS3.h"
#include "Wire.h"

//Create an instance of class LSM6DS3
LSM6DS3 xIMU(I2C_MODE, 0x6A);    //I2C device address 0x6A

#define CONVERT_G_TO_MS2    9.80665f
```

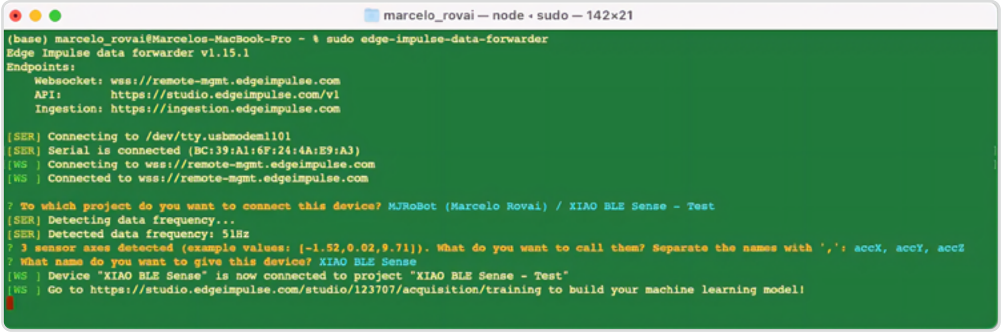
```
#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))

static unsigned long last_interval_ms = 0;

void setup() {
    Serial.begin(115200);
    while (!Serial);

    // configure the IMU
    if (xIMU.begin() != 0) {
        Serial.println(“Device error”);
    } else {
        Serial.println(“Device OK!”);
    }

     Serial.println(“Data Forwarder – Built-in IMU (Accelerometer) on the XIAO BLE 
Sense\n”);
}

void loop() {
    float x, y, z;

    if (millis() > last_interval_ms + INTERVAL_MS) {
        last_interval_ms = millis();

        x = xIMU.readFloatAccelX();
        y = xIMU.readFloatAccelY();
        z = xIMU.readFloatAccelZ();

        Serial.print(x * CONVERT_G_TO_MS2);
        Serial.print(‘\t’);
        Serial.print(y * CONVERT_G_TO_MS2);
        Serial.print(‘\t’);
        Serial.println(z * CONVERT_G_TO_MS2);
    }
}
```

Get this code online https://github.com/Mjrovai/Seeed-XIAO-BLE-Sense/blob/main/XIAO_BLE_Sense_Accelerometer_Data_Forewarder/XIAO_BLE_Sense_Accelerometer_Data_Forewarder.ino

Go to the Edge Impulse page and create a project. Next, start the CLI Data Forwarder on your terminal, entering (if it is the first time) the following command:

```
$ edge-impulse-data-forwarder --clean
```

Next, enter your EI credentials, and choose your project, variable, and device names:

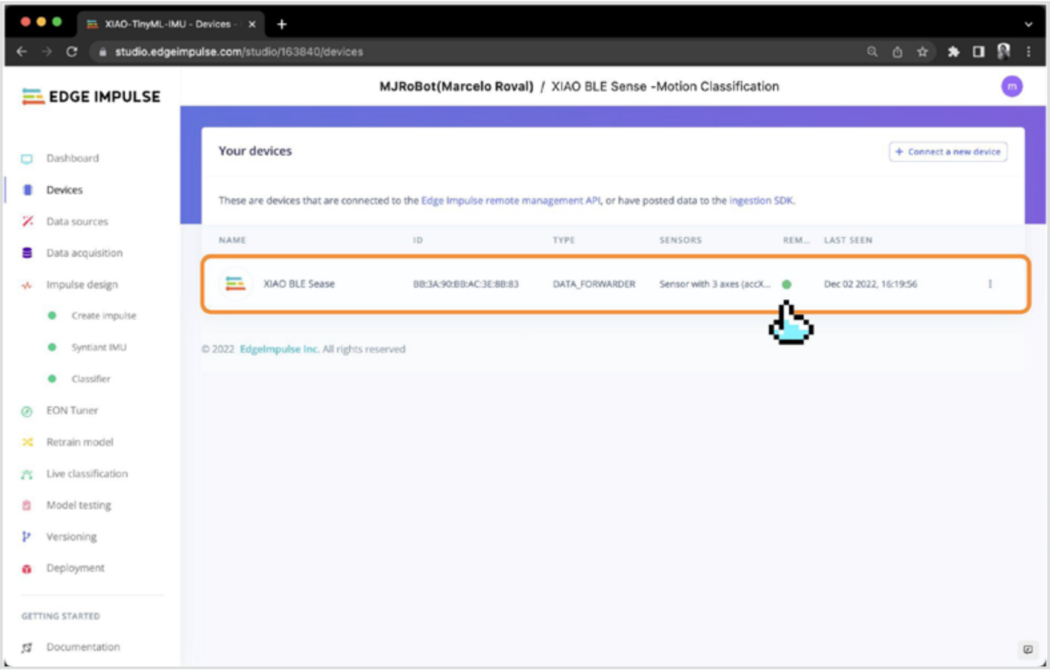


marcelo_rovai — node ‹ sudo — 142×21

```
(base) marcelo_rovai@Marcelos-MacBook-Pro ~ % sudo edge-impulse-data-forwarder
Edge Impulse data forwarder v1.15.1
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com/v1
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/tty.usbmodem1101
[SER] Serial is connected (BC:39:A1:6F:24:4A:E9:A3)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

? To which project do you want to connect this device? MJRoBot (Marcelo Rovai) / XIAO BLE Sense - Test
[SER] Detecting data frequency...
[SER] Detected data frequency: 51Hz
? 3 sensor axes detected (example values: [-1.52,0.02,9.71]). What do you want to call them? Separate the names with ',': accX, accY, accZ
? What name do you want to give this device? XIAO BLE Sense
[WS ] Device "XIAO BLE Sense" is now connected to project "XIAO BLE Sense - Test"
[WS ] Go to https://studio.edgeimpulse.com/studio/123707/acquisition/training to build your machine learning model!
```

> *The Studio can read the sampled frequency as 51Hz instead of the 50Hz previously defined in the code. It is OK.*

Go to the `Devices` section on your EI Project and verify if the device is connected (the dot should be green):



Your devices

These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

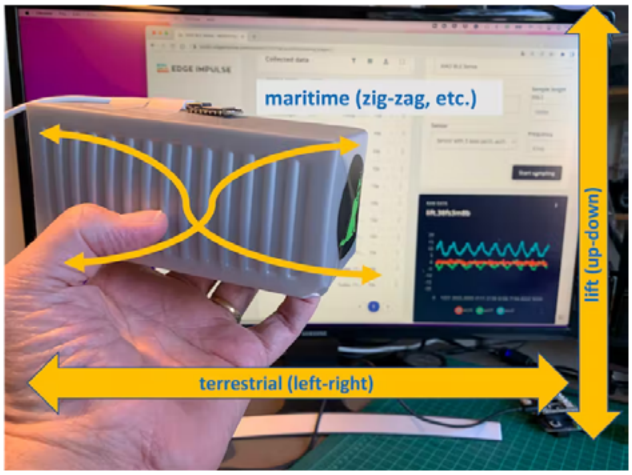
NAME	ID	TYPE	SENSORS	REM...	LAST SEEN
XIAO BLE Sease	BB:3A:90:BB:AC:3E:BB:83	DATA_FORWARDER	Sensor with 3 axes (accX...	●	Dec 02 2022, 16:19:56

© 2022 EdgeImpulse Inc. All rights reserved

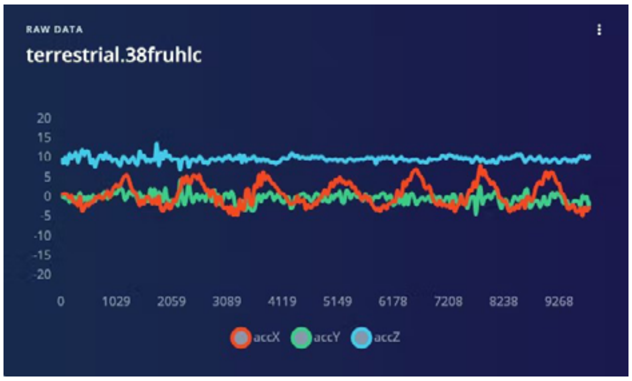
Data Collection

As discussed before, we should capture data from all four Transportation Classes:

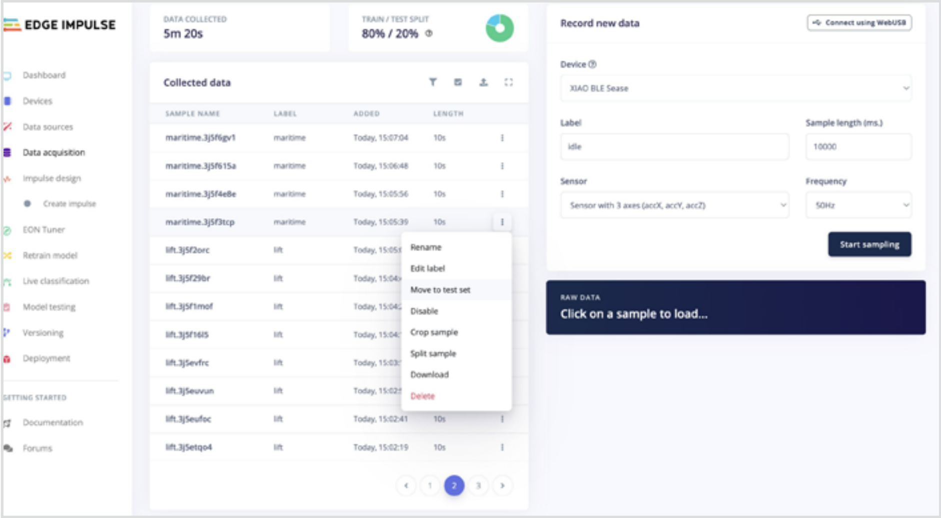
- **lift** (up-down)
- **terrestrial** (left-right)
- **maritime** (zig-zag, etc.)
- **idle**



Below is one sample (10 seconds of raw data):



You can capture, for example, around 2 minutes (twelve samples of 10 seconds) for each of the four classes (a total of 8 minutes of data). Using the `three dots` menu after each one of the samples, select 2 of them, reserving them for the Test set. Alternatively, you can use the automatic Train/Test Split tool on the Danger Zone of Dashboard tab.



DATA COLLECTED
5m 20s

TRAIN / TEST SPLIT
80% / 20%

Collected data

SAMPLE NAME	LABEL	ADDED	LENGTH
maritime.3j5f6gv1	maritime	Today, 15:07:04	10s
maritime.3j5f615a	maritime	Today, 15:06:48	10s
maritime.3j5f4e8e	maritime	Today, 15:05:56	10s
maritime.3j5f3tcp	maritime	Today, 15:05:39	10s
lift.3j5f2orc	lift	Today, 15:05:	
lift.3j5f29br	lift	Today, 15:04:	
lift.3j5f1mof	lift	Today, 15:04:	
lift.3j5f16l5	lift	Today, 15:04:	
lift.3j5evfrc	lift	Today, 15:03:	
lift.3j5euvun	lift	Today, 15:02:	
lift.3j5eufoc	lift	Today, 15:02:41	10s
lift.3j5etqo4	lift	Today, 15:02:19	10s

Record new data

Device: XIAO BLE Sease

Label: idle

Sample length (ms.): 10000

Sensor: Sensor with 3 axes (accX, accY, accZ)

Frequency: 50Hz

RAW DATA
Click on a sample to load...

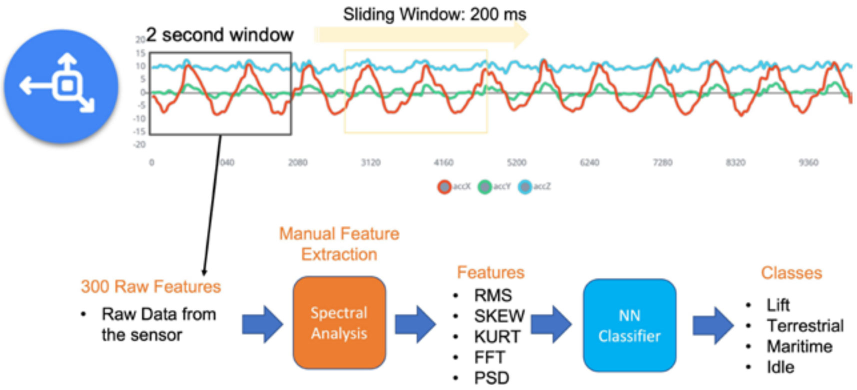
Once you have captured your dataset, you can explore it in more detail using the Data Explorer, a visual tool to find outliers or mislabeled data (helping to correct them). The data explorer first tries to extract meaningful features from your data (by applying signal processing and neural network embeddings) and then uses a dimensionality reduction algorithm such as PCA or t-SNE to map these features to a 2D space. This gives you a one-look overview of your complete dataset.

Data Pre-Processing

Data pre-processing is extracting features from the dataset captured with the accelerometer, which involves processing and analyzing the raw data. Accelerometers measure the acceleration of an object along one or more axes (typically three, denoted as X, Y, and Z). These measurements can be used to understand various aspects of the object's motion, such as movement patterns and vibrations.

Raw accelerometer data can be noisy and contain errors or irrelevant information. Preprocessing steps, such as filtering and normalization, can clean and standardize the data, making it more suitable for feature extraction. In our case, we should divide the data into smaller segments or **windows**. This can help focus on specific events or activities within the dataset, making feature extraction more manageable and meaningful. The **window size** and overlap (**window increase**) choice depend on the application and the frequency of the events of interest. As a thumb rule, we should try to capture a couple of "cycles of data".

With a sampling rate (SR) of 50Hz and a window size of 2 seconds, we will get 100 samples per axis, or 300 in total (3 axis x 2 seconds x 50 samples). We will slide this window every 200ms, creating a larger dataset where each instance has 300 raw features.



Once the data is preprocessed and segmented, you can extract features that describe the motion's characteristics. Some typical features extracted from accelerometer data include: - **Time-domain** features describe the data's statistical properties within each segment, such as mean, median, standard deviation, skewness, kurtosis, and zero-crossing rate. - **Frequency-domain** features are obtained by transforming the data into the frequency domain using techniques like the Fast Fourier Transform (FFT). Some typical frequency-domain features include the power spectrum, spectral energy, dominant frequencies (amplitude and frequency), and spectral entropy. - **Time-frequency** domain features combine the time and frequency domain information, such as the Short-Time Fourier Transform (STFT) or the Discrete Wavelet Transform (DWT). They can provide a more detailed understanding of how the signal's frequency content changes over time.

In many cases, the number of extracted features can be large, which may lead to overfitting or increased computational complexity. Feature selection techniques, such as mutual information, correlation-based methods, or principal component analysis (PCA), can help identify the most relevant features for a given application and reduce the dimensionality of the dataset. The Studio can help with such feature importance calculations.

EI Studio Spectral Features

Data preprocessing is a challenging area for embedded machine learning. Still, Edge Impulse helps overcome this with its digital signal processing (DSP) preprocessing step and, more specifically, the Spectral Features Block.

On the Studio, the collected raw dataset will be the input of a Spectral Analysis block, which is excellent for analyzing repetitive motion, such as data from accelerometers. This block will perform a DSP (Digital Signal Processing), extracting features such as FFT or Wavelets.

For our project, once the time signal is continuous, we should use FFT with, for example, a length of `[32]`.

The per axis/channel **Time Domain Statistical features** are:

- RMS: 1 feature
- Skewness: 1 feature
- Kurtosis: 1 feature

The per axis/channel **Frequency Domain Spectral features** are:

- Spectral Power: 16 features (FFT Length/2)
- Skewness: 1 feature
- Kurtosis: 1 feature

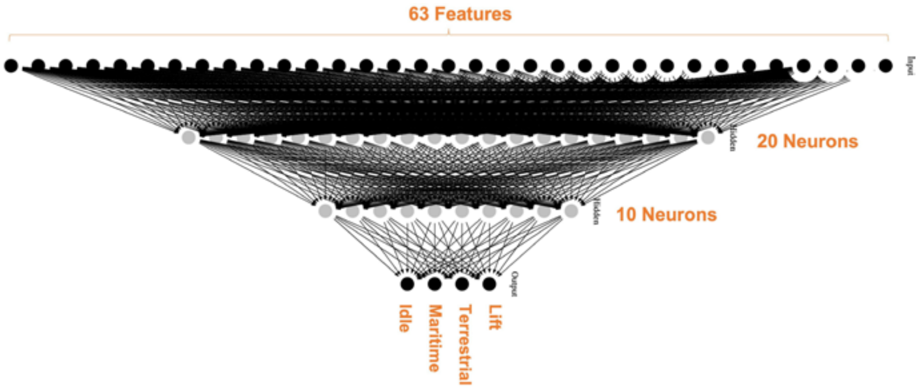
So, for an FFT length of 32 points, the resulting output of the Spectral Analysis Block will be 21 features per axis (a total of 63 features).

> *You can learn more about how each feature is calculated by downloading the notebook Edge Impulse - Spectral Features Block Analysis TinyML under the hood: Spectral Analysis or opening it directly on Google CoLab.*

Those 63 features will be the Input Tensor of a Neural Network Classifier.

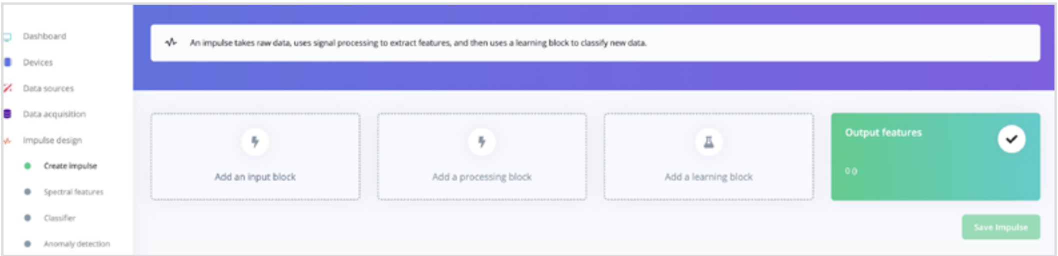
Model Design

Our classifier will be a Dense Neural Network (DNN) that will have 63 neurons on its input layer, two hidden layers with 20 and 10 neurons, and an output layer with four neurons (one per each class), as shown here:



Impulse Design

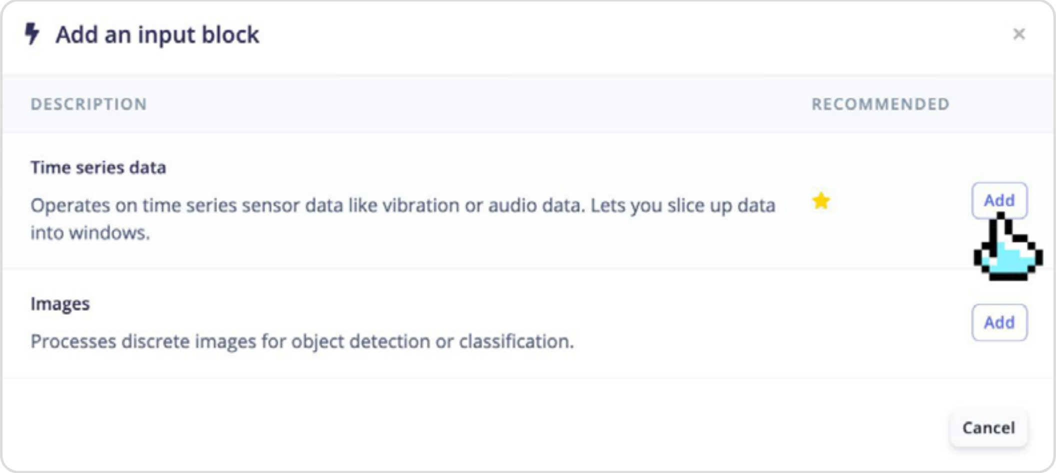
A complete Impulse comprises three primary building blocks: the input block - which obtains the raw data, the processing block - which extracts features, and the learning block - which classifies the data. The following image shows the interface when the three building blocks still need to be added, and our machine-learning pipeline will be implemented by adding these three blocks.


An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Impulse obtains raw data through the input block, uses the processing block to extract features, and then uses the learning block to classify new data. In our continuous action recognition, the added blocks include:

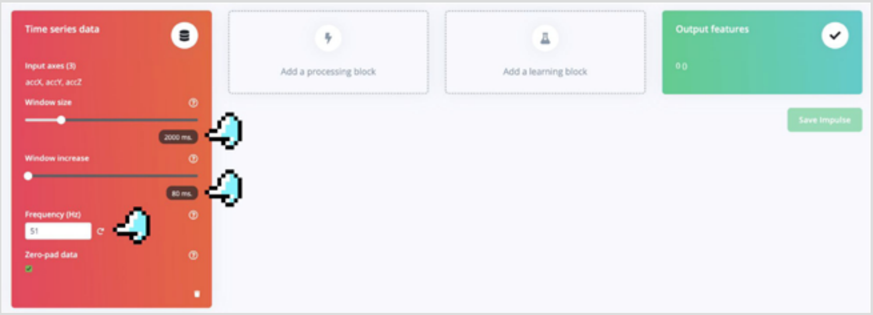
1. Adding the input block: Time Series Data

Click the **“Add an Input Block”** button and select **Time Series Data** in the pop-up window as shown below to match the sensor data type we collected.


Add an input block

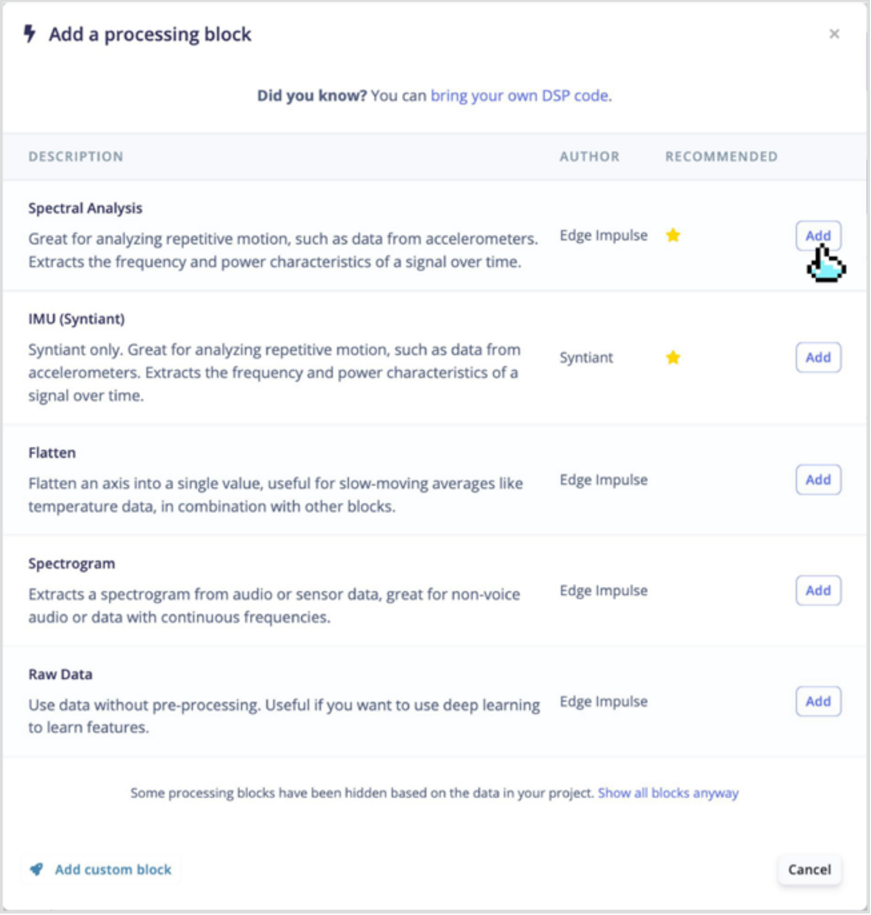
DESCRIPTION	RECOMMENDED	
Time series data Operates on time series sensor data like vibration or audio data. Lets you slice up data into windows.	★	Add
Images Processes discrete images for object detection or classification.		Add

As shown in the figure below, set the **Window Size** to `2000` ms (2 seconds), the **Window Increase** to `80` milliseconds, and the **Frequency** to 51 Hz based on the calculations we made in the data preprocessing section on the Time Series Data block that appears.



2. Adding the processing block: Spectral Analysis

Click the **"Add a Processing Block"** button and select **Spectral Analysis** in the pop-up window as shown below to match our motion analysis task type.



Add a processing block

Did you know? You can bring your own DSP code.

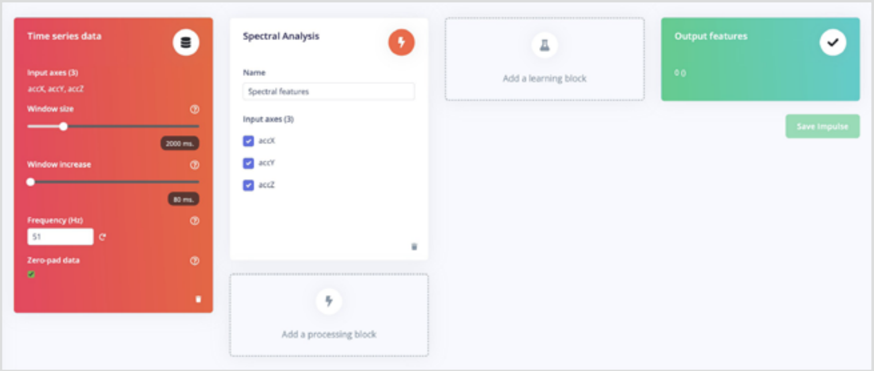
DESCRIPTION	AUTHOR	RECOMMENDED	
Spectral Analysis Great for analyzing repetitive motion, such as data from accelerometers. Extracts the frequency and power characteristics of a signal over time.	Edge Impulse	★	Add
IMU (Syntiant) Syntiant only. Great for analyzing repetitive motion, such as data from accelerometers. Extracts the frequency and power characteristics of a signal over time.	Syntiant	★	Add
Flatten Flatten an axis into a single value, useful for slow-moving averages like temperature data, in combination with other blocks.	Edge Impulse		Add
Spectrogram Extracts a spectrogram from audio or sensor data, great for non-voice audio or data with continuous frequencies.	Edge Impulse		Add
Raw Data Use data without pre-processing. Useful if you want to use deep learning to learn features.	Edge Impulse		Add

Some processing blocks have been hidden based on the data in your project. Show all blocks anyway

Add custom block

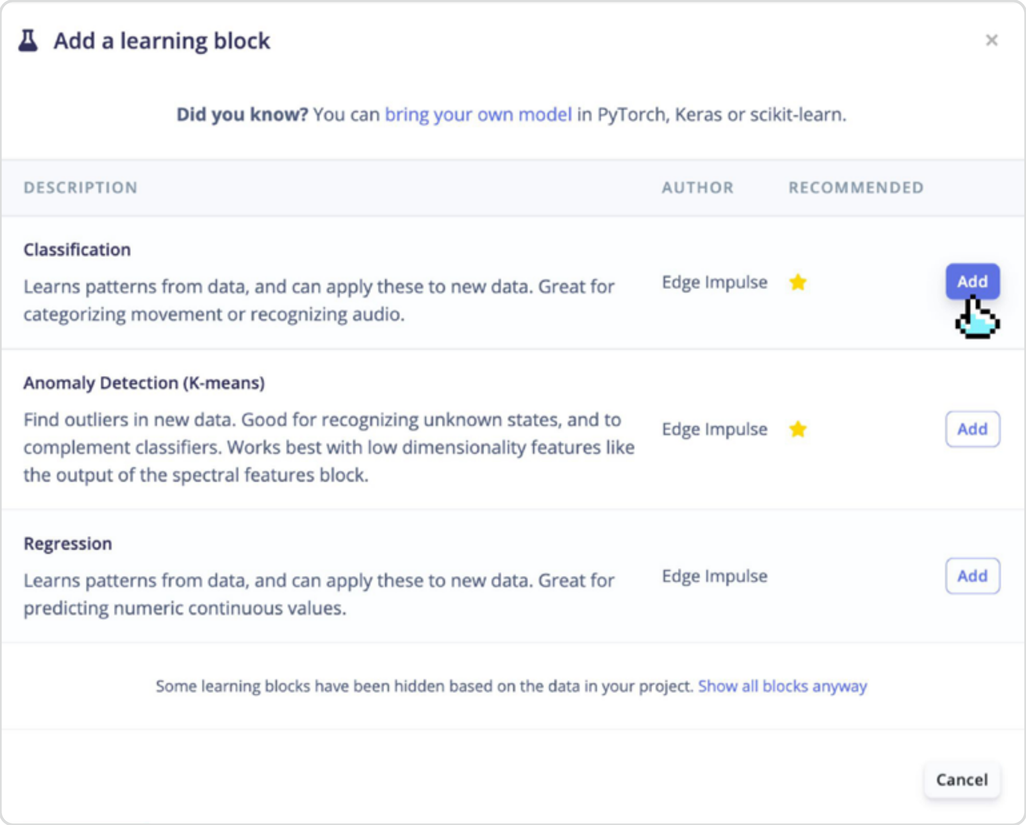
Cancel

The effect after adding the processing block is shown in the figure below.



3. Adding the learning block: Classification

Click the “Add Learning Block” button and select **Classification** in the pop-up window as shown below to match our motion analysis task type.



Add a learning block

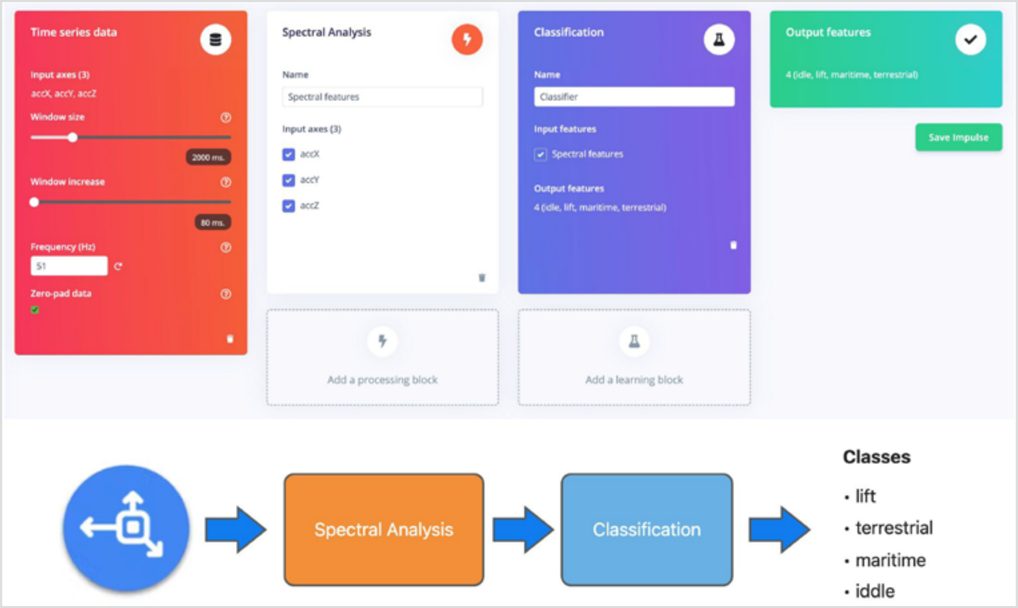
Did you know? You can bring your own model in PyTorch, Keras or scikit-learn.

DESCRIPTION	AUTHOR	RECOMMENDED	
Classification Learns patterns from data, and can apply these to new data. Great for categorizing movement or recognizing audio.	Edge Impulse	★	Add
Anomaly Detection (K-means) Find outliers in new data. Good for recognizing unknown states, and to complement classifiers. Works best with low dimensionality features like the output of the spectral features block.	Edge Impulse	★	Add
Regression Learns patterns from data, and can apply these to new data. Great for predicting numeric continuous values.	Edge Impulse		Add

Some learning blocks have been hidden based on the data in your project. Show all blocks anyway

Cancel

The interface of Impulse design after addition is shown in the figure below, and now the machine learning pipeline has been built.



Time series data

Input axes (3)
accX, accY, accZ

Window size: 2000 ms.

Window increase: 80 ms.

Frequency (Hz): 51

Zero-pad data ☑

Spectral Analysis

Name: Spectral features

Input axes (3)
☑ accX
☑ accY
☑ accZ

Classification

Name: Classifier

Input features
☑ Spectral features

Output features
4 (idle, lift, maritime, terrestrial)

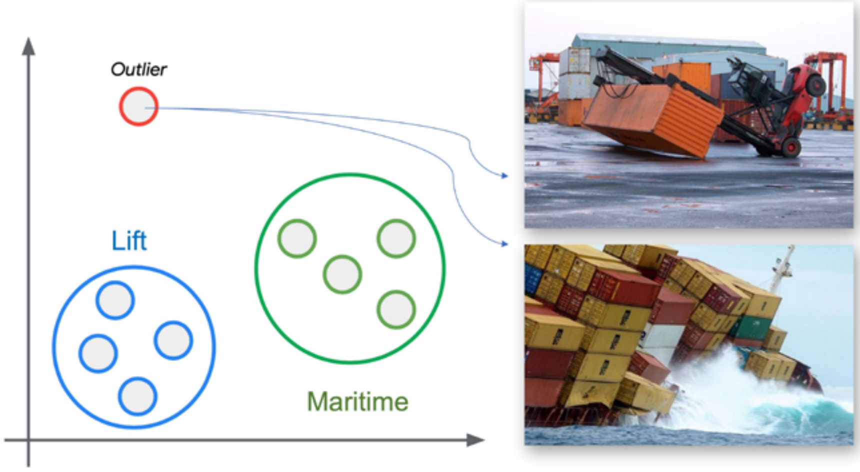
Output features
4 (idle, lift, maritime, terrestrial)

Save Impulse

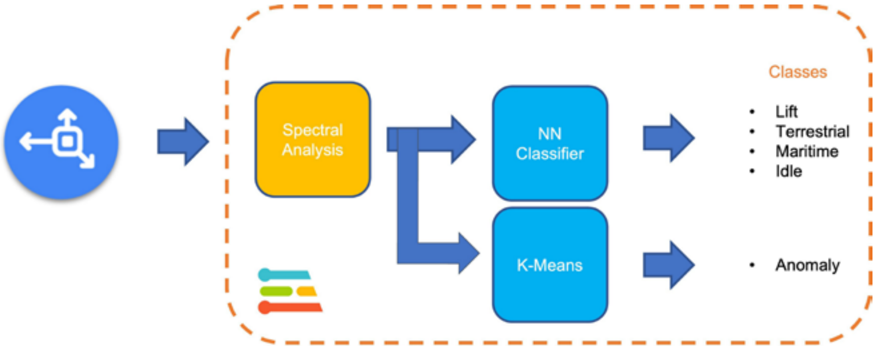
Add a processing block

Add a learning block

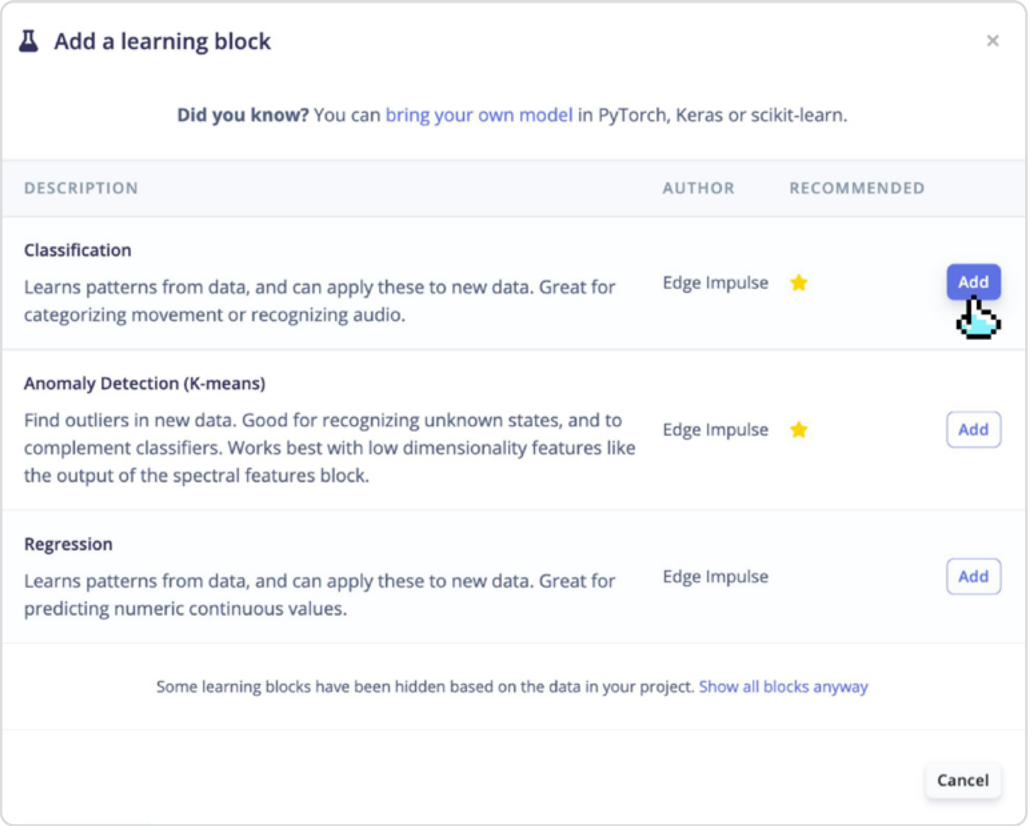
In addition, we can also use a second model - K-means, which can be used for anomaly detection. If we imagine that we can treat our known classes as clusters, then any sample that does not fit into it might be an anomaly (for example, a container falling into the sea when the ship is at sea).



For this, we can use the same input tensor entering the NN classifier as the input to the K-means model:



Click the "Add Learning Block" button again and select **Anomaly Detection (K-means)** in the pop-up window below.



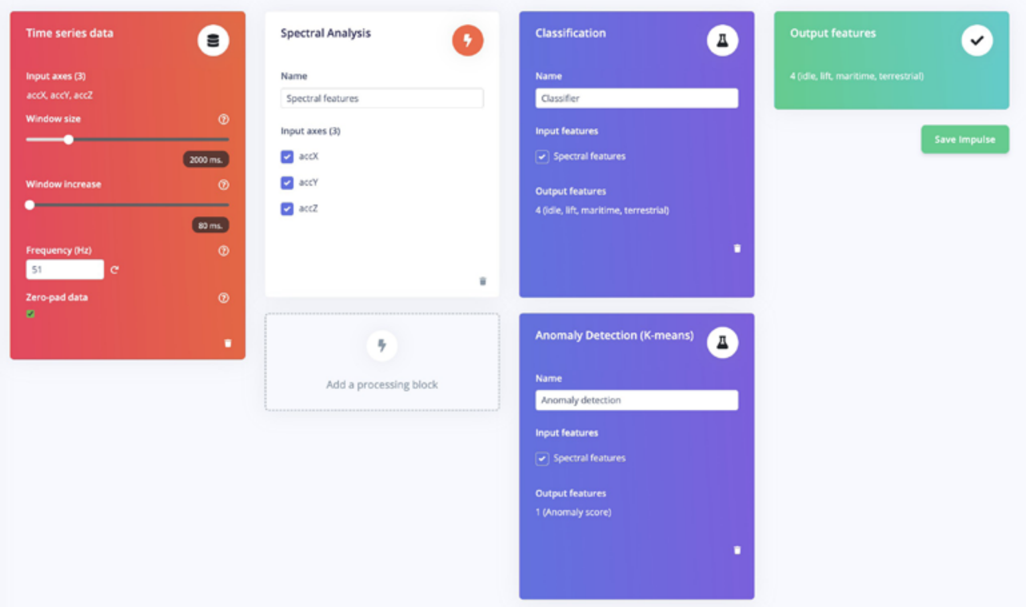
Add a learning block

Did you know? You can bring your own model in PyTorch, Keras or scikit-learn.

DESCRIPTION	AUTHOR	RECOMMENDED	
Classification Learns patterns from data, and can apply these to new data. Great for categorizing movement or recognizing audio.	Edge Impulse	★	Add
Anomaly Detection (K-means) Find outliers in new data. Good for recognizing unknown states, and to complement classifiers. Works best with low dimensionality features like the output of the spectral features block.	Edge Impulse	★	Add
Regression Learns patterns from data, and can apply these to new data. Great for predicting numeric continuous values.	Edge Impulse		Add

Some learning blocks have been hidden based on the data in your project. Show all blocks anyway

The final Impulse design is as shown in the figure below, click the **Save Impulse** button on the far right.


Time series data
Input axes (3)
accX, accY, accZ
Window size: 2000 ms.
Window increase: 80 ms.
Frequency (Hz): 51
Zero-pad data: ☑

Spectral Analysis
Name: Spectral features
Input axes (3)
☑ accX
☑ accY
☑ accZ

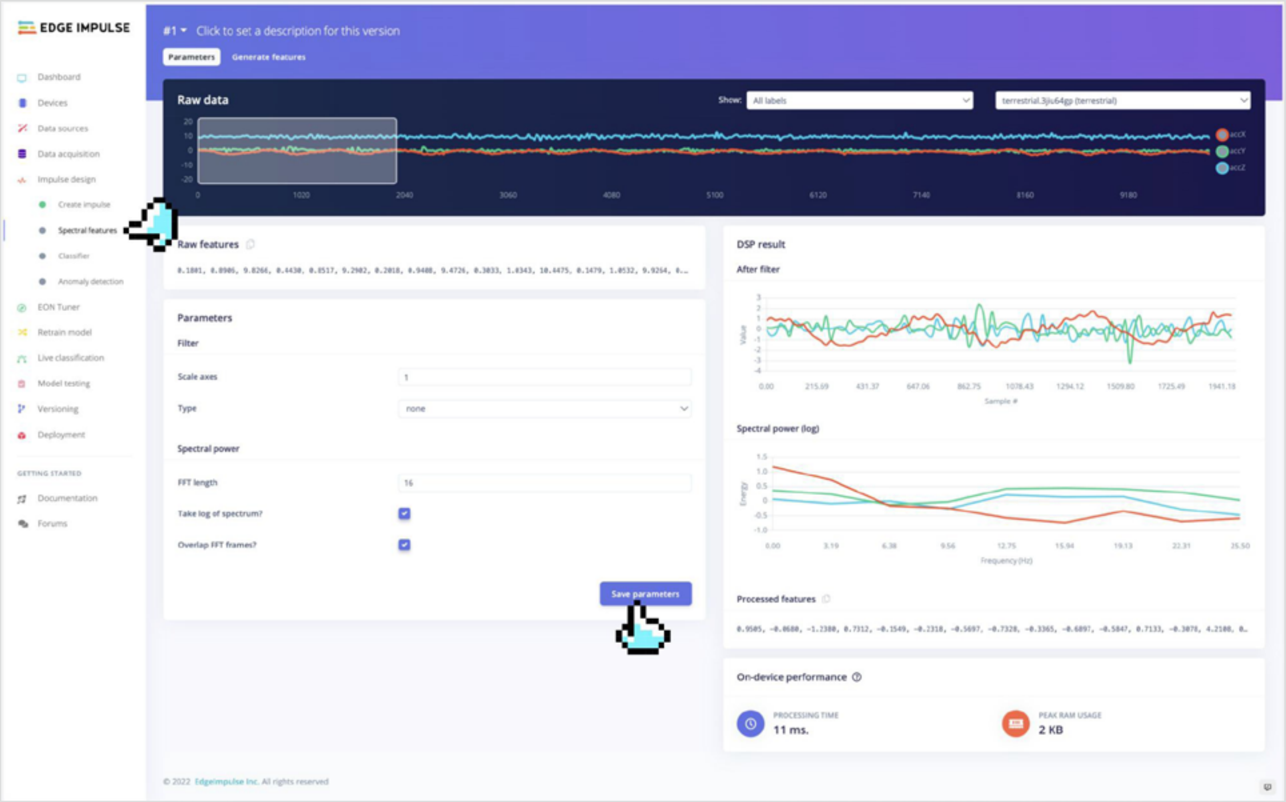
Classification
Name: Classifier
Input features
☑ Spectral features
Output features
4 (idle, lift, maritime, terrestrial)

Anomaly Detection (K-means)
Name: Anomaly detection
Input features
☑ Spectral features
Output features
1 (Anomaly score)

Output features
4 (idle, lift, maritime, terrestrial)

Generating features

At this point in our project, we have defined the pre-processing method and the model designed. Now, it is time to have the job done. First, let's take the raw data (time-series type) and convert it to tabular data. Go to the `Spectral Features` tab, select `Save Parameters`, and at the top menu, select `Generate Features` option and `Generate Features button`:


#1 Click to set a description for this version

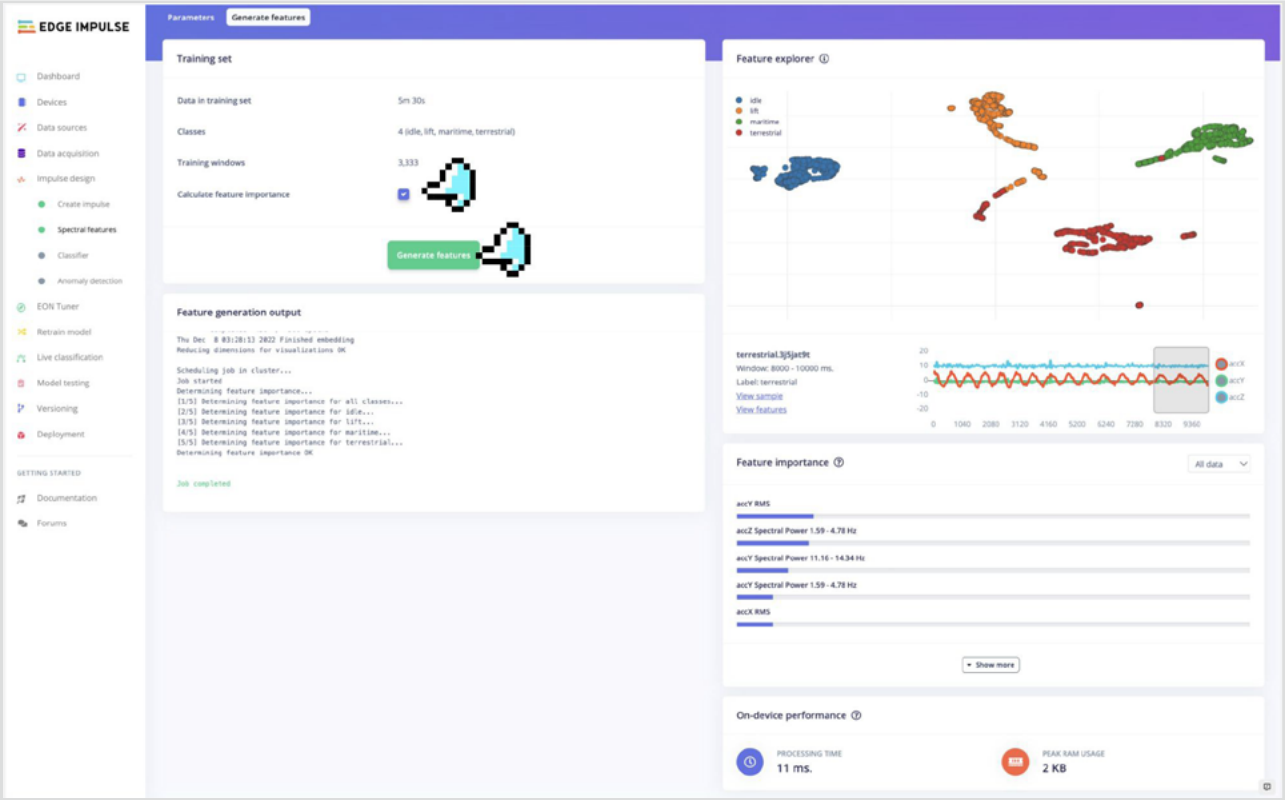
Raw features
0.1801, 0.0906, 9.8266, 0.4430, 0.8517, 9.2982, 0.2018, 0.9408, 9.4726, 0.3033, 1.0343, 10.4475, 0.1479, 1.0532, 9.9264, 0...

Parameters
Filter
Scale axes: 1
Type: none
Spectral power
FFT length: 16
Take log of spectrum? ☑
Overlap FFT frames? ☑

Processed features
0.9505, -0.0680, -1.2380, 0.7312, -0.1549, -0.2318, -0.5697, -0.7328, -0.3365, -0.6897, -0.5847, 0.7133, -0.3078, 4.2108, 0...

On-device performance
PROCESSING TIME: 11 ms.
PEAK RAM USAGE: 2 KB

© 2022 EdgeImpulse Inc. All rights reserved



Training set

Data in training set 5m 30s

Classes 4 (idle, lift, maritime, terrestrial)

Training windows 3,333

Calculate feature importance ☑

Generate features

Feature generation output

```
Thu Dec  8 03:28:13 2022 Finished embedding
Reducing dimensions for visualizations OK

Scheduling job in cluster...
Job started
Determining feature importance...
[1/5] Determining feature importance for all classes...
[2/5] Determining feature importance for idle...
[3/5] Determining feature importance for lift...
[4/5] Determining feature importance for maritime...
[5/5] Determining feature importance for terrestrial...
Determining feature importance OK


Job completed
```

terrestrial.3j5jat9t
Window: 8000 - 10000 ms.
Label: terrestrial
View sample
View features

Feature importance

accY RMS
accZ Spectral Power 1.59 - 4.78 Hz
accY Spectral Power 11.16 - 14.34 Hz
accY Spectral Power 1.59 - 4.78 Hz
accX RMS

On-device performance

PROCESSING TIME 11 ms.

PEAK RAM USAGE 2 KB

Each 2-second window data will be converted into one data point of 63 features. The Feature Explorer will show those data in 2D using UMAP.

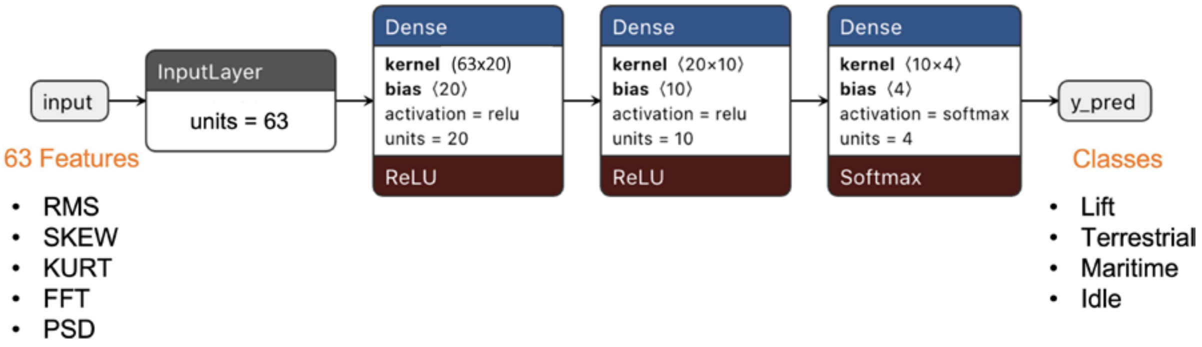
> *Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can be used for visualisation similarly to t-SNE, but also for general non-linear dimension reduction.*

With the visualization, it is possible to verify that the classes present an excellent separation, which indicates that the classifier should work well.

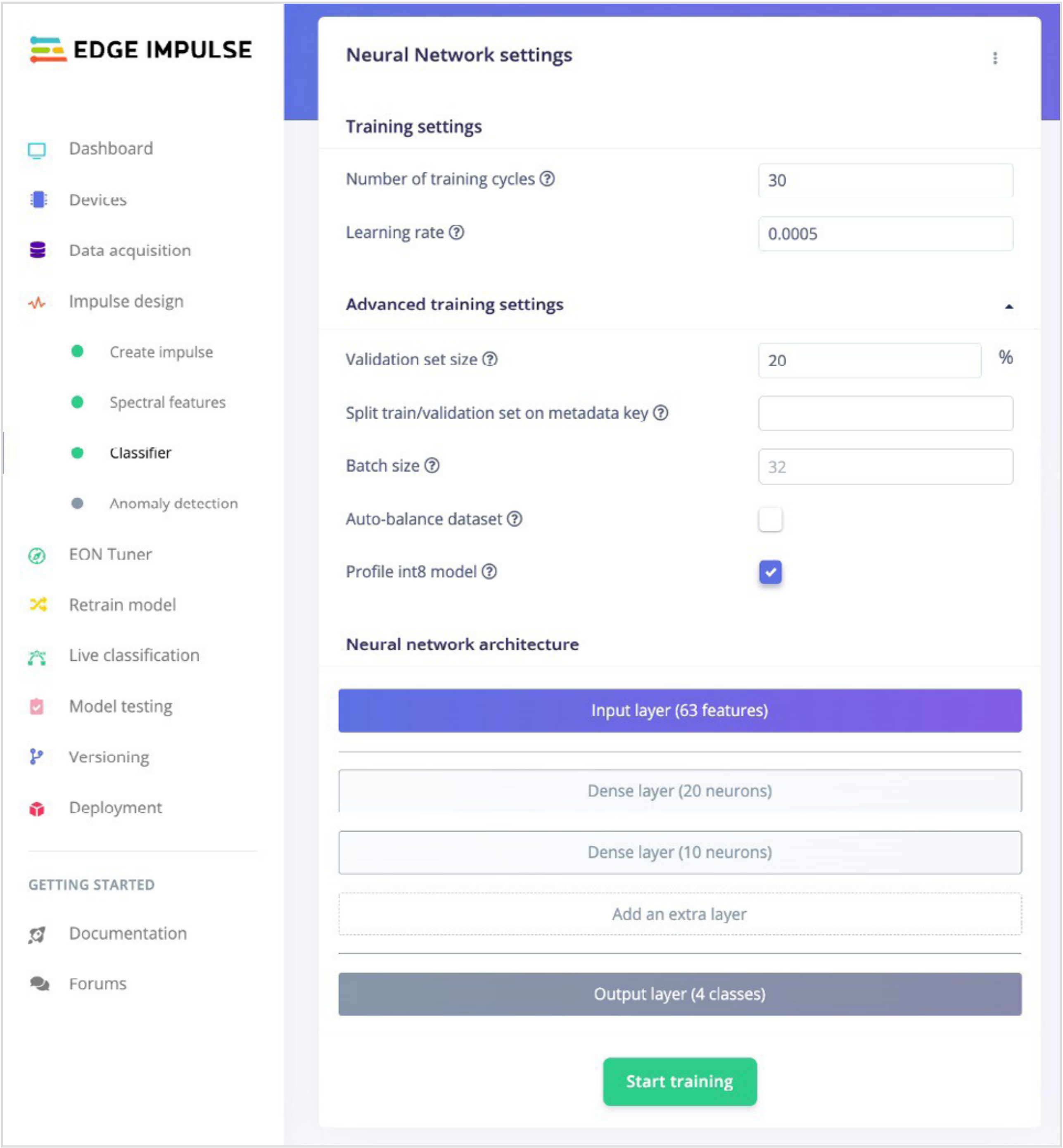
> *Optionally you can analyze how important each one of the features is for one class compared with other classes.*

Training

Our model has four layers, as shown below:



As hyperparameters, we will use a Learning Rate of 0.005 and 20% of data for validation for 30 epochs.



Neural Network settings

Training settings

Number of training cycles: 30

Learning rate: 0.0005

Advanced training settings

Validation set size: 20 %

Split train/validation set on metadata key:

Batch size: 32

Auto-balance dataset: □

Profile int8 model: ☑

Neural network architecture

Input layer (63 features)

Dense layer (20 neurons)

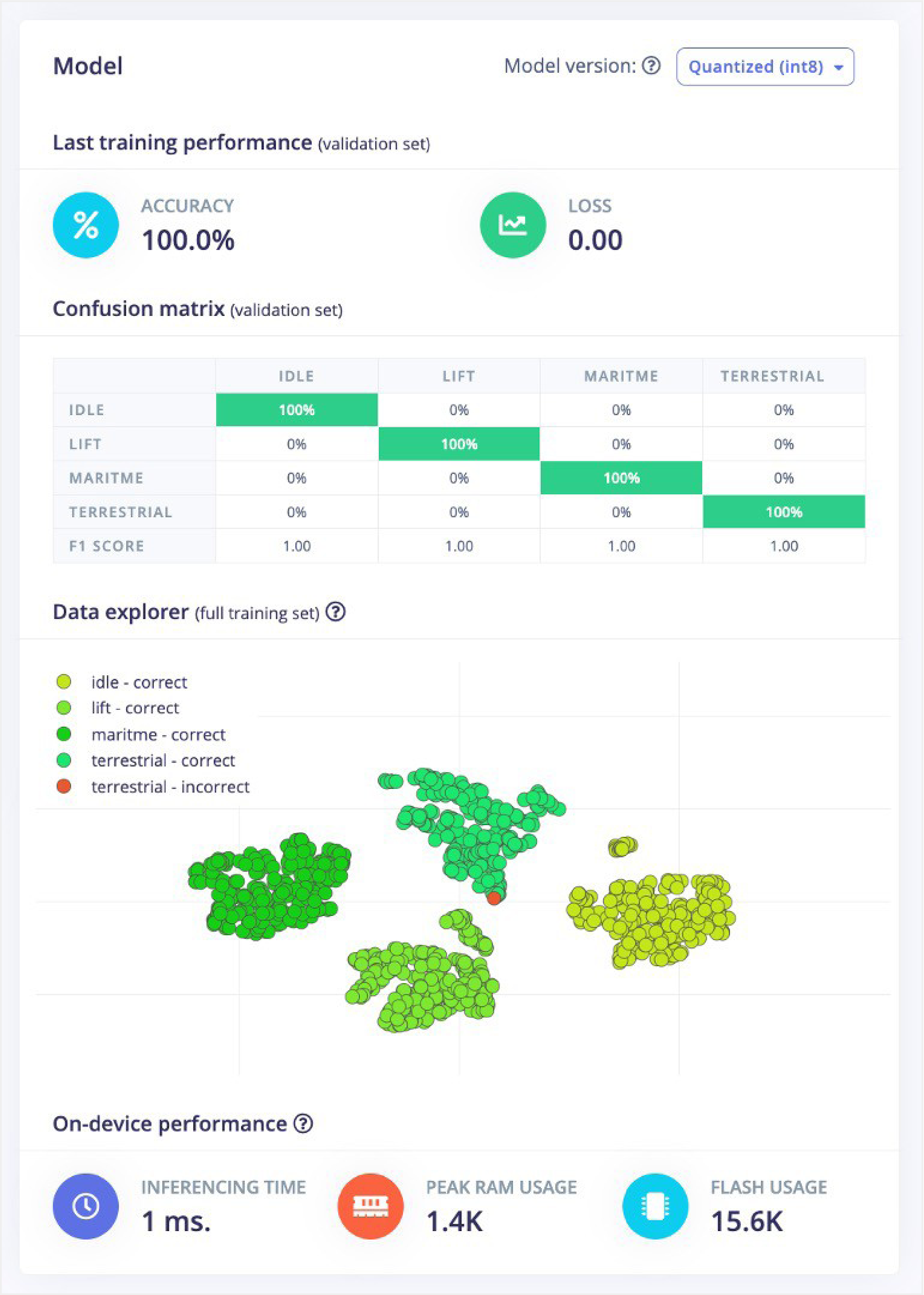
Dense layer (10 neurons)

Add an extra layer

Output layer (4 classes)

Start training

After training, we can see that the accuracy is 100%.



Model

Model version: ⓘ Quantized (int8) ▾

Last training performance (validation set)

ACCURACY
100.0%

LOSS
0.00

Confusion matrix (validation set)

	IDLE	LIFT	MARITME	TERRESTRIAL
IDLE	100%	0%	0%	0%
LIFT	0%	100%	0%	0%
MARITME	0%	0%	100%	0%
TERRESTRIAL	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00

Data explorer (full training set) ⓘ

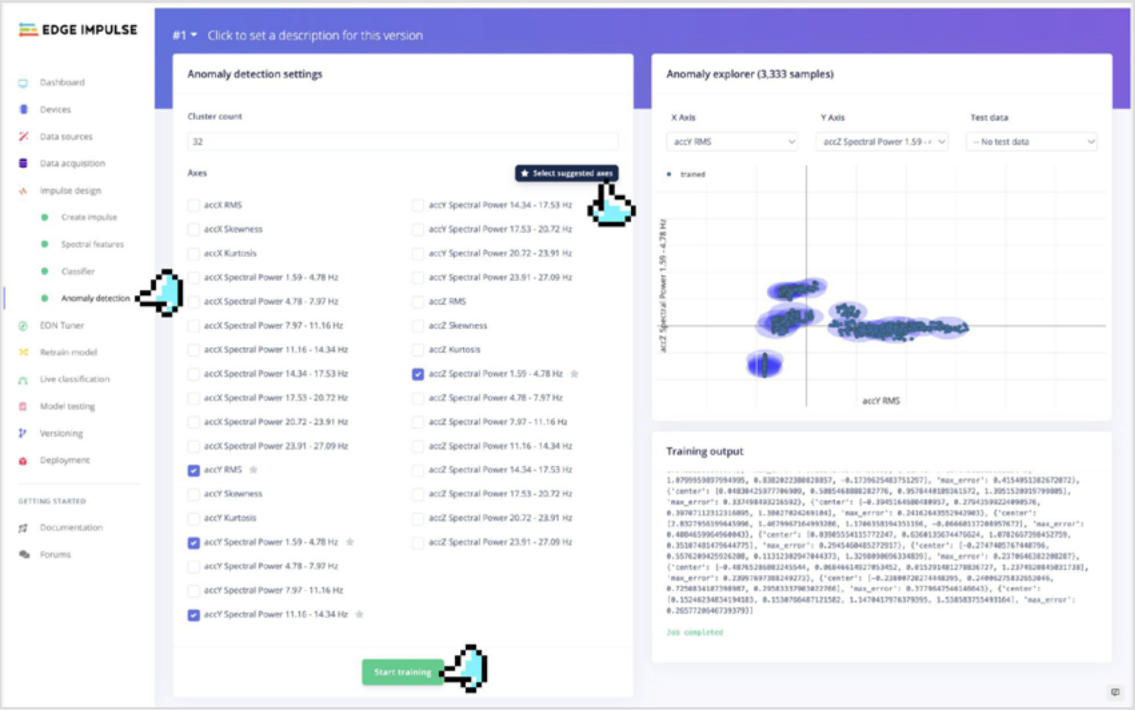
On-device performance ⓘ

INFERENCING TIME
1 ms.

PEAK RAM USAGE
1.4K

FLASH USAGE
15.6K

If a K-means block for anomaly detection has been added during model design, an additional section for `Anomaly Detection` will appear under the `Impulse Design` column on the left, as shown in the image below. Once inside the Anomaly Detection section, click `[Select Suggested Axes]`, and the system will automatically make selections based on previously calculated important features. Then click on the `[Start Training]` button to begin the training. Results will be output in the Anomaly Explorer on the right after completion.



Anomaly detection settings

Cluster count: 32

Axes

□ accX RMS	□ accY Spectral Power 14.34 - 17.53 Hz
□ accX Skewness	□ accY Spectral Power 17.53 - 20.72 Hz
□ accX Kurtosis	□ accY Spectral Power 20.72 - 23.91 Hz
□ accX Spectral Power 1.59 - 4.78 Hz	□ accY Spectral Power 23.91 - 27.09 Hz
□ accX Spectral Power 4.78 - 7.97 Hz	□ accZ RMS
□ accX Spectral Power 7.97 - 11.16 Hz	□ accZ Skewness
□ accX Spectral Power 11.16 - 14.34 Hz	□ accZ Kurtosis
□ accX Spectral Power 14.34 - 17.53 Hz	☑ accZ Spectral Power 1.59 - 4.78 Hz
□ accX Spectral Power 17.53 - 20.72 Hz	□ accZ Spectral Power 4.78 - 7.97 Hz
□ accX Spectral Power 20.72 - 23.91 Hz	□ accZ Spectral Power 7.97 - 11.16 Hz
□ accX Spectral Power 23.91 - 27.09 Hz	□ accZ Spectral Power 11.16 - 14.34 Hz
☑ accY RMS	□ accZ Spectral Power 14.34 - 17.53 Hz
□ accY Skewness	□ accZ Spectral Power 17.53 - 20.72 Hz
□ accY Kurtosis	□ accZ Spectral Power 20.72 - 23.91 Hz
☑ accY Spectral Power 1.59 - 4.78 Hz	□ accZ Spectral Power 23.91 - 27.09 Hz
□ accY Spectral Power 4.78 - 7.97 Hz	
□ accY Spectral Power 7.97 - 11.16 Hz	
☑ accY Spectral Power 11.16 - 14.34 Hz	

X Axis: accY RMS
Y Axis: accZ Spectral Power 1.59 - 4.78 Hz
Test data: -- No test data

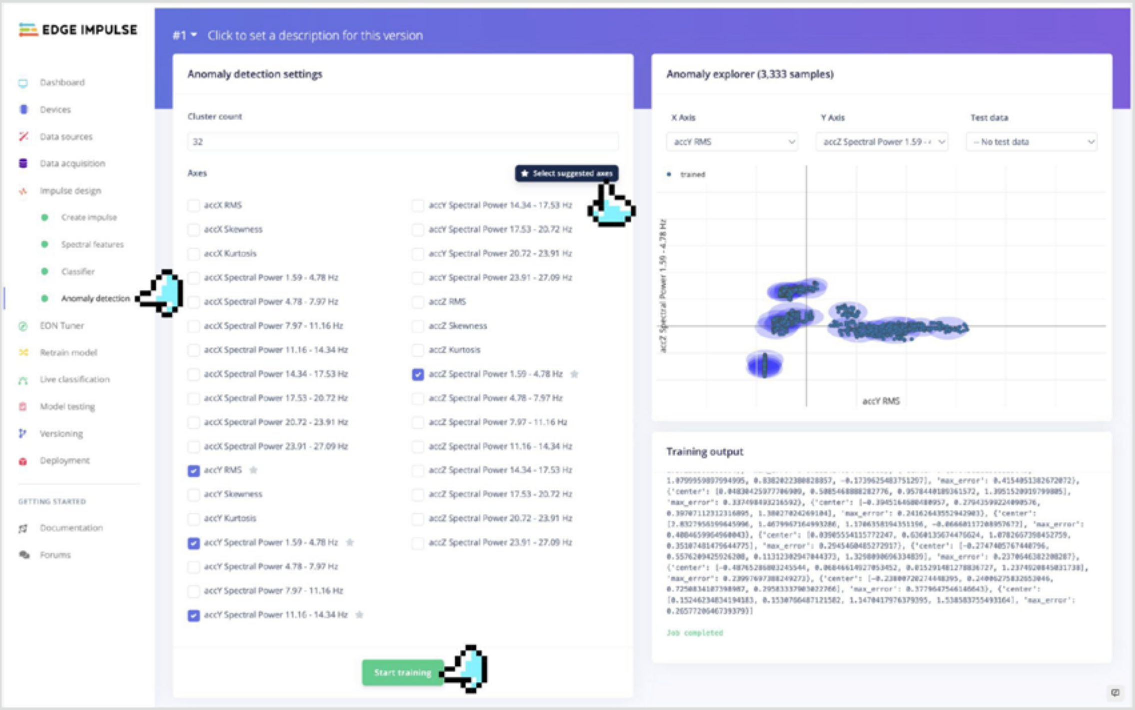
Training output

```
1.0799959837994995, 0.8382022380828857, -0.1739625483751297], 'max_error': 0.4154051382672072},
{'center': [0.0483042597706909, 0.5085468888282776, 0.9578440189361572, 1.3951520919799805],
'max_error': 0.3374984932165923}, {'center': [-0.3945164680480957, 0.27943599224090576,
0.3970711231231689, 1.3802702426910401], 'max_error': 0.24162643552942903}, {'center':
[2.8327956399645996, 1.4679967164993286, 1.1706358194351196, -0.06660115720895767], 'max_error':
0.40046599649600043}, {'center': [0.03905554115772247, 0.6360135674476624, 1.0782657398452759,
0.35107481479644775], 'max_error': 0.2945460485272917}, {'center': [-0.2747405767440796,
0.5576209425926208, 0.11312302947044373, 1.3298009069633483], 'max_error': 0.2370546382208287},
{'center': [-0.48765286083245544, 0.06846614927053452, 0.015291481278836727, 1.2374920845031738],
'max_error': 0.23997697388249273}, {'center': [-0.23800720274448395, 0.24006275832653046,
0.7250834187398987, 0.29583337903022766], 'max_error': 0.3779647546146643}, {'center':
[0.15246234834194183, 0.1530766487121582, 1.3470417976379395, 1.538583755493164], 'max_error':
0.2657720646739379}]

Job completed
```

At this point, we have completed the basic machine learning training process.

Testing

Using the 20% of data set aside during the data collection phase, we can verify the model's performance with unknown data. As shown in the image below, click on the `Model Testing` section on the left side of the Edge Impulse interface. Next to the `[Classify All]` button, there is an icon with three dots, click on it to open the **Set Confidence Thresholds** popup window. Here, you can set confidence thresholds for the results of the two learning blocks. We should define an acceptable threshold for results considered as anomalies. If a result is not 100% (which is often the case) but is within the threshold range, it is still usable.


Anomaly detection settings

Cluster count: 32

Axes

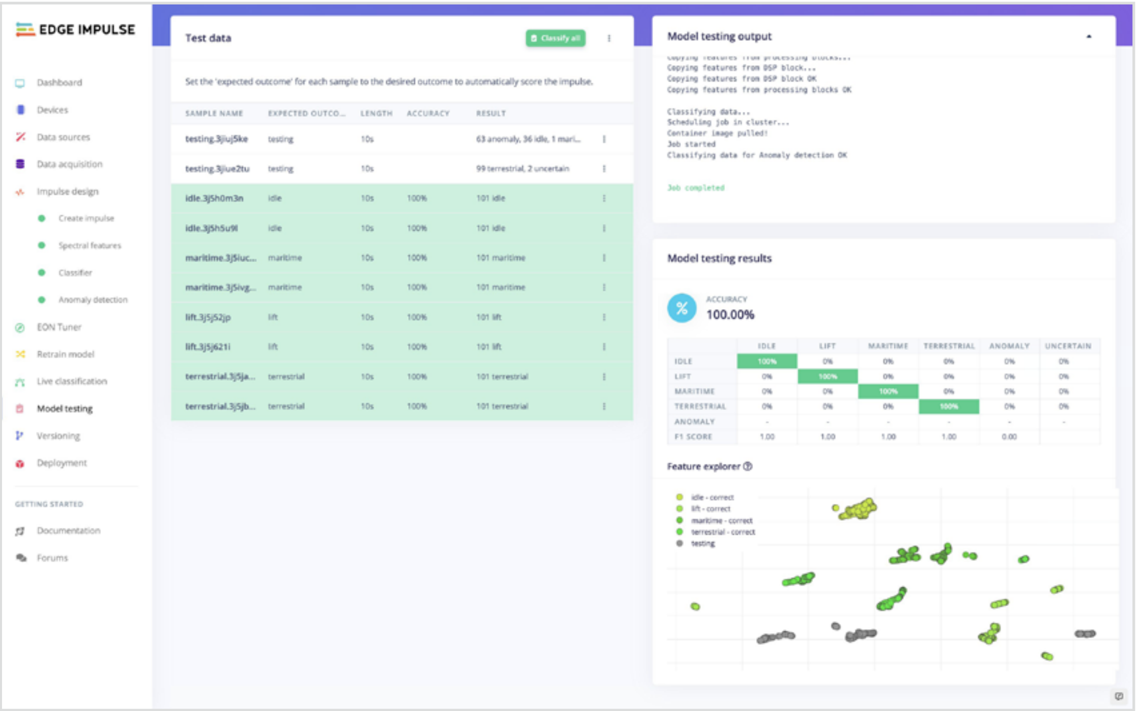
☑ accY RMS
☑ accY Spectral Power 1.59 - 4.78 Hz
☑ accY Spectral Power 11.16 - 14.34 Hz
☑ accZ Spectral Power 1.59 - 4.78 Hz

Anomaly explorer (3,333 samples)

Training output

Job completed

Press the `Classify All` button to start the model testing. The model test results will be displayed upon completion, as shown in the image below.



Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTCO...	LENGTH	ACCURACY	RESULT
testing.3jiuj5ke	testing	10s		63 anomaly, 36 idle, 1 mari...
testing.3jiue2tu	testing	10s		99 terrestrial, 2 uncertain
idle.3j5h0m3n	idle	10s	100%	101 idle
idle.3j5h5u9l	idle	10s	100%	101 idle
maritime.3j5iuc...	maritime	10s	100%	101 maritime
maritime.3j5ivg...	maritime	10s	100%	101 maritime
lift.3j5j52jp	lift	10s	100%	101 lift
lift.3j5j621i	lift	10s	100%	101 lift
terrestrial.3j5ja...	terrestrial	10s	100%	101 terrestrial
terrestrial.3j5jb...	terrestrial	10s	100%	101 terrestrial

Model testing output

```
Copying features from processing blocks...
Copying features from DSP block...
Copying features from DSP block OK
Copying features from processing blocks OK

Classifying data...
Scheduling job in cluster...
Container image pulled!
Job started
Classifying data for Anomaly detection OK


Job completed
```

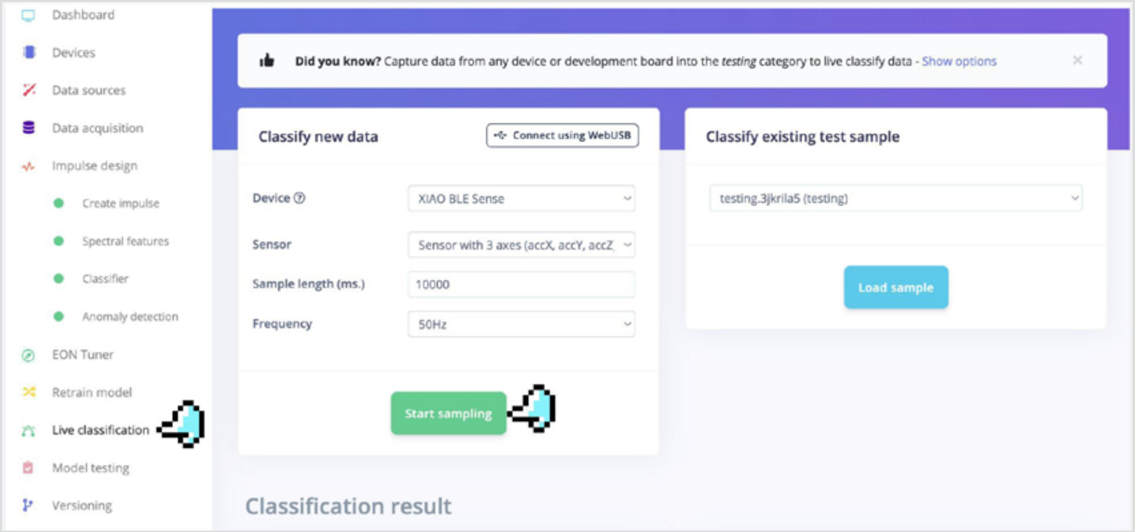
Model testing results

ACCURACY
100.00%

	IDLE	LIFT	MARITIME	TERRESTRIAL	ANOMALY	UNCERTAIN
IDLE	100%	0%	0%	0%	0%	0%
LIFT	0%	100%	0%	0%	0%	0%
MARITIME	0%	0%	100%	0%	0%	0%
TERRESTRIAL	0%	0%	0%	100%	0%	0%
ANOMALY	-	-	-	-	-	-
F1 SCORE	1.00	1.00	1.00	1.00	0.00	

Live Classification

Once the model is obtained, you should use the opportunity to test the Live Classification when your device is still connected to the Edge Impulse Studio. As shown in the image below, click on the `Live Classification` section on the left side of the Edge Impulse interface, then click the `[Start Sampling]` button.



Did you know? Capture data from any device or development board into the *testing* category to live classify data - Show options

Classify new data

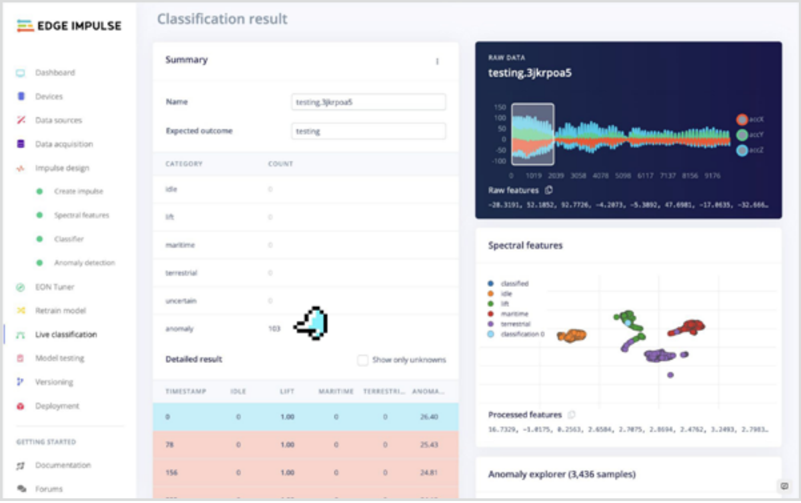
Device	XIAO BLE Sense
Sensor	Sensor with 3 axes (accX, accY, accZ
Sample length (ms.)	10000
Frequency	50Hz

Classify existing test sample

testing.3jkrila5 (testing)

Classification result

At this time, you can, for example, shake the XIAO, the process is the same as the sampling; wait a few seconds, and the classification results will be given. As shown in the image below, I shook the XIAO vigorously, and the model unhesitatingly inferred that the entire process was **anomalous**.



Classification result

Summary

Name: testing.3jkrpoa5

Expected outcome: testing

CATEGORY	COUNT
idle	0
lift	0
maritime	0
terrestrial	0
uncertain	0
anomaly	103

Detailed result

□ Show only unknowns

TIMESTAMP	IDLE	LIFT	MARITIME	TERRESTRI...	ANOMA...
0	0	1.00	0	0	26.40
78	0	1.00	0	0	25.43
156	0	1.00	0	0	24.81

Raw features

-28.3191, 52.1852, 92.7726, -4.2873, -5.3892, 47.6981, -17.8635, -32.666...

Processed features

16.7329, -1.0175, 0.2563, 2.6584, 2.7875, 2.8694, 2.4762, 3.2493, 2.7983...

Anomaly explorer (3,436 samples)

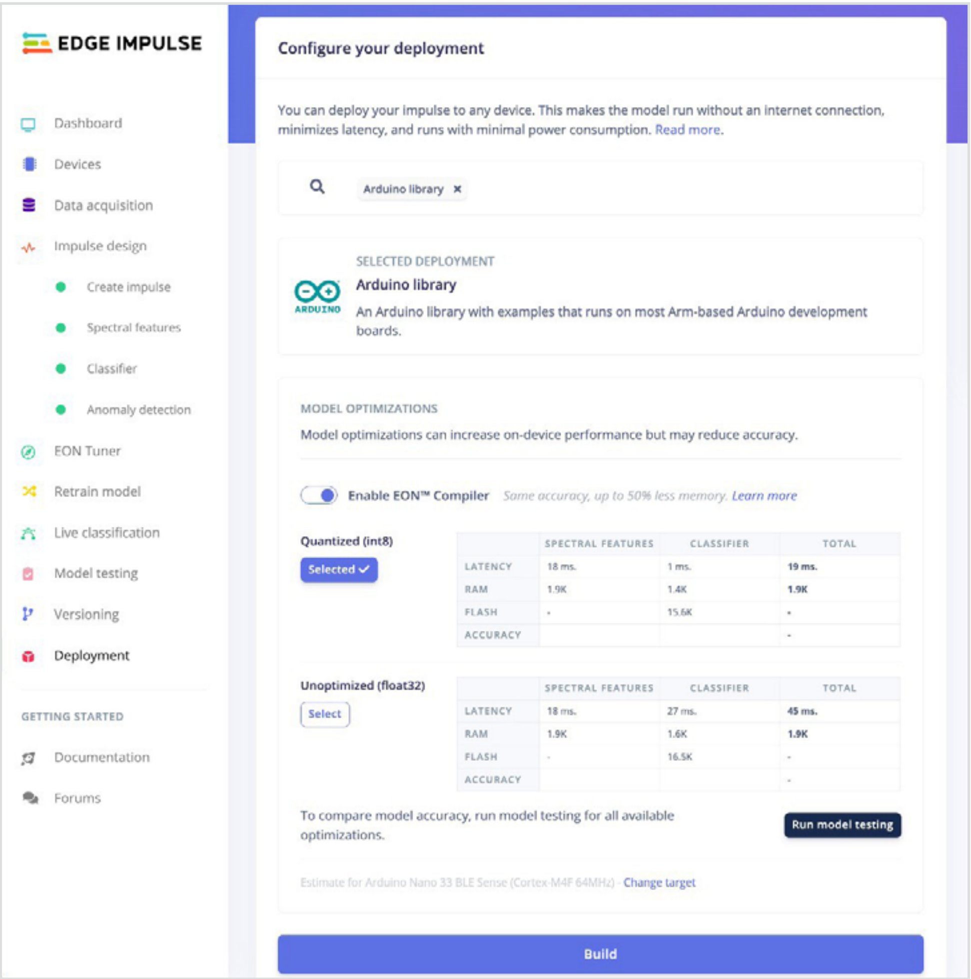
Try now with the same movements used during data capture. The result should match the class used for training.

- Attention -

Here, you will capture real data with your device and upload it to the Edge Impulse Studio, where the trained model will be used for inference (though the model is not in your device).

Deployment

Now it is time for magic˜! The Studio will package all the needed libraries, preprocessing functions, and trained models, downloading them to your computer. You should select the option Arduino Library and at the bottom, select `Quantized (Int8)` and `Build`.



Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

SELECTED DEPLOYMENT

Arduino library

An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

● Enable EON™ Compiler *Same accuracy, up to 50% less memory. Learn more*

Quantized (int8)

	SPECTRAL FEATURES	CLASSIFIER	TOTAL
LATENCY	18 ms.	1 ms.	19 ms.
RAM	1.9K	1.4K	1.9K
FLASH	-	15.6K	-
ACCURACY			-

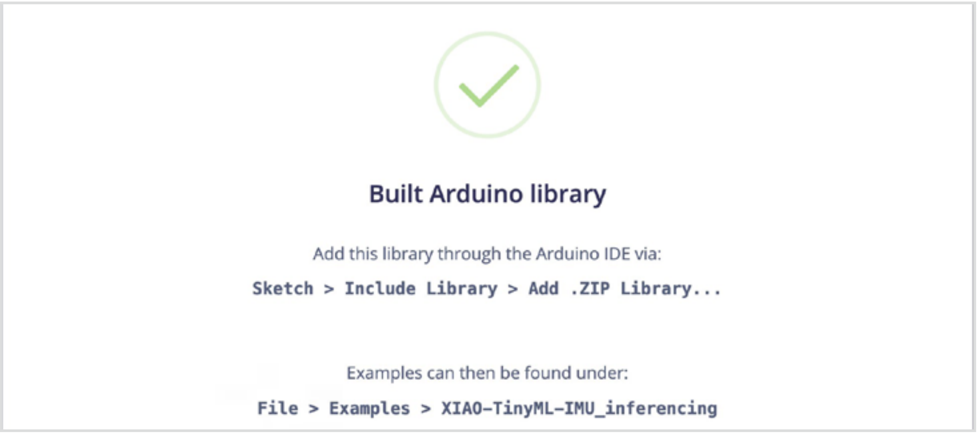
Unoptimized (float32)

	SPECTRAL FEATURES	CLASSIFIER	TOTAL
LATENCY	18 ms.	27 ms.	45 ms.
RAM	1.9K	1.6K	1.9K
FLASH	-	16.5K	-
ACCURACY			-

To compare model accuracy, run model testing for all available optimizations.

Estimate for Arduino Nano 33 BLE Sense (Cortex-M4F 64MHz) - Change target

A Zip file will be created and downloaded to your computer.



Built Arduino library

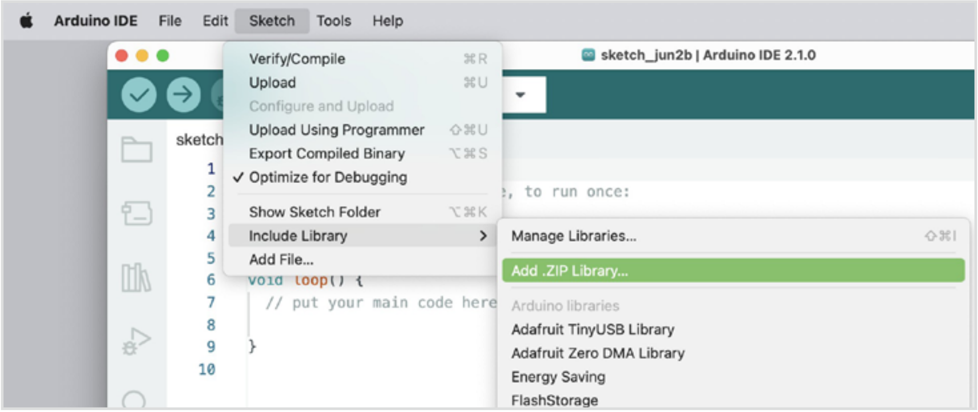
Add this library through the Arduino IDE via:

`Sketch > Include Library > Add .ZIP Library...`

Examples can then be found under:

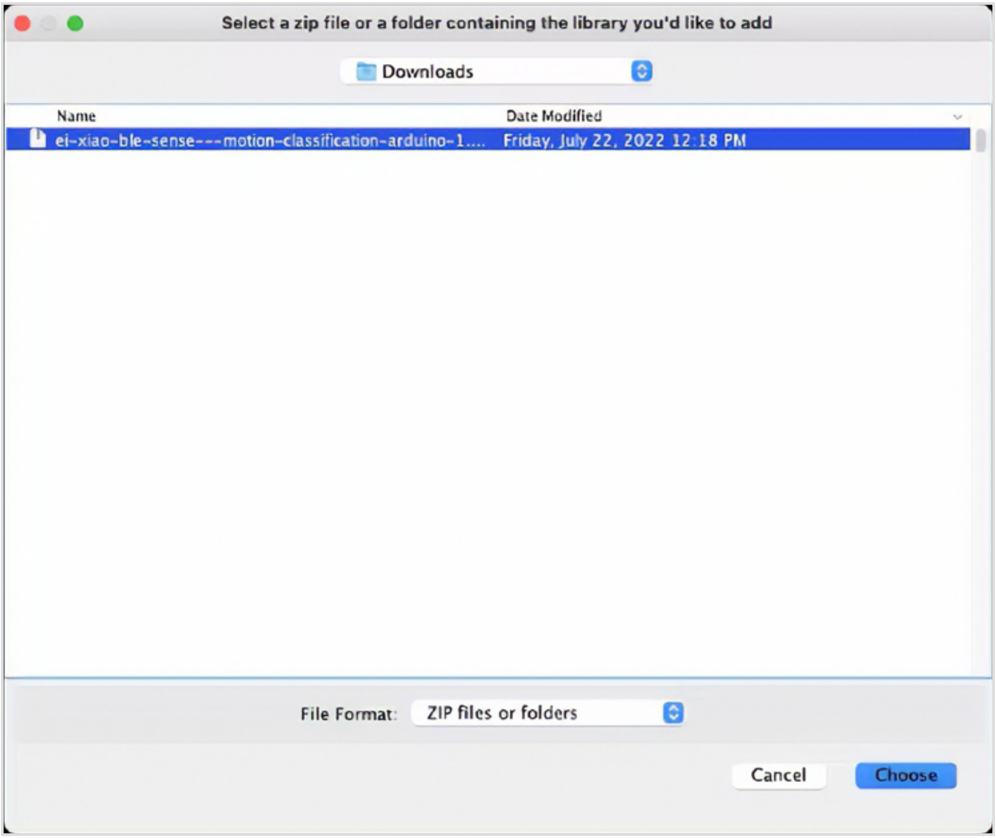
`File > Examples > XIAO-TinyML-IMU_inferencing`

On your Arduino IDE, go to `Sketch` tab and select the option `Add .ZIP Library`.



```
e, to run once:
void loop() {
  // put your main code here
}
```

and Choose the.zip file downloaded by the Studio:



Select a zip file or a folder containing the library you'd like to add

Downloads

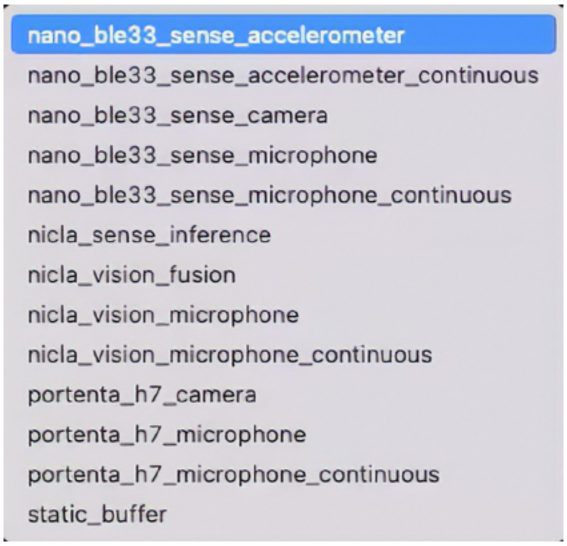
Name	Date Modified
ei-xiao-ble-sense---motion-classification-arduino-1....	Friday, July 22, 2022 12:18 PM

File Format: ZIP files or folders

Inference

Now, it is time for a real test. We will make inferences wholly disconnected from the Studio. Let’s change one of the code examples created when you deploy the Arduino Library.

In your Arduino IDE, go to `File/Examples` tab and look for your project, and on examples, select `nano_ble_sense_accelerometer`:



nano_ble33_sense_accelerometer
nano_ble33_sense_accelerometer_continuous
nano_ble33_sense_camera
nano_ble33_sense_microphone
nano_ble33_sense_microphone_continuous
nicla_sense_inference
nicla_vision_fusion
nicla_vision_microphone
nicla_vision_microphone_continuous
portenta_h7_camera
portenta_h7_microphone
portenta_h7_microphone_continuous
static_buffer

Of course, the Arduino Nano BLE 33 differs from your board, the XIAO, but we can have the code working with only a few changes. For example, at the beginning of the code, you have the library related to Arduino Sense IMU:

```
/* Includes ---------------------------------------------------------------- */
#include <XIAO_BLE_Sense_-_Motion_Classification_inferencing.h>
#include <Arduino_LSM9DS1.h>
```

Change the “includes” portion with the code related to the XIAO nRF52840 Sense IMU:

```
/* Includes ---------------------------------------------------------------- */
#include <XIAO_BLE_Sense_-_Motion_Classification_inferencing.h>
#include “LSM6DS3.h”
#include “Wire.h”

//Create an instance of class LSM6DS3
LSM6DS3 xIMU(I2C_MODE, 0x6A);    //I2C device address 0x6A
```

On the setup function, initiate the IMU using the name that you stated before:

```
if (xIMU.begin() != 0) {
    ei_printf(“Failed to initialize IMU!\r\n”);
}
else {
    ei_printf(“IMU initialized\r\n”);
}
```

At the loop function, the buffers: `buffer[ix]`, `buffer[ix + 1]` and `buffer[ix + 2]` will receive the 3 axis data captured by the accelerometer. On the original code, you have the line:

```
IMU.readAcceleration(buffer[ix], buffer[ix + 1], buffer[ix + 2]);
```

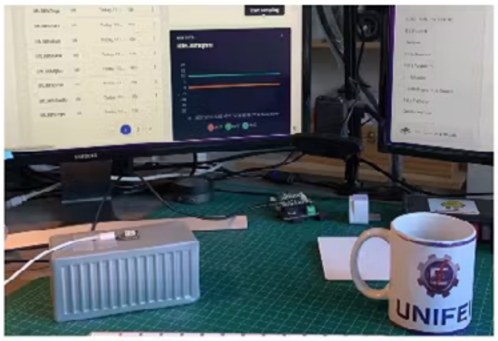
Change it with this block of code:

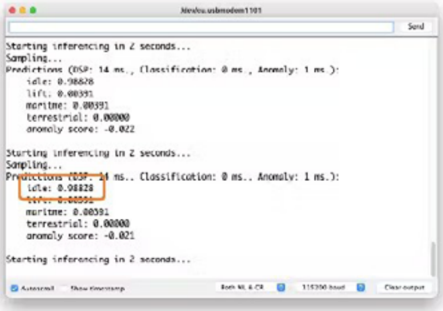
```
buffer[ix]     = xIMU.readFloatAccelX();
buffer[ix + 1] = xIMU.readFloatAccelY();
buffer[ix + 2] = xIMU.readFloatAccelZ();
```

Get this code online https://github.com/Mjrovai/Seeed-XIAO-BLE-Sense/blob/main/XIAO_BLE_Sense_accelerometer/XIAO_BLE_Sense_accelerometer.ino

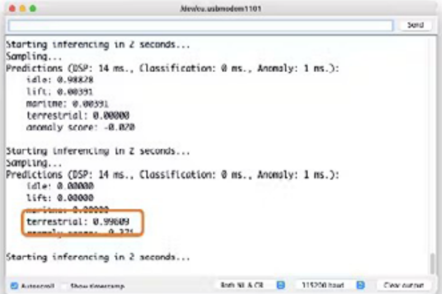
And that is it! You can now upload the code to your device and proceed with the inferences.

You can see the result of the inference of each class on the images:

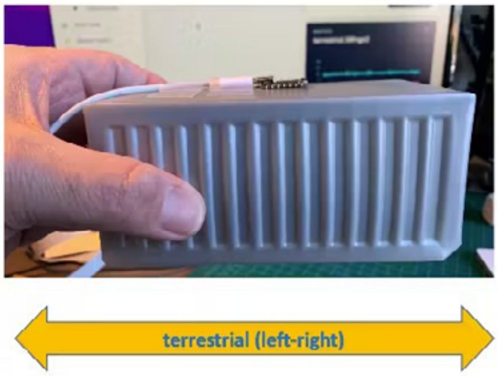


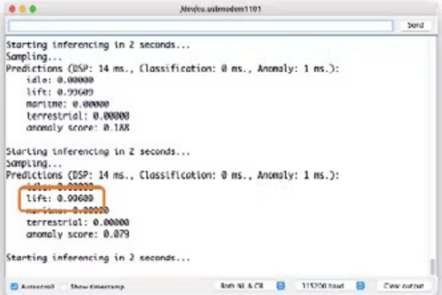


```
Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.98828
    lift: 0.00391
    maritme: 0.00391
    terrestrial: 0.00000
    anomaly score: -0.022

Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.98828
    lift: 0.00391
    maritme: 0.00391
    terrestrial: 0.00000
    anomaly score: -0.021

Starting inferencing in 2 seconds...
```

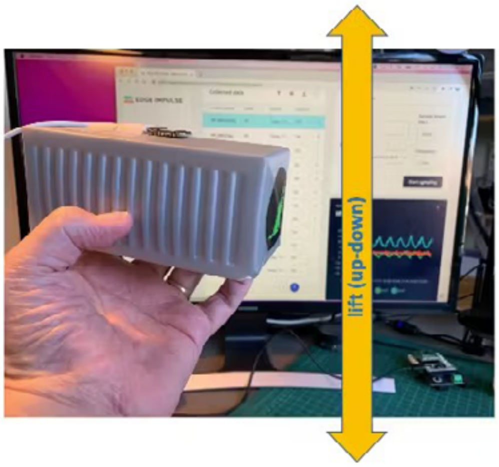


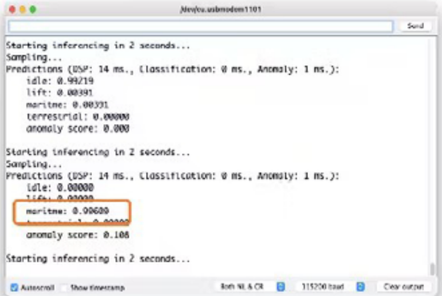
```
Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.98828
    lift: 0.00391
    maritme: 0.00391
    terrestrial: 0.00000
    anomaly score: -0.020

Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.00000
    lift: 0.00000
    maritme: 0.00000
    terrestrial: 0.99609
    anomaly score: -0.371

Starting inferencing in 2 seconds...
```



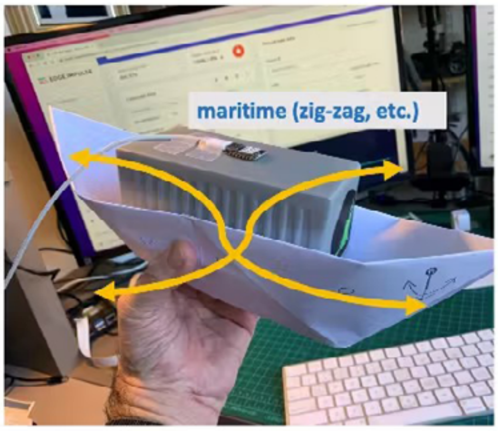


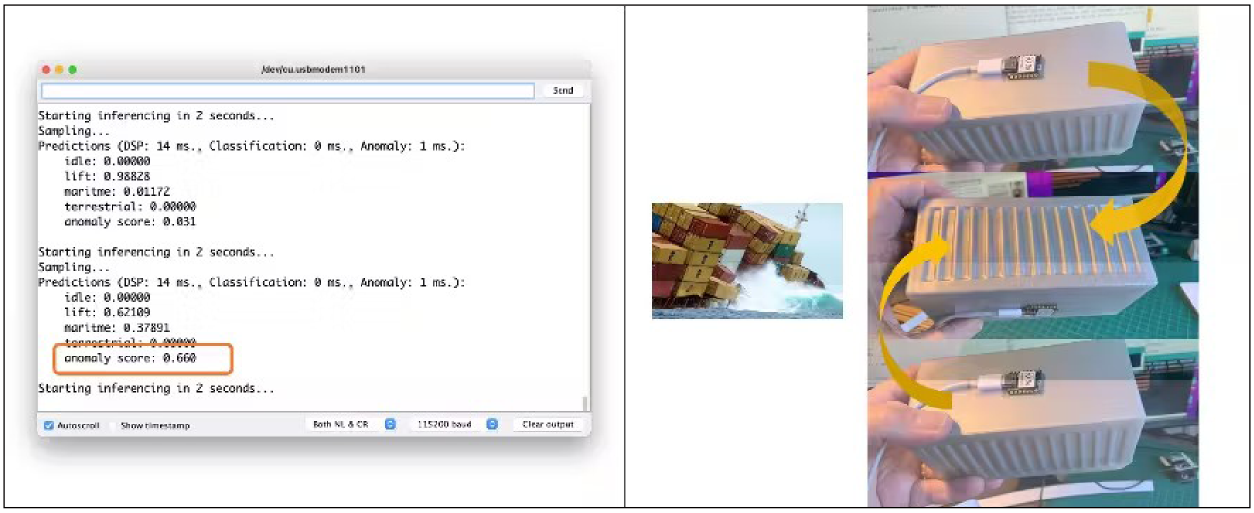
```
Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.00000
    lift: 0.99609
    maritme: 0.00000
    terrestrial: 0.00000
    anomaly score: 0.188

Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.00000
    lift: 0.99609
    maritme: 0.00000
    terrestrial: 0.00000
    anomaly score: 0.079

Starting inferencing in 2 seconds...
```





```
Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.99219
    lift: 0.00391
    maritme: 0.00391
    terrestrial: 0.00000
    anomaly score: 0.000

Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.00000
    lift: 0.00000
    maritme: 0.99609
    terrestrial: 0.00000
    anomaly score: 0.108

Starting inferencing in 2 seconds...
```





```
Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.00000
    lift: 0.98828
    maritme: 0.01172
    terrestrial: 0.00000
    anomaly score: 0.031

Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 14 ms., Classification: 0 ms., Anomaly: 1 ms.):
    idle: 0.00000
    lift: 0.62109
    maritme: 0.37891
    terrestrial: 0.00000
    anomaly score: 0.660

Starting inferencing in 2 seconds...
```

Post-processing

Now that we know the model is working since it detects the movements, we suggest that you modify the code to see the result with the XIAO completely offline (disconnected from the PC and powered by a battery, a power bank, or an independent 5V power supply).

The idea is that if one specific movement is detected, a particular LED could be lit. For example, if terrestrial is detected, the Green LED will light; if maritime, the Red LED will light, if it is a lift, the Blue LED will light; and if no movement is detected (idle), the LEDs will be OFF. You can also add a condition when an anomaly is detected, in this case, for example, a white color can be used (all e LEDs light simultaneously).

Conclusion

The Seeed Studio XIAO nRF52840 Sense is a giant tiny device! It is powerful, trustworthy, not expensive, low power, and has suitable sensors to be used on the most common embedded machine learning applications. Even though Edge Impulse does not officially support XIAO nRF52840 Sense, we also realized that it could be easily connected with the Studio.

> *On the GitHub repository, you will find the last version of the codes: Seeed-XIAO-BLE-Sense.*

The applications for motion classification and anomaly detection are extensive, and the XIAO is well-suited for scenarios where low power consumption and edge processing are advantageous. Its small form factor and efficiency in processing make it an ideal choice for deploying portable and remote applications where real-time processing is crucial and connectivity may be limited.

Case Applications

Before we finish, consider that Movement Classification and Object Detection can be utilized in many applications across various domains. Here are some of the potential applications:

Industrial and Manufacturing

- **Predictive Maintenance:** Detecting anomalies in machinery motion to predict failures before they occur.
- **Quality Control:** Monitoring the motion of assembly lines or robotic arms for precision assessment and deviation detection from the standard motion pattern.
- **Warehouse Logistics:** Managing and tracking the movement of goods with automated systems that classify different types of motion and detect anomalies in handling.

Healthcare

- **Patient Monitoring:** Detecting falls or abnormal movements in the elderly or those with mobility issues.
- **Rehabilitation:** Monitoring the progress of patients recovering from injuries by classifying motion patterns during physical therapy sessions.
- **Activity Recognition:** Classifying types of physical activity for fitness applications or patient monitoring.

Consumer Electronics

- **Gesture Control:** Interpreting specific motions to control devices, such as turning on lights with a hand wave.
- **Gaming:** Enhancing gaming experiences with motion-controlled inputs.

Transportation and Logistics

- **Vehicle Telematics:** Monitoring vehicle motion for unusual behavior such as hard braking, sharp turns, or accidents.
- **Cargo Monitoring:** Ensuring the integrity of goods during transport by detecting unusual movements that could indicate tampering or mishandling.

Smart Cities and Infrastructure

- **Structural Health Monitoring:** Detecting vibrations or movements within structures that could indicate potential failures or maintenance needs.
- **Traffic Management:** Analyzing the flow of pedestrians or vehicles to improve urban mobility and safety.

Security and Surveillance

- **Intruder Detection:** Detecting motion patterns typical of unauthorized access or other security breaches.
- **Wildlife Monitoring:** Detecting poachers or abnormal animal movements in protected areas.

Agriculture

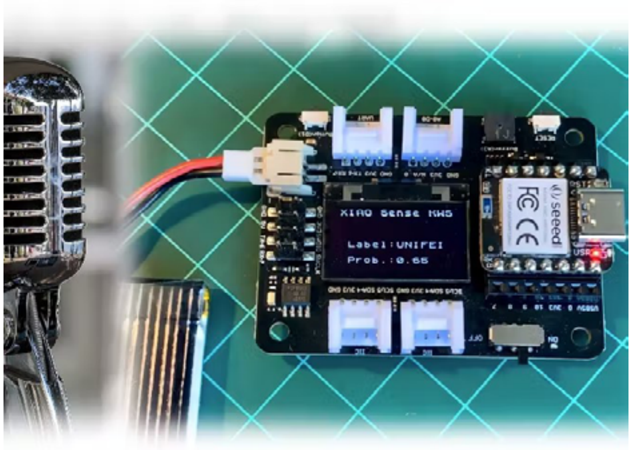
- **Equipment Monitoring:** Tracking the performance and usage of agricultural machinery.
- **Animal Behavior Analysis:** Monitoring livestock movements to detect behaviors indicating health issues or stress.

Environmental Monitoring

- **Seismic Activity:** Detecting irregular motion patterns that precede earthquakes or other geologically relevant events.
- **Oceanography:** Studying wave patterns or marine movements for research and safety purposes.

4.3
Sound Classification (KWS)

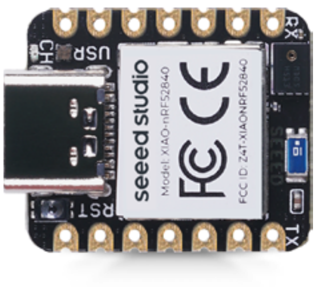
In this section, we continue our exploration of Machine Learning on Seeed Studio XIAO nRF52840 Sense (also called XIAO BLE Sense), now classifying sound waves.



4.3.1 Things used in this project

Hardware components

- Seeed Studio XIAO nRF52840 Sense × 1
- Seeed Studio Seeeduino XIAO Expansion board × 1



Software apps and online services

- Arduino IDE
- Edge Impulse Studio

4.3.2 Introduction

In the last section, **Anomaly Detection & Motion Classification**, we explored Embedded Machine Learning, or simply TinyML, running on the Seeed XIAO nRF52840 Sense. Besides installing and testing the device, we explored motion classification using actual data signals from its onboard accelerometer. This new project will use the same XIAO nRF52840 Sense to classify sound, explicitly working as "Key Word Spotting" (KWS). A KWS is a typical TinyML application and an essential part of a voice assistant.

> *But how does a voice assistant work?*

To start, it is essential to realize that Voice Assistants on the market, like Google Home or Amazon Echo-Dot, only react to humans when they are "waked up" by particular keywords such as " Hey Google" on the first one and "Alexa" on the second.

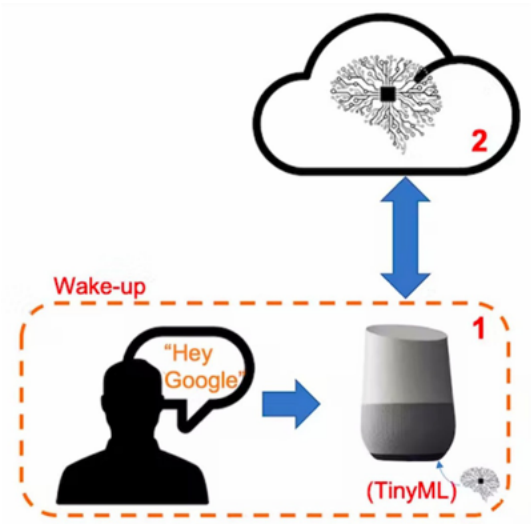
In other words, recognizing voice commands is based on a multi-stage model or Cascade Detection.

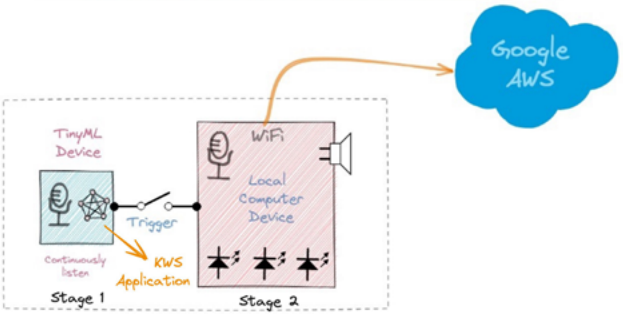
Stage 1: A smaller microprocessor inside the Echo Dot or Google Home continuously listens to the sound, waiting for the keyword to be spotted. For such detection, a TinyML model at the edge is used (KWS application).

Stage 2: Only when triggered by the KWS application on Stage 1 is the data sent to the cloud and processed on a larger model.

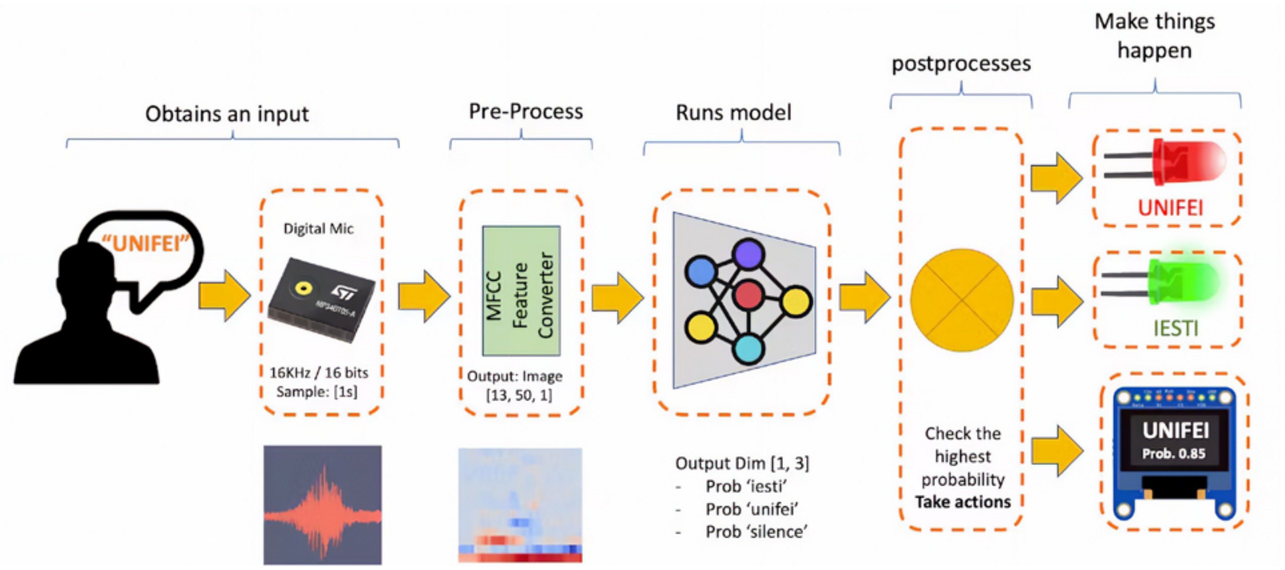
The video below shows an example of a Google Assistant being programmed on a Raspberry Pi (Stage 2), with an Arduino Nano 33 BLE as the tinyML device (Stage 1): https://youtu.be/e_OPgcnsyvM

To explore the above Google Assistant project, please see the tutorial: Building an Intelligent Voice Assistant From Scratch.





4.3.3 The KWS Project



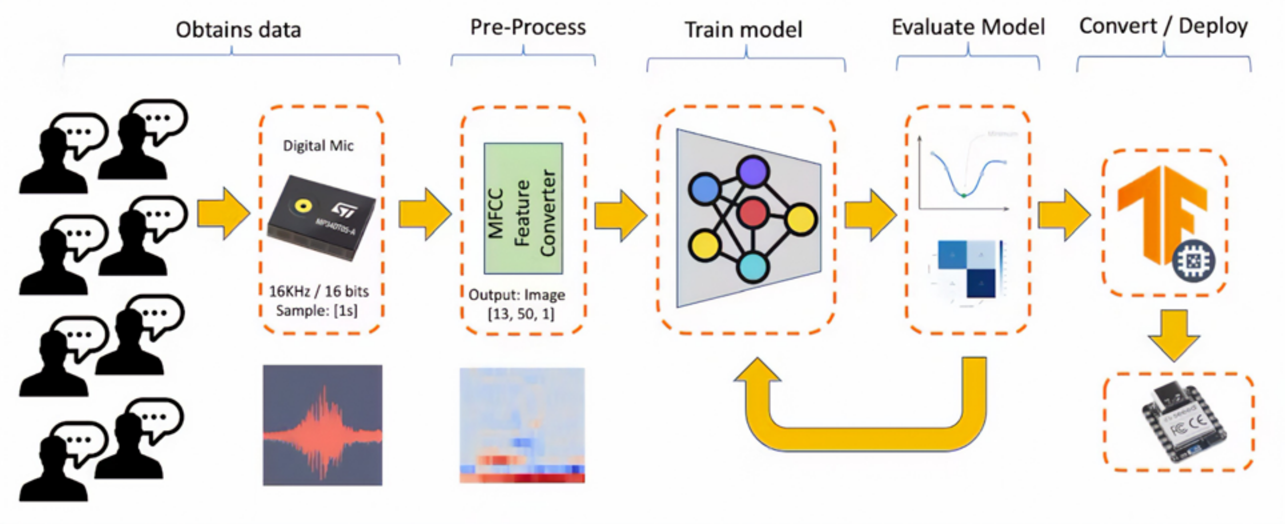
Our KWS application will recognize three classes of sound:

- Keyword 1: **UNIFEI**
- Keyword 2: **IESTI**
- "**SILENCE**" (no keywords spoken, only background noise is present)

Optionally, for real-world projects, it is advised to include different words than keywords 1 and 2 in the class "Silence" (or Background) or even create an extra class with such words (for example a class "others").

The Machine Learning Workflow

The main component of the KWS application is its model. So, we must train such a model with our specific keywords:

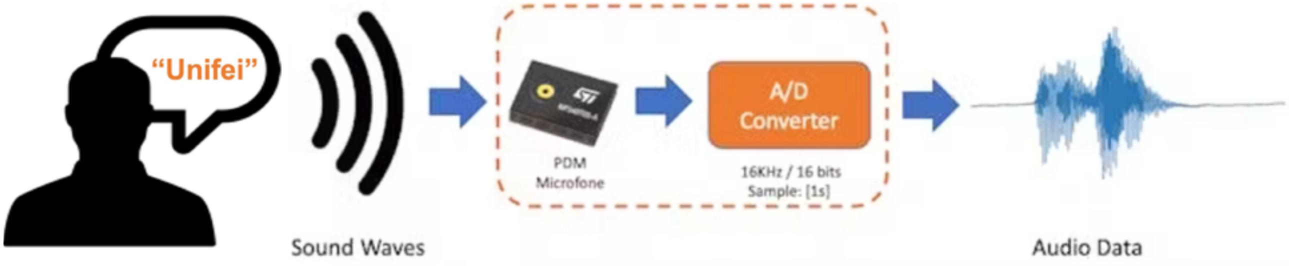


Dataset

The critical component of Machine Learning Workflow is the dataset. Once we have decided on specific keywords (UNIFEI and IESTI), all datasets should be created from zero. When working with accelerometers, creating a dataset with data captured by the same type of sensor was essential. In the case of sound, it is different because of what we will classify as audio data.

> *The critical difference between sound and audio is the type of energy. Sound is mechanical perturbation (longitudinal sound waves) that propagate through a medium, causing variations of pressure in it. Audio is an electrical (analog or digital) signal representing sound.*

The sound waves should be converted to audio data when we speak a keyword. The conversion should be done by sampling the signal generated by the microphone in 16KHz with a 16-bit depth.



So, any device that can generate audio data with this basic specification (16Khz/16bits) will work fine. As a device, we can use the proper XIAO nRF52840 Sense, a computer, or even your mobile phone.

Capturing online Audio Data with Edge Impulse and a smartphone

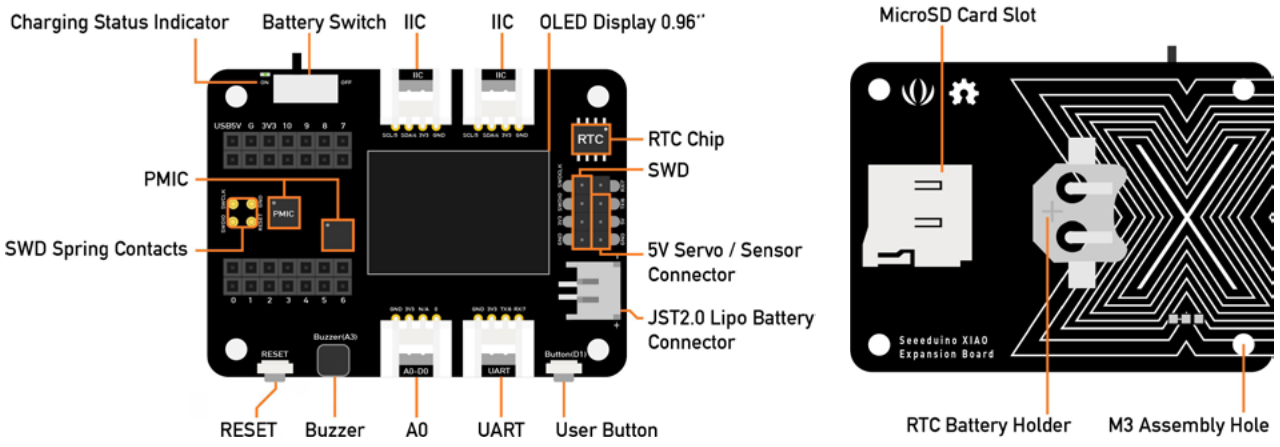
In the TinyML Made Easy: Anomaly Detection & Motion Classification section, we learned how to install and test our device using the Arduino IDE and connect it to Edge Impulse Studio for data capturing. For that, we use the EI CLI function Data Forwarder, but according to Jan Jongboom, Edge Impulse CTO, audio goes too fast for the data forwarder to be captured. If you have PCM data already, then turning it into a WAV file and uploading it with the uploader is the easiest. With accelerometers, our sample frequency was around 50Hz, with audio being 16KHz.

So, we can not connect the XIAO directly to the Studio. But we can capture sound using any smartphone connected to the Studio online.

> *We will not explore this option here, but you can easily follow the EI documentation and tutorial.*

Capturing Audio Data with the XIAO nRF52840 Sense

The easiest way to capture audio and save it locally as a .wav file is using an expansion board for the XIAO family of devices, the Seeed Studio XIAO Expansion board.

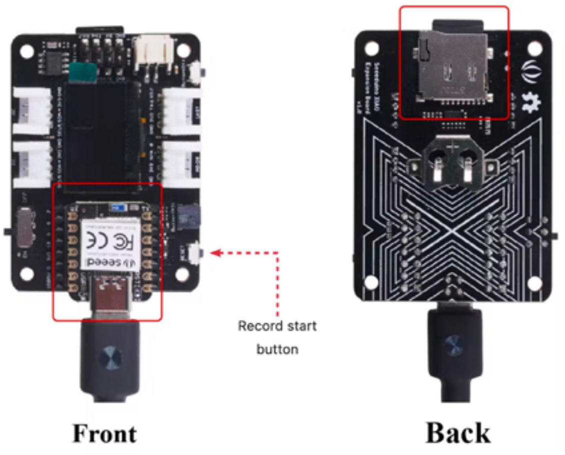


This expansion board enables the building of prototypes and projects easily and quickly, using its rich peripherals such as OLED Display, SD Card interface, RTC, passive buzzer, RESET/User button, 5V servo connector, and multiple data interfaces.

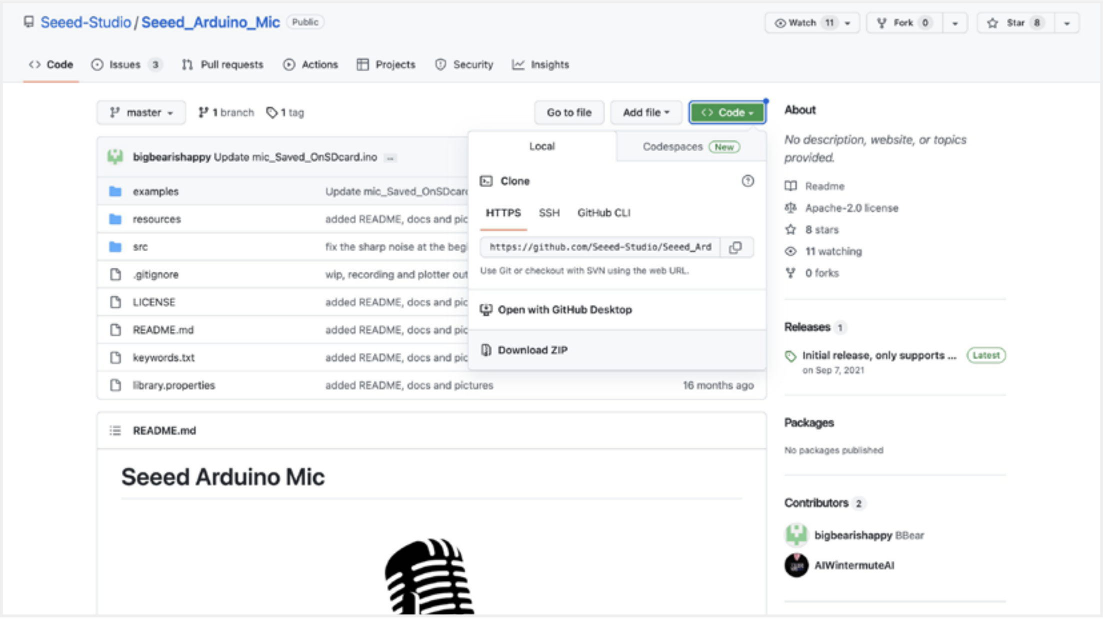
This project will focus on classifying keywords, and the MicroSD card available on the device will be very important in helping us with data capture.

Saving recorded audio from the microphone on an SD card

Connect the XIAO nRF52840 Sense on the Expansion Board and insert an SD card into the SD card slot at the back. > The SD Card should be pre-formated as FAT or exFAT.



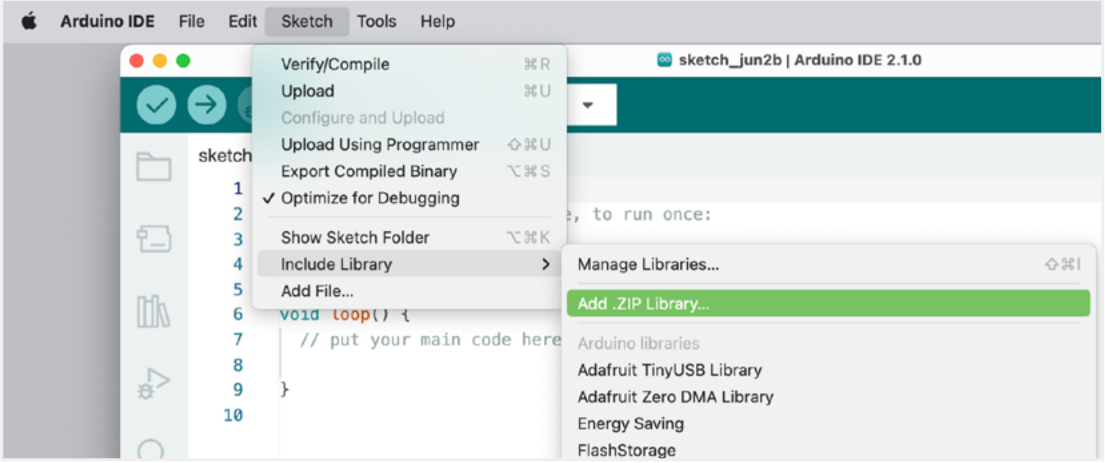
Next, download the Seeed_Arduino_FS Library as a zip file:



bigbearishappy Update mic_Saved_OnSDcard.ino	
examples	Update mic_Saved_OnSDcard
resources	added README, docs and pic
src	fix the sharp noise at the beg
.gitignore	wip, recording and plotter out
LICENSE	added README, docs and pic
README.md	added README, docs and pic
keywords.txt	added README, docs and pic
library.properties	added README, docs and pictures 16 months ago

README.md

And install the downloaded library: `Seeed_Arduino_Mic-master.zip` on your Arduino IDE: `Sketch -> Include Library -> Add .ZIP Library...`



```
e, to run once:

void loop() {
  // put your main code here

}
```

Next, navigate to `File > Examples > Seeed Arduino Mic > mic_Saved_OnSDcard` to open the sketch: **mic_Saved_OnSDcard**.

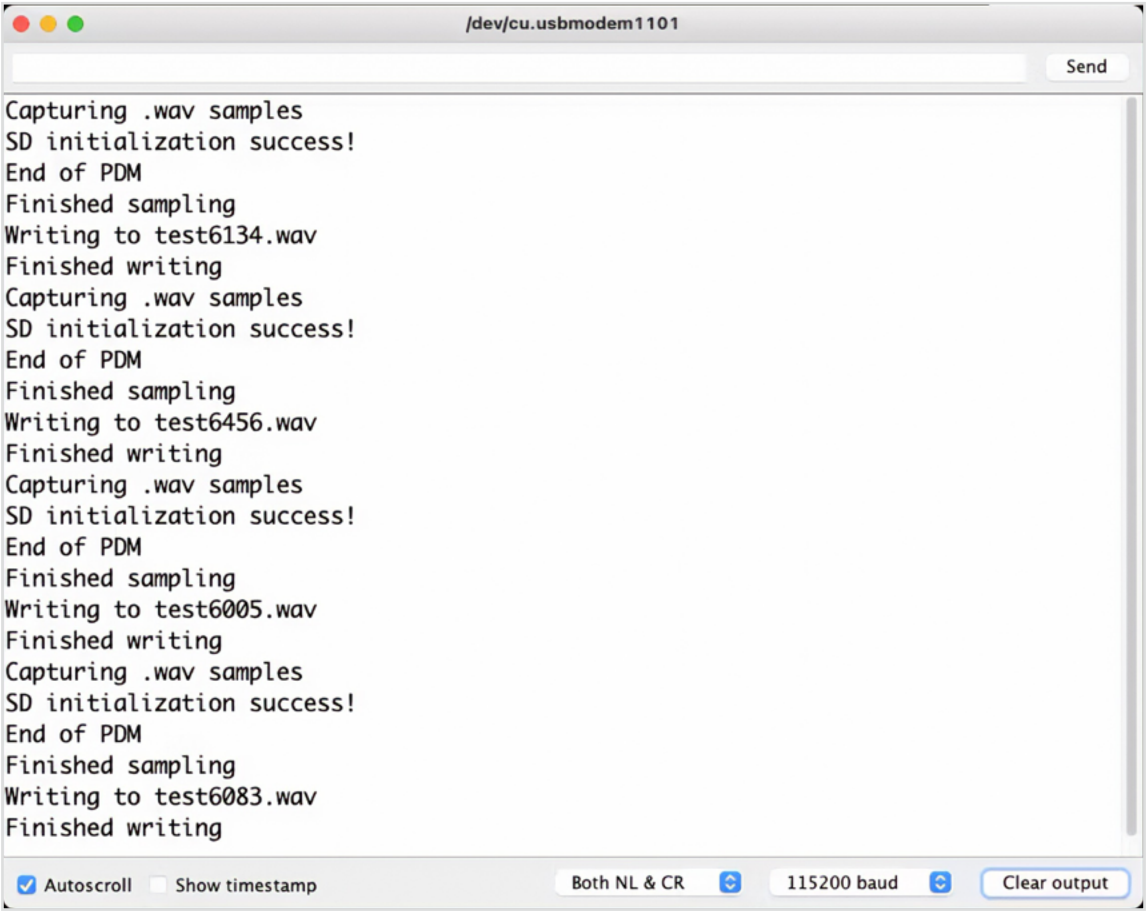
Each time you press the reset button, a **5 seconds audio sample** is recorded and saved on the SD card. I changed the original file to add LEDs to help during the recording process as below:

- During the time that LED Red is ON is possible to record ==> RECORD
- During the file writing process, LED Red is OFF ==> WAIT
- When finished writing, LED Green is ON ==> Press Reset Button once and wait for LED Red ON again, and proceed with a new sample recording

I realized that sometimes at the beginning and the end of each sample, a "spike" was recorded, so I cut the initial 300ms from each 5s sample. The spike verified at the end always happened after the recording process and should be eliminated on Edge Impulse Studio before training. Also, I increased the microphone gain to 30 dB.

The complete file (Xiao_mic_Saved_OnSDcard.ino) can be found on the Git Hub (3_KWS): Seeed-XIAO-BLE-Sense.

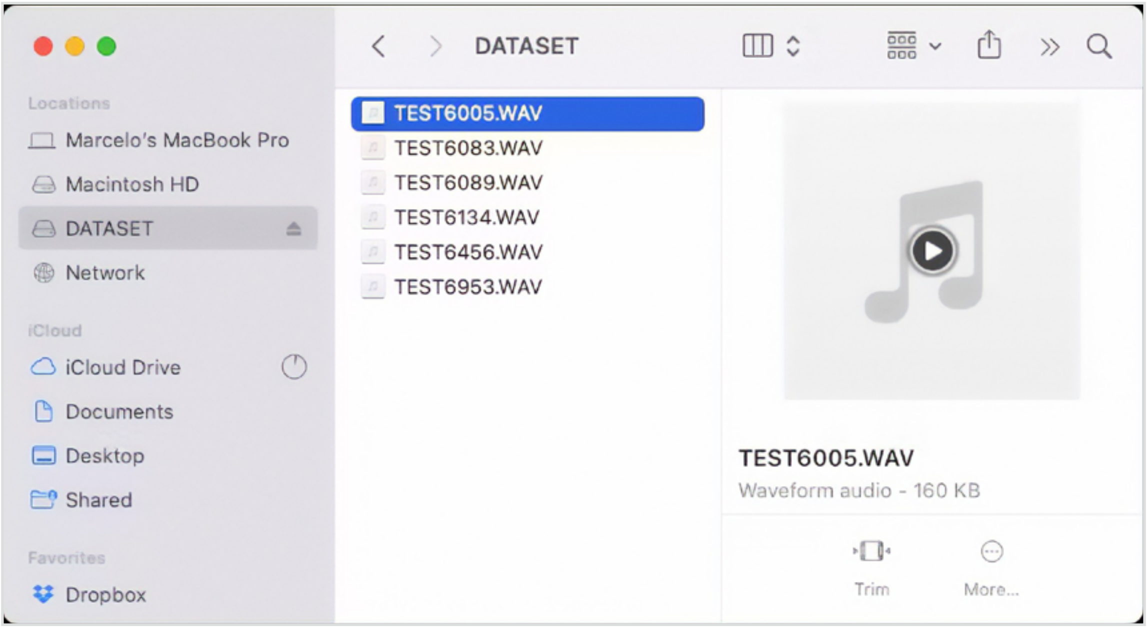
During the recording process, the.wav file names are shown on Serial Monitor:



```
Capturing .wav samples
SD initialization success!
End of PDM
Finished sampling
Writing to test6134.wav
Finished writing
Capturing .wav samples
SD initialization success!
End of PDM
Finished sampling
Writing to test6456.wav
Finished writing
Capturing .wav samples
SD initialization success!
End of PDM
Finished sampling
Writing to test6005.wav
Finished writing
Capturing .wav samples
SD initialization success!
End of PDM
Finished sampling
Writing to test6083.wav
Finished writing
```

☑ Autoscroll □ Show timestamp Both NL & CR 115200 baud Clear output

Take the SD card from the Expansion Board and insert it into your computer:



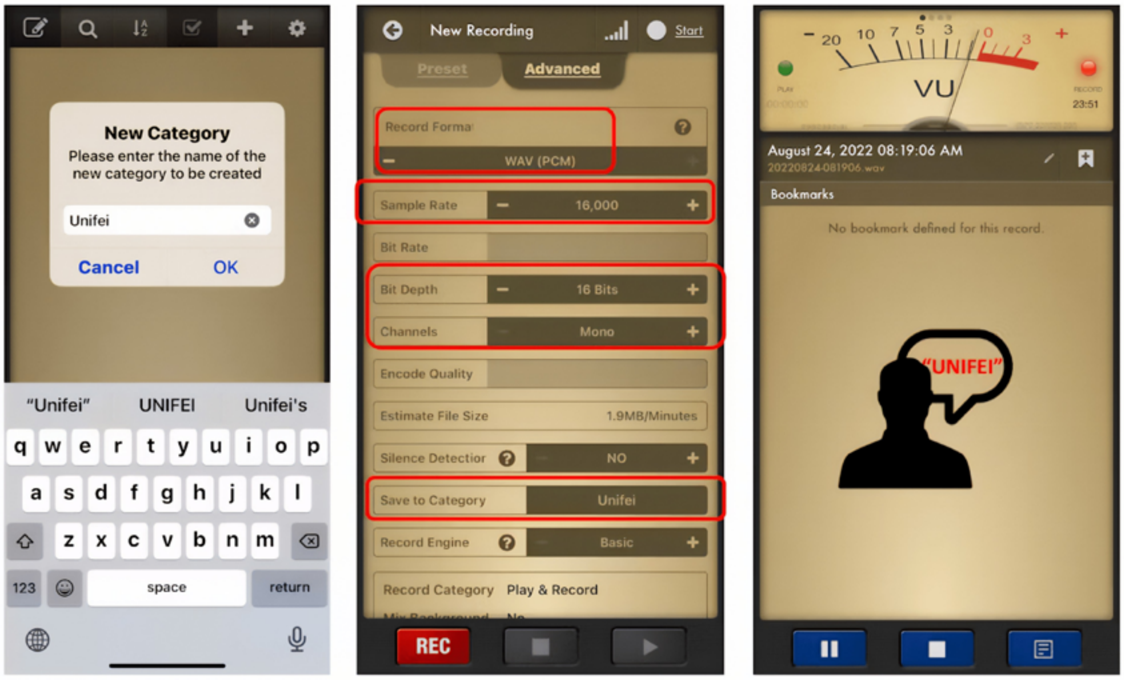
TEST6005.WAV
TEST6083.WAV
TEST6089.WAV
TEST6134.WAV
TEST6456.WAV
TEST6953.WAV

TEST6005.WAV
Waveform audio - 160 KB

The files are ready to be uploaded to Edge Impulse Studio

Capturing (offline) Audio Data with a smartphone or PC

Alternatively, you can use your PC or smartphone to capture audio data with a sampling frequency 16KHz and a bit depth of 16 Bits. A good app for that is Voice Recorder Pro (IOS). Save your record as .wav files and send them to your computer.


New Category
Please enter the name of the new category to be created
Unifei
Cancel OK

New Recording

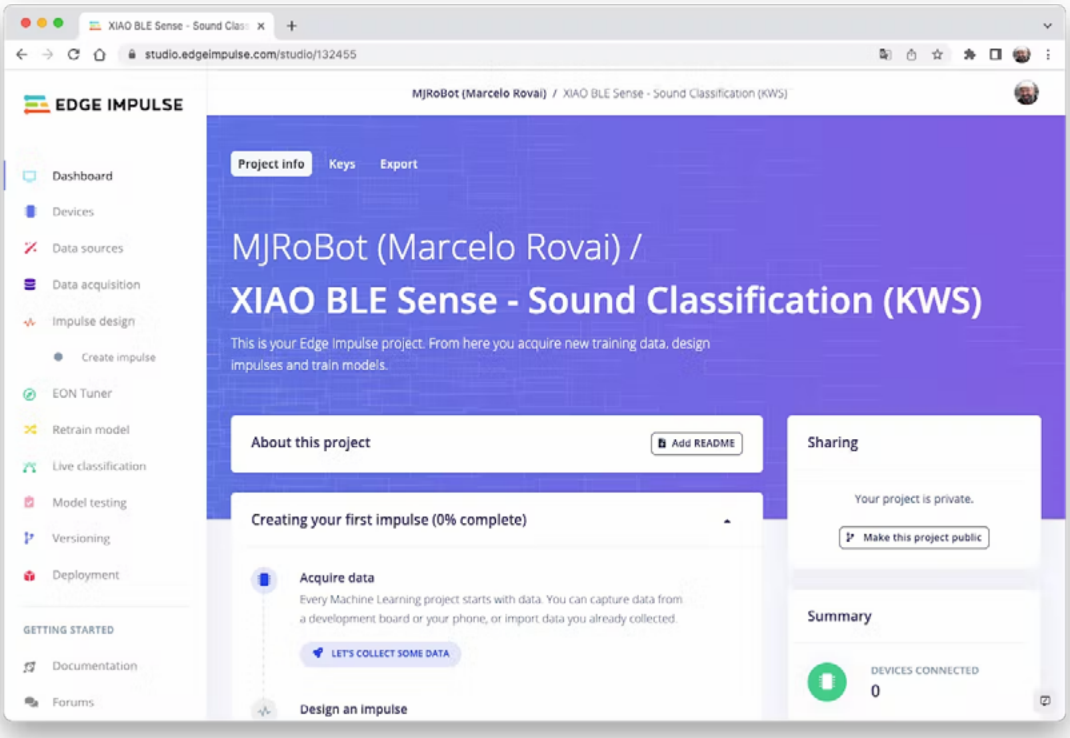
Record Format	WAV (PCM)
Sample Rate	16,000
Bit Rate	
Bit Depth	16 Bits
Channels	Mono
Encode Quality	
Estimate File Size	1.9MB/Minutes
Silence Detection	NO
Save to Category	Unifei
Record Engine	Basic
Record Category	Play & Record

August 24, 2022 08:19:06 AM
20220824-081906.wav
Bookmarks
No bookmark defined for this record.

Note that any smartphone app can be used for audio recording or even your computer, for example using Audacity.

Training model with Edge Impulse Studio

When the raw dataset is created, you should initiate a new project at Edge Impulse Studio:


studio.edgeimpulse.com/studio/132455

MJRoBot (Marcelo Rovai) / XIAO BLE Sense - Sound Classification (KWS)

MJRoBot (Marcelo Rovai) /
XIAO BLE Sense - Sound Classification (KWS)

This is your Edge Impulse project. From here you acquire new training data, design impulses and train models.

About this project

Creating your first impulse (0% complete)

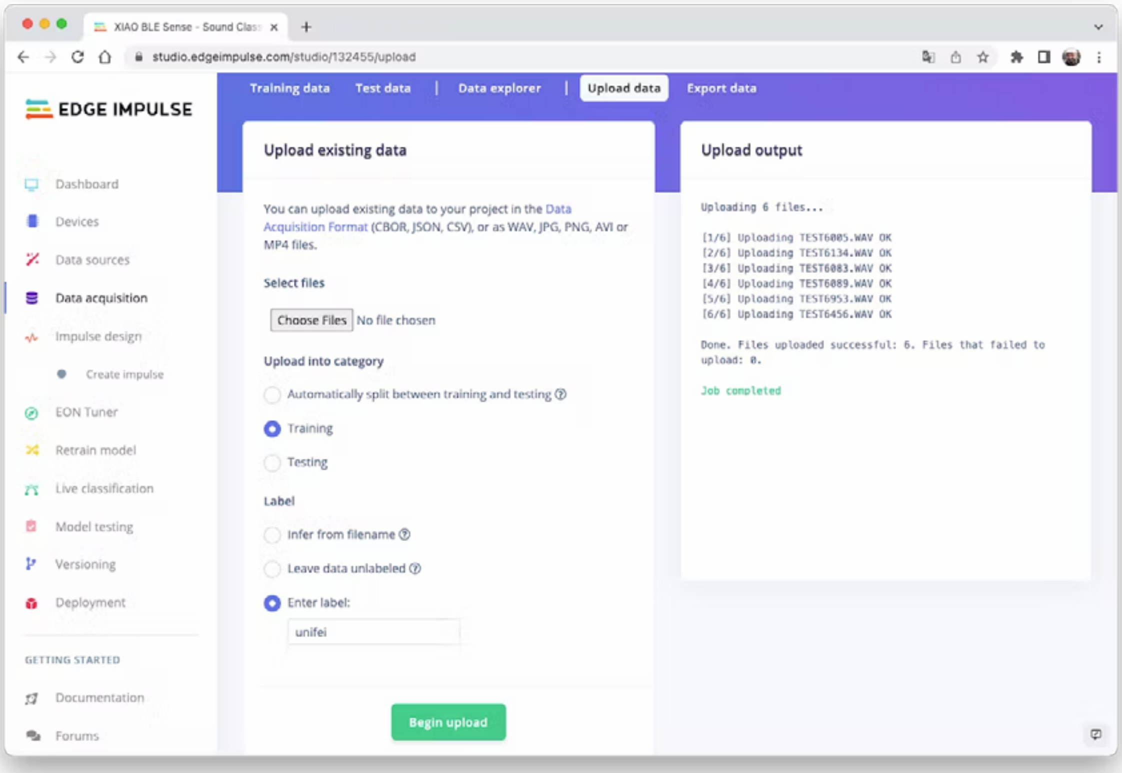
Acquire data
Every Machine Learning project starts with data. You can capture data from a development board or your phone, or import data you already collected.

Design an impulse

Sharing
Your project is private.

Summary
DEVICES CONNECTED
0

Once the project is created, go to the `Data Acquisition` section and select the `Upload Existing Data` tool. Choose the files to be uploaded, for example, I started uploading the samples recorded with the XIAO nRF52840 Sense:



Upload existing data

You can upload existing data to your project in the Data Acquisition Format (CBOR, JSON, CSV), or as WAV, JPG, PNG, AVI or MP4 files.

Select files

Choose Files No file chosen

Upload into category

○ Automatically split between training and testing

● Training

○ Testing

Label

○ Infer from filename

○ Leave data unlabeled

● Enter label:

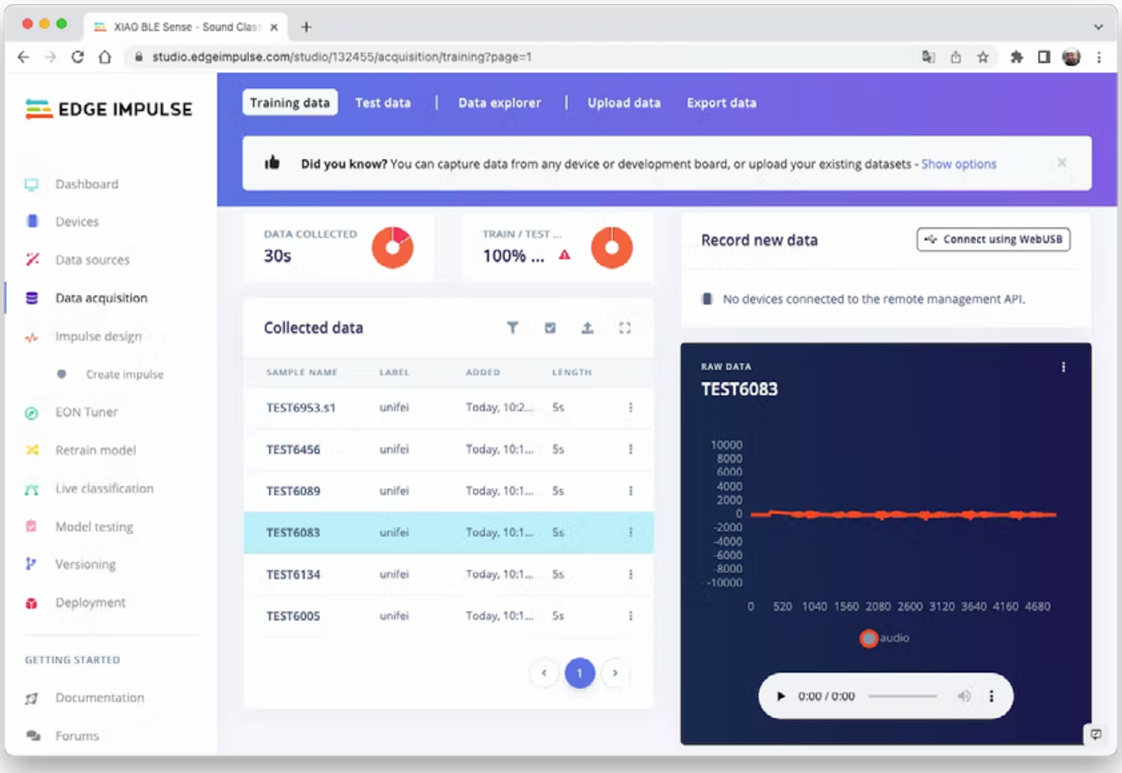
unifei

Begin upload

Upload output

```
Uploading 6 files...

[1/6] Uploading TEST6005.WAV OK
[2/6] Uploading TEST6134.WAV OK
[3/6] Uploading TEST6083.WAV OK
[4/6] Uploading TEST6089.WAV OK
[5/6] Uploading TEST6953.WAV OK
[6/6] Uploading TEST6456.WAV OK

Done. Files uploaded successful: 6. Files that failed to
upload: 0.

Job completed
```

The samples will now appear in the `Data acquisition` section:



Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLECTED 30s

TRAIN / TEST ... 100% ...

Record new data

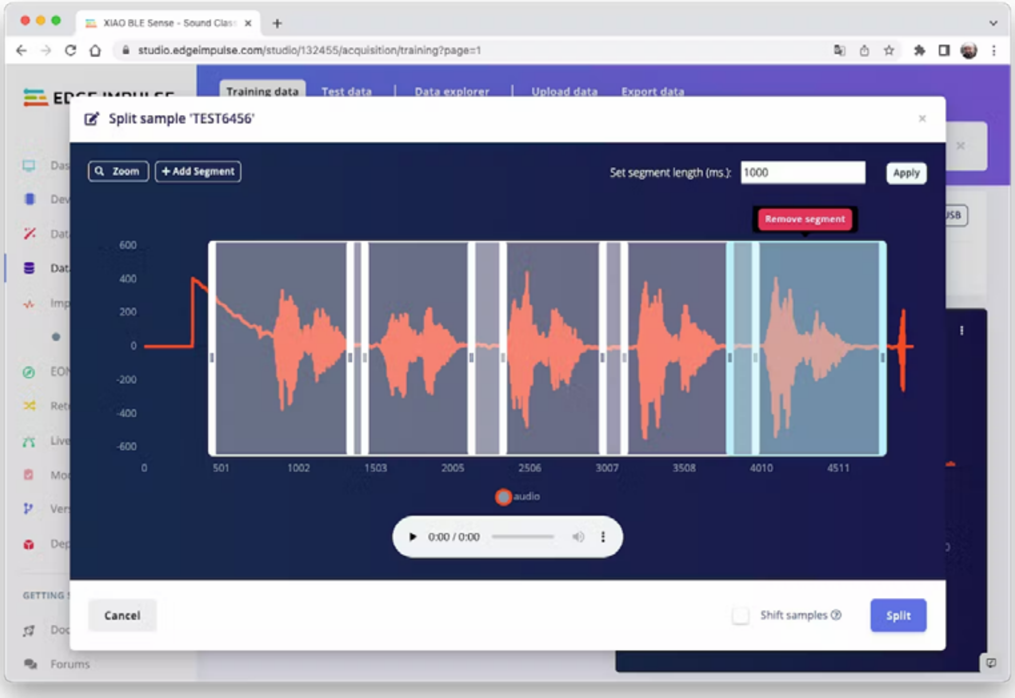
No devices connected to the remote management API.

Collected data

SAMPLE NAME	LABEL	ADDED	LENGTH
TEST6953.s1	unifei	Today, 10:2...	5s
TEST6456	unifei	Today, 10:1...	5s
TEST6089	unifei	Today, 10:1...	5s
TEST6083	unifei	Today, 10:1...	5s
TEST6134	unifei	Today, 10:1...	5s
TEST6005	unifei	Today, 10:1...	5s

TEST6083

Click on three dots after the sample name and select `Split sample`. Once inside de tool, split the data into 1-second records (try to avoid start and end portions):



Split sample 'TEST6456'

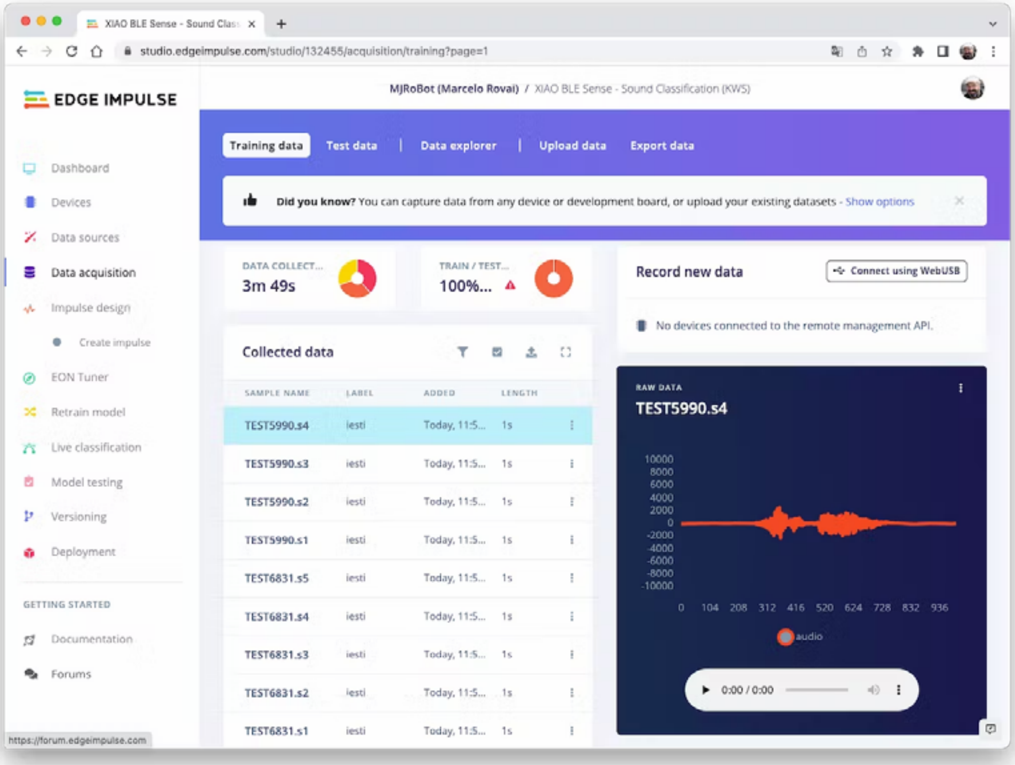
Set segment length (ms.): 1000

This procedure should be repeated for all samples. After that, upload other class samples (IESTI and SILENCE) captured with the XIAO and your PC or smartphone.

- Attention -

For longer audio files (minutes), first, split into 10-second segments and after that, use the tool again to get the final 1-second splits.

In the end, the dataset has around 70 1-second samples for each class:



MJRoBot (Marcelo Rovai) / XIAO BLE Sense - Sound Classification (KWS)

Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLECT... 3m 49s

TRAIN / TEST... 100%...

Record new data

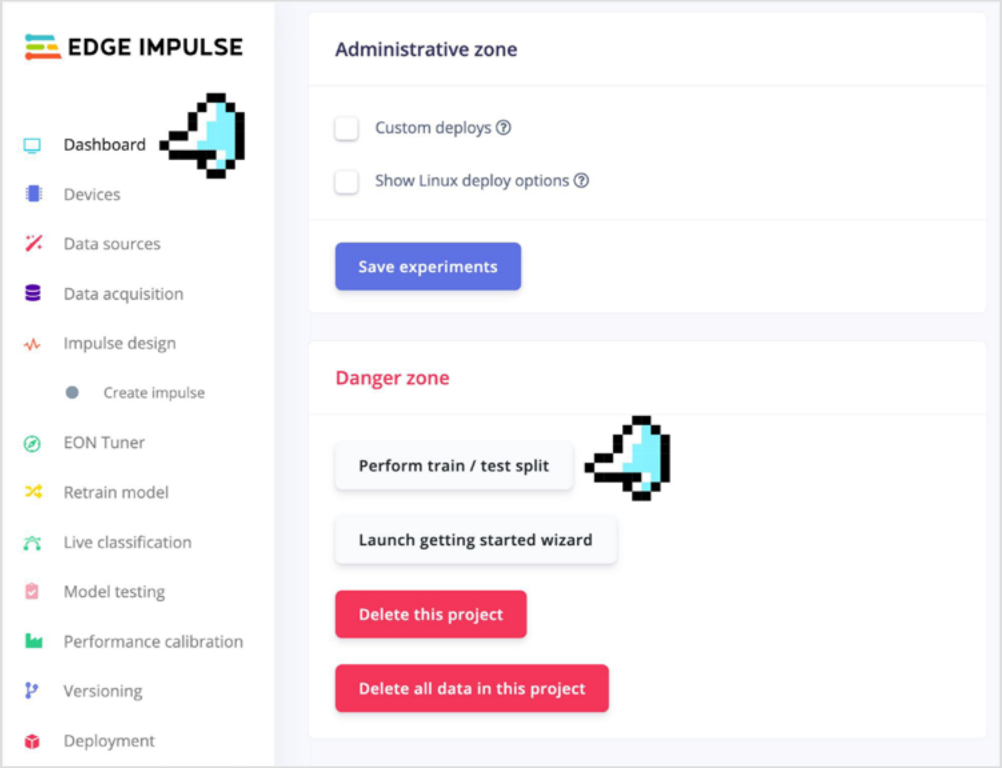
No devices connected to the remote management API.

Collected data

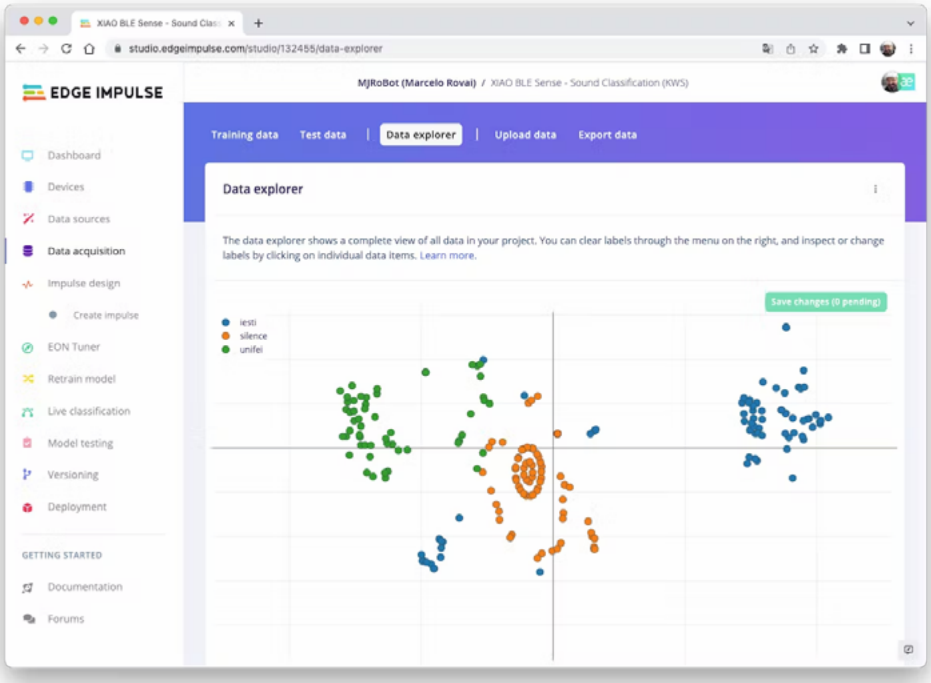
SAMPLE NAME	LABEL	ADDED	LENGTH
TEST5990.s4	iesti	Today, 11:5...	1s
TEST5990.s3	iesti	Today, 11:5...	1s
TEST5990.s2	iesti	Today, 11:5...	1s
TEST5990.s1	iesti	Today, 11:5...	1s
TEST6831.s5	iesti	Today, 11:5...	1s
TEST6831.s4	iesti	Today, 11:5...	1s
TEST6831.s3	iesti	Today, 11:5...	1s
TEST6831.s2	iesti	Today, 11:5...	1s
TEST6831.s1	iesti	Today, 11:5...	1s

RAW DATA
TEST5990.s4

Now, you should split that dataset into Train/Test. You can do it manually (using the `three dots` menu, moving samples individually) or using `Perform Train / Test Split` on `Dashboard – Danger Zone`.



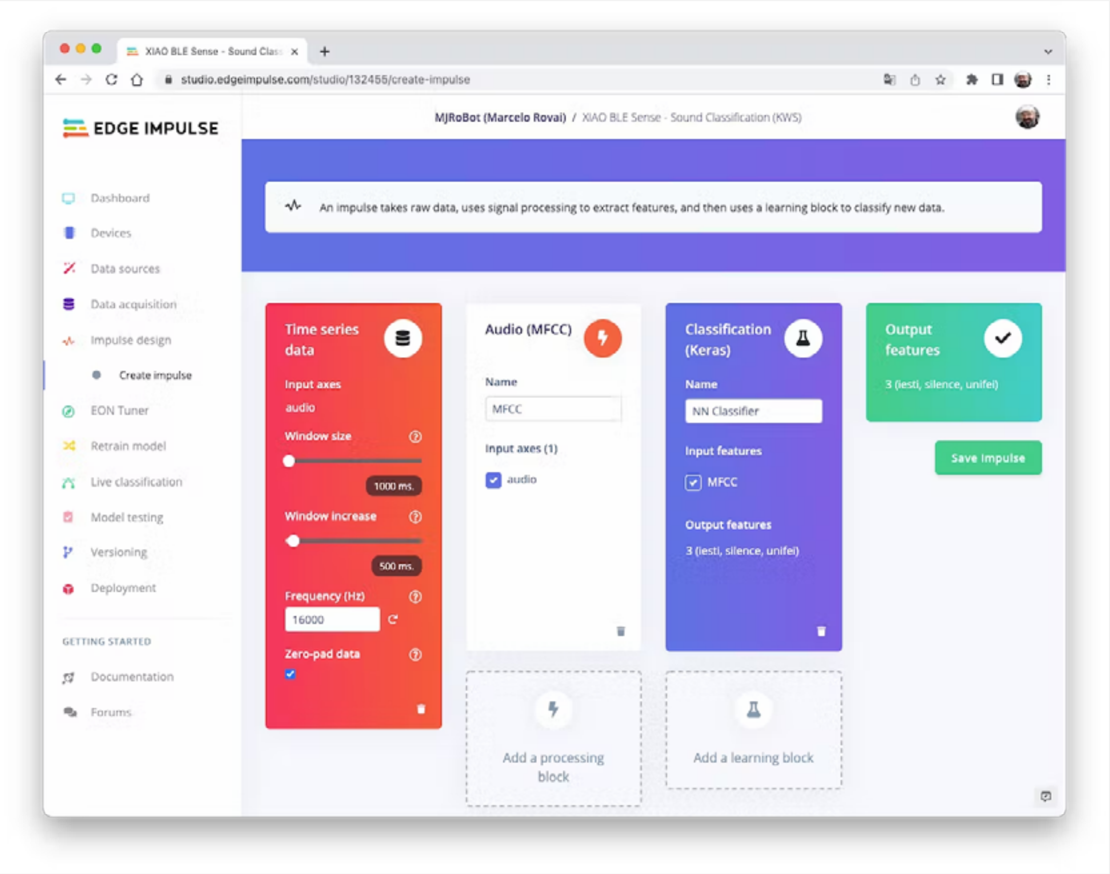
We can optionally check all datasets using the tab Data Explorer. The data points seem apart, which means that the classification model should work:



The data explorer shows a complete view of all data in your project. You can clear labels through the menu on the right, and inspect or change labels by clicking on individual data items. Learn more.

Creating Impulse (Pre-Process / Model definition)

An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.



An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Time series data
Input axes: audio
Window size: 1000 ms.
Window increase: 500 ms.
Frequency (Hz): 16000
Zero-pad data: ☑

Audio (MFCC)
Name: MFCC
Input axes (1): ☑ audio

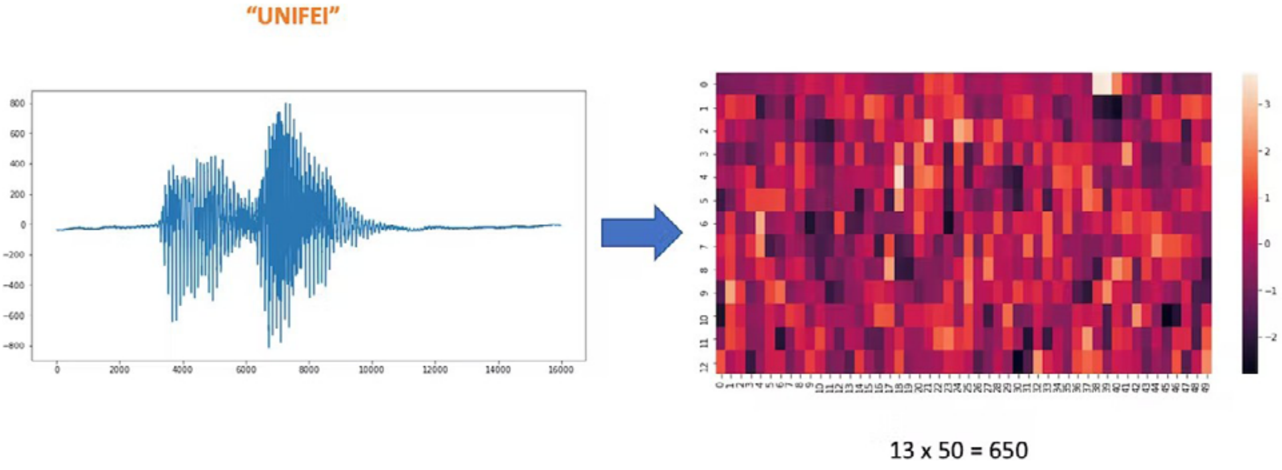
Classification (Keras)
Name: NN Classifier
Input features: ☑ MFCC
Output features: 3 (iesti, silence, unifei)

Output features
3 (iesti, silence, unifei)

Save Impulse

First, we will take the data points with a 1-second window, augmenting the data, sliding that window each 500ms. Note that the option zero-point pad is set. It is important to fill with zeros samples smaller than 1 second in some cases, I reduced the 1000 ms window on the split tool to avoid noises and spikes.

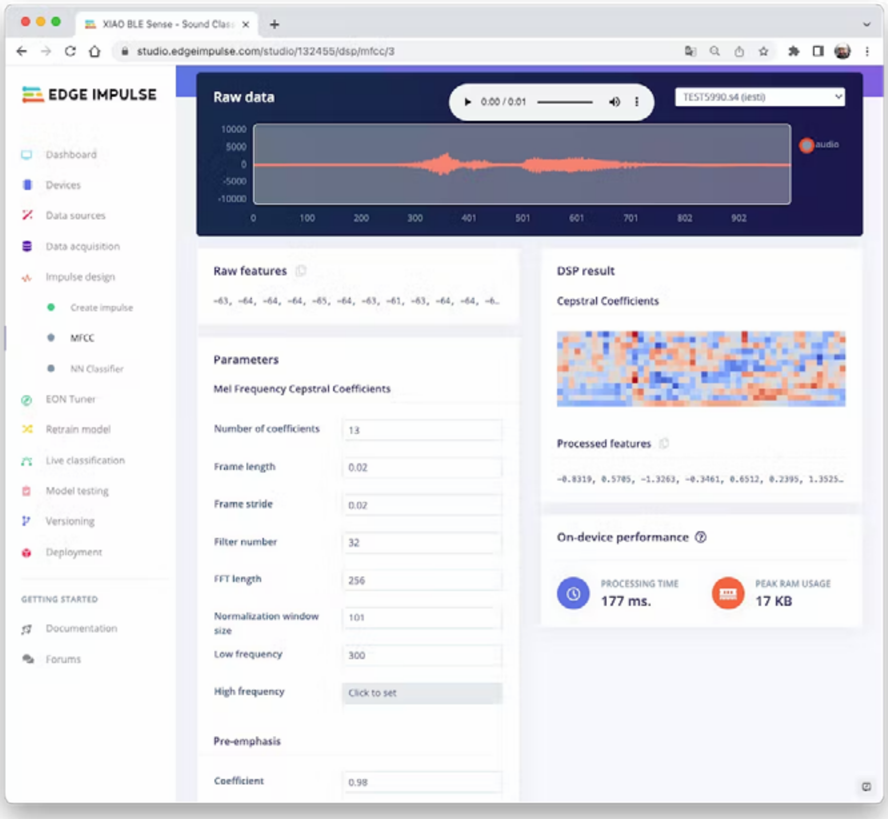
Each 1-second audio sample should be pre-processed and converted to an image (for example, 13 x 50 x 1). We will use `Audio (MFCC)`, which extracts features from audio signals using Mel Frequency Cepstral Coefficients, which are well suited for the human voice, which is our case here.



Next, we select the `Classification` block to build our model from scratch using a Convolution Neural Network (CNN).

Pre-Processing (MFCC)

The next step is to create the images to be trained in the next phase:



Raw features

-63, -64, -64, -64, -65, -64, -63, -61, -63, -64, -64, -6…

Parameters

Mel Frequency Cepstral Coefficients

Number of coefficients	13
Frame length	0.02
Frame stride	0.02
Filter number	32
FFT length	256
Normalization window size	101
Low frequency	300
High frequency	Click to set

Pre-emphasis

Coefficient	0.98

DSP result

Processed features

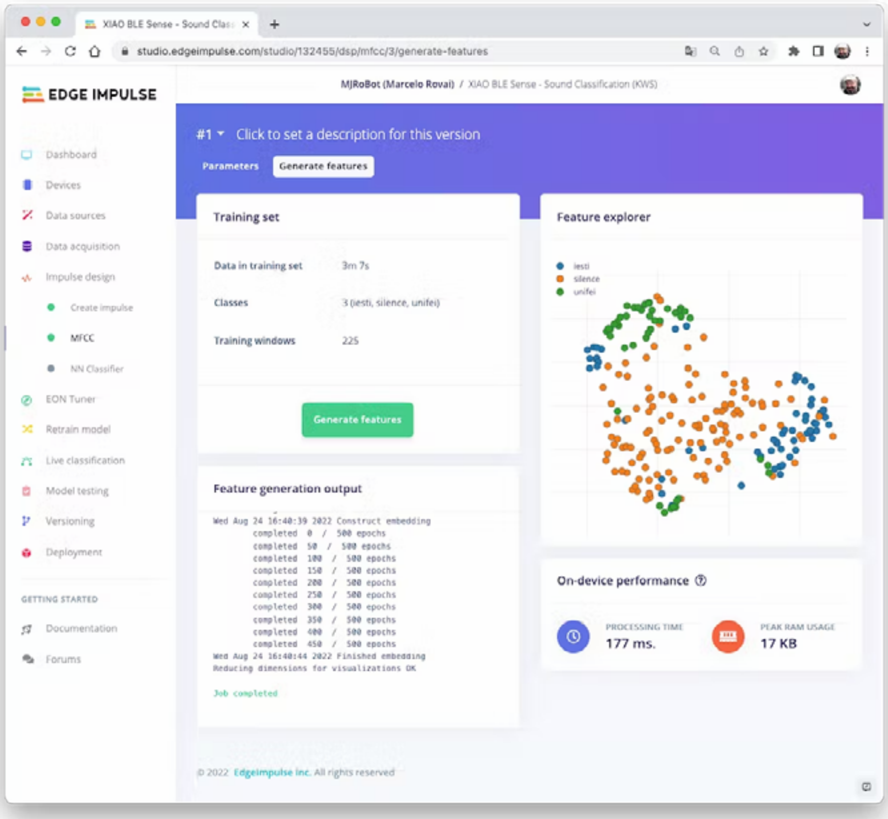
-0.8319, 0.5705, -1.3263, -0.3461, 0.6512, 0.2395, 1.3525…

On-device performance

PROCESSING TIME 177 ms.

PEAK RAM USAGE 17 KB

We will keep the default parameter values. We do not spend much memory to pre-process data (only 17KB), but the processing time is relatively high (177 ms for a Cortex-M4 CPU as our XIAO). Save parameters and generate features:



Training set

Data in training set	3m 7s
Classes	3 (iesti, silence, unifei)
Training windows	225

Feature generation output

```
Wed Aug 24 16:40:39 2022 Construct embedding
        completed  0  /  500 epochs
        completed  50  /  500 epochs
        completed  100  /  500 epochs
        completed  150  /  500 epochs
        completed  200  /  500 epochs
        completed  250  /  500 epochs
        completed  300  /  500 epochs
        completed  350  /  500 epochs
        completed  400  /  500 epochs
        completed  450  /  500 epochs
Wed Aug 24 16:40:44 2022 Finished embedding
Reducing dimensions for visualizations OK

Job completed
```

On-device performance

PROCESSING TIME 177 ms.

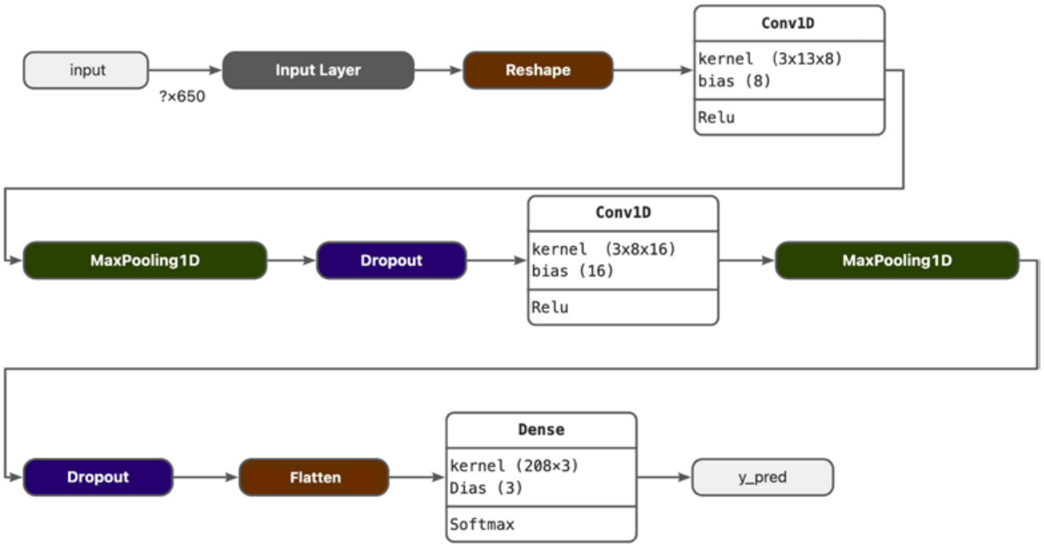
PEAK RAM USAGE 17 KB

Going under the hood

To understand better how the raw sound is preprocessed, look at the Feature Engineering for Audio Classification chapter. You can play with the MFCC features generation by downloading this notebook from GitHub or Opening it In Colab.

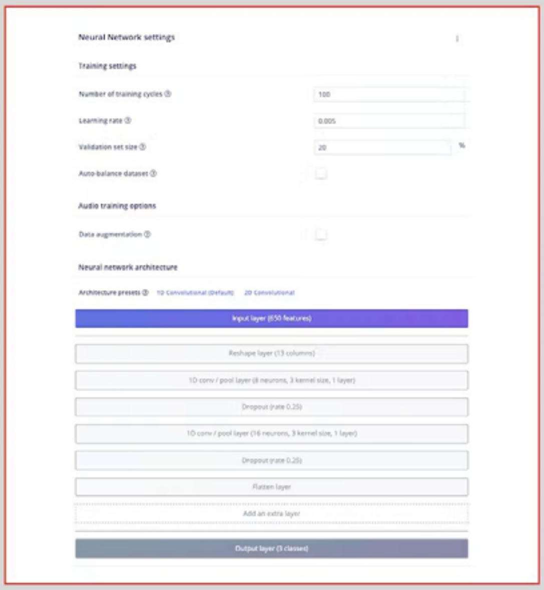
Model Design and Training

We will use a simple Convolution Neural Network (CNN) model, tested with 1D and 2D convolutions. The basic architecture has two blocks of Convolution + MaxPooling ([8] and [16] filters, respectively) and a Dropout of [0.25] for the 1D and [0.5] for the 2D. For the last layer, after Flattening, we have [3] neurons, one for each class:



As hyper-parameters, we will have a `Learning Rate` of [0.005] and a model trained by [100] epochs. We will also include a data augmentation method based on SpecAugment. We trained the 1D and the 2D models with the same hyperparameters. The 1D architecture had a better overall result (91.1% accuracy) when compared with 88% of the 2D, so we will use the 1D.

> *Using 1D convolutions is more efficient because it requires fewer parameters than 2D convolutions, making them more suitable for resource-constrained environments.*



Neural Network settings

Training settings

Number of training cycles ⓘ 100

Learning rate ⓘ 0.005

Validation set size ⓘ 20 %

Auto-balance dataset ⓘ □

Audio training options

Data augmentation ⓘ □

Neural network architecture

Architecture presets ⓘ 1D Convolutional (Default) 2D Convolutional

Input layer (650 features)

Reshape layer (13 columns)

1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

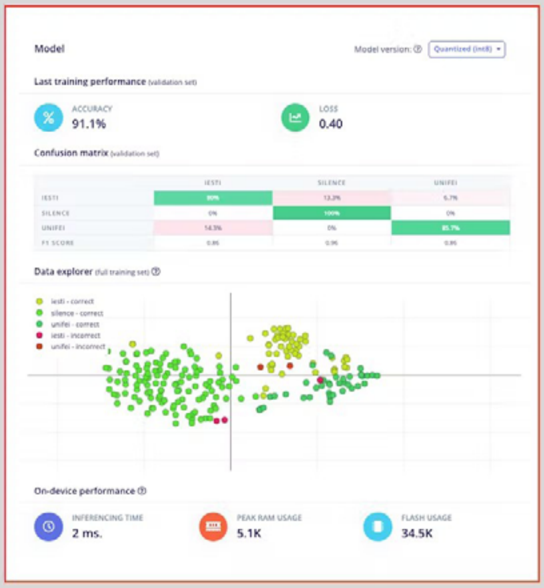
1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

Flatten layer

Add an extra layer

Output layer (3 classes)



Model

Model version: ⓘ Quantized (int8)

Last training performance (validation set)

ACCURACY 91.1%

LOSS 0.40

Confusion matrix (validation set)

	IESTI	SILENCE	UNIFEI
IESTI	80%	13.3%	6.7%
SILENCE	0%	100%	0%
UNIFEI	14.3%	0%	85.7%
F1 SCORE	0.86	0.96	0.86

Data explorer (full training set) ⓘ

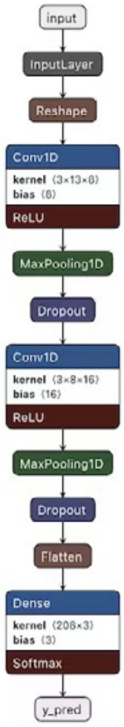
On-device performance ⓘ

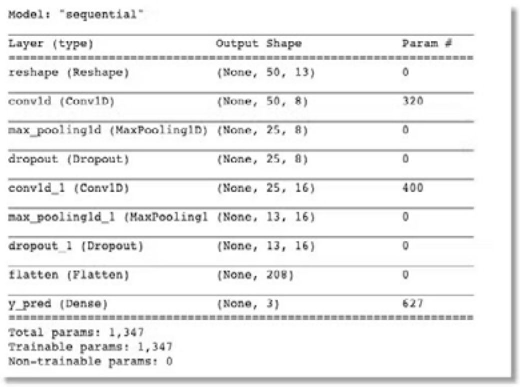
INFERENCING TIME 2 ms.

PEAK RAM USAGE 5.1K

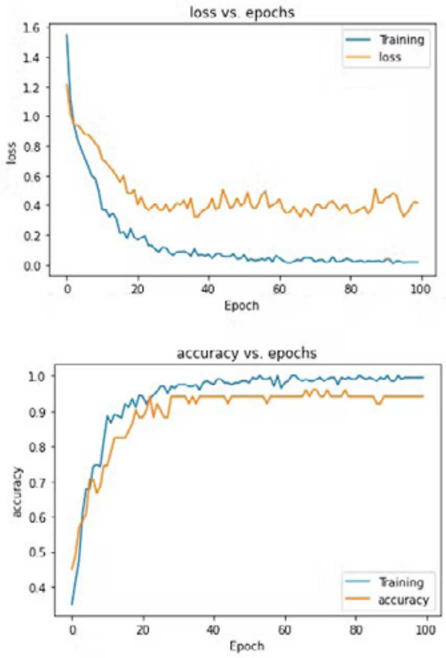
FLASH USAGE 34.5K

If you want to understand what is happening "under the hood," you can download the pre-processed dataset (MFCC training data) from the Dashboard tab and run this Jupyter Notebook, playing with the code or Opening it In Colab. You should adapt the notebook for your data and model. For example, you can analyze the accuracy by each epoch:



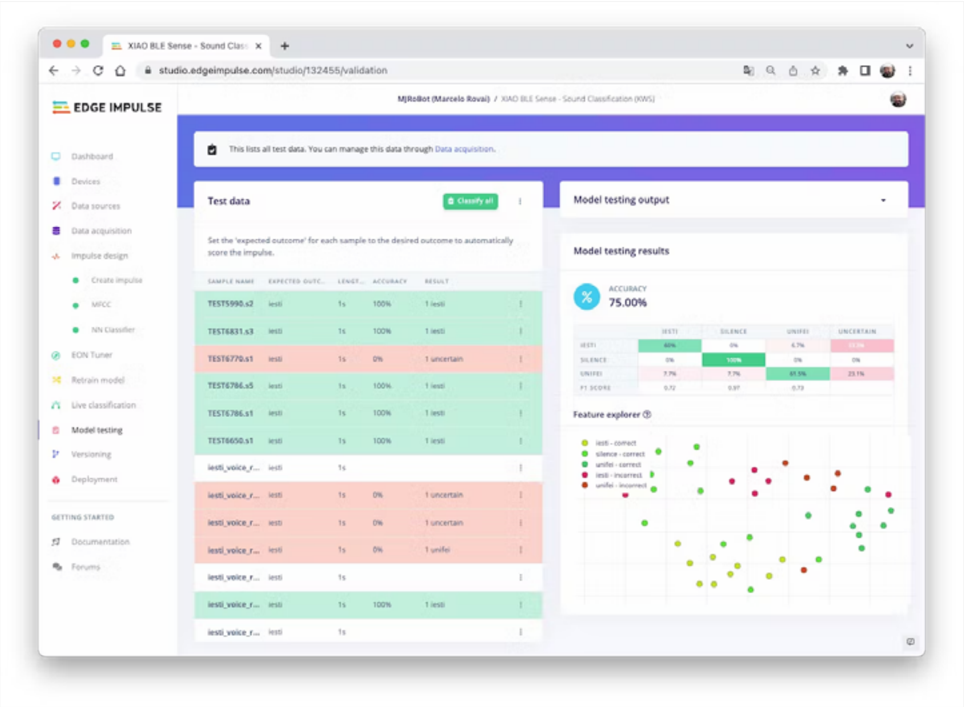


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 50, 13)            0
_________________________________________________________________
conv1d (Conv1D)              (None, 50, 8)             320
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 25, 8)             0
_________________________________________________________________
dropout (Dropout)            (None, 25, 8)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 25, 16)            400
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 13, 16)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 16)            0
_________________________________________________________________
flatten (Flatten)            (None, 208)               0
_________________________________________________________________
y_pred (Dense)               (None, 3)                 627
=================================================================
Total params: 1,347
Trainable params: 1,347
Non-trainable params: 0
```



Testing

Testing the model with the data put apart before training (Test Data), we got an accuracy of 75%. Based on the small amount of data used, it is OK, but I strongly suggest increasing the number of samples.



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

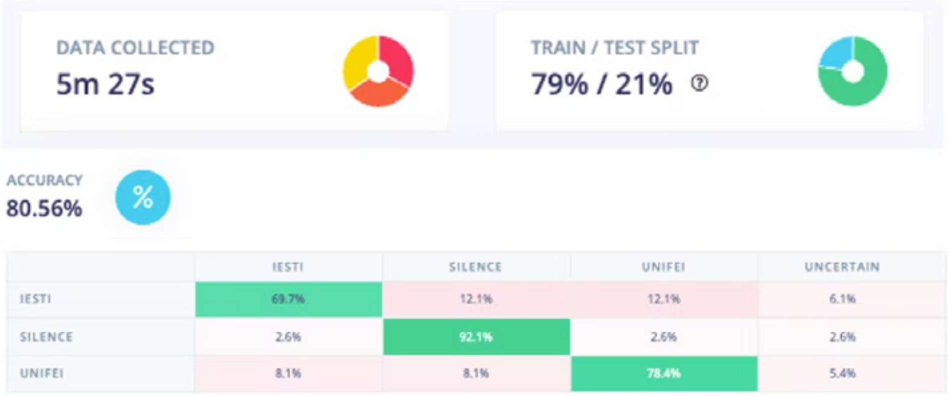
SAMPLE NAME	EXPECTED OUTC...	LENGT...	ACCURACY	RESULT
TEST5990.s2	iesti	1s	100%	1 iesti
TEST6831.s3	iesti	1s	100%	1 iesti
TEST6770.s1	iesti	1s	0%	1 uncertain
TEST6786.s5	iesti	1s	100%	1 iesti
TEST6786.s1	iesti	1s	100%	1 iesti
TEST6650.s1	iesti	1s	100%	1 iesti
iesti_voice_r...	iesti	1s		
iesti_voice_r...	iesti	1s	0%	1 uncertain
iesti_voice_r...	iesti	1s	0%	1 uncertain
iesti_voice_r...	iesti	1s	0%	1 unifei
iesti_voice_r...	iesti	1s		
iesti_voice_r...	iesti	1s	100%	1 iesti
iesti_voice_r...	iesti	1s		

Model testing results

ACCURACY
75.00%

	IESTI	SILENCE	UNIFEI	UNCERTAIN
IESTI	60%	0%	6.7%	33.3%
SILENCE	0%	100%	0%	0%
UNIFEI	7.7%	7.7%	61.5%	23.1%
F1 SCORE	0.72	0.97	0.73	

Collecting more data, the Test accuracy moved up around 5%, going from 75% to around 81%:



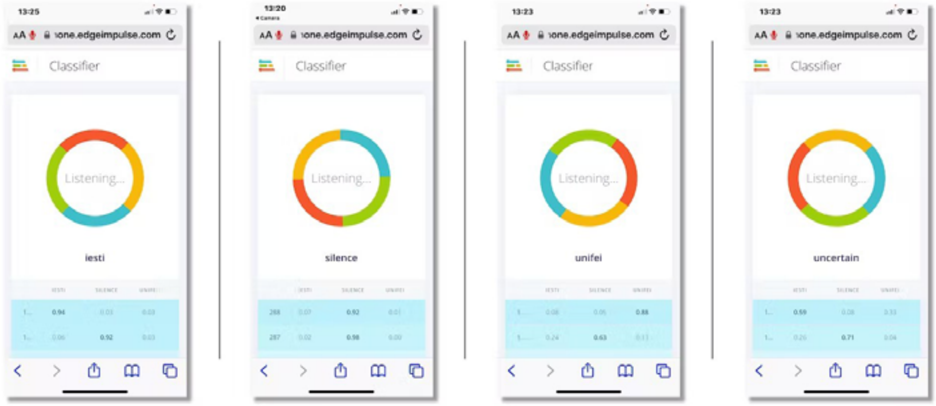
DATA COLLECTED
5m 27s

TRAIN / TEST SPLIT
79% / 21%

ACCURACY
80.56%

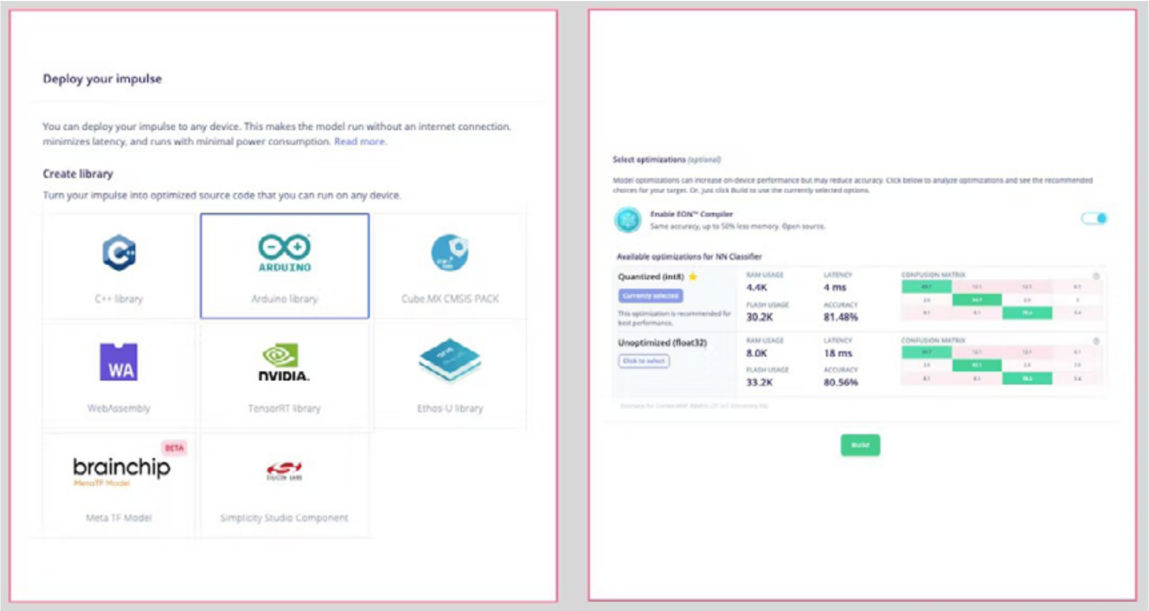
	IESTI	SILENCE	UNIFEI	UNCERTAIN
IESTI	69.7%	12.1%	12.1%	6.1%
SILENCE	2.6%	92.1%	2.6%	2.6%
UNIFEI	8.1%	8.1%	78.4%	5.4%

Now, we can proceed with the project, but before deployment on our device, it is possible to perform Live Classification using a Smart Phone, confirming that the model is working with live and real data:



Deploy and Inference

The Studio will package all the needed libraries, preprocessing functions, and trained models, downloading them to your computer. You should select the option Arduino Library and at the bottom, choose `Quantized (Int8)` and `[Build]`.



Deploy your impulse

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Create library

Turn your impulse into optimized source code that you can run on any device.

Select optimizations (optional)

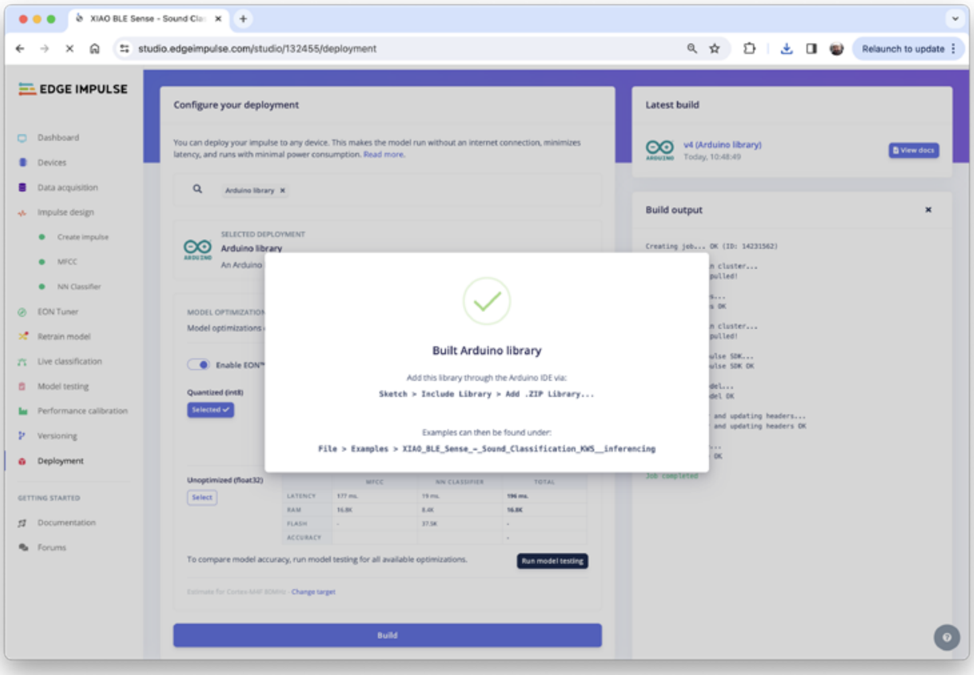
Model optimizations can increase on-device performance but may reduce accuracy. Click below to analyze optimizations and see the recommended choices for your target. Or, just click Build to use the currently selected options.

Enable EON™ Compiler
Same accuracy, up to 50% less memory. Open source.

Available optimizations for NN Classifier

	RAM USAGE	LATENCY	FLASH USAGE	ACCURACY
Quantized (int8) ★ Currently selected. This optimization is recommended for best performance.	4.4K	4 ms	30.2K	81.48%
Unoptimized (float32) Click to select	8.0K	18 ms	33.2K	80.56%

A Zip file will be created and downloaded to your computer:



You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

To compare model accuracy, run model testing for all available optimizations.

Built Arduino library

Add this library through the Arduino IDE via:

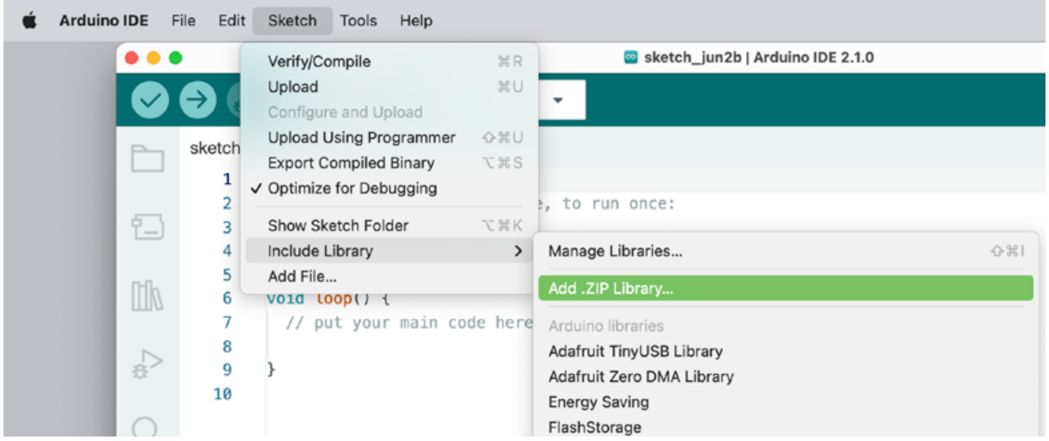
`Sketch > Include Library > Add .ZIP Library...`

Examples can then be found under:

`File > Examples > XIAO_BLE_Sense_-_Sound_Classification_KWS__inferencing`

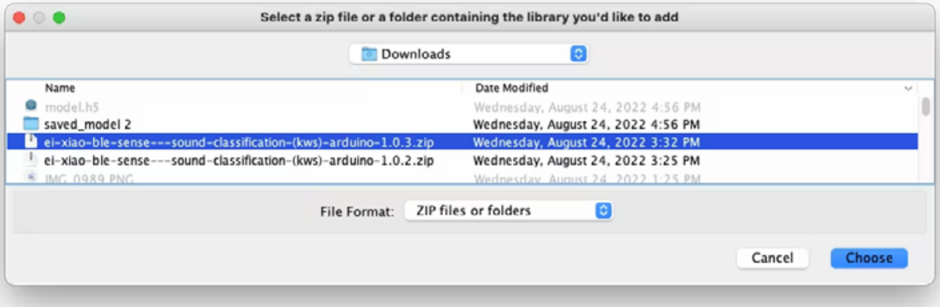
```
Creating job... OK (ID: 14231562)
```

On your Arduino IDE, go to the `Sketch` tab and select the option `Add .ZIP Library`.



```
, to run once:
void loop() {
  // put your main code here
}
```

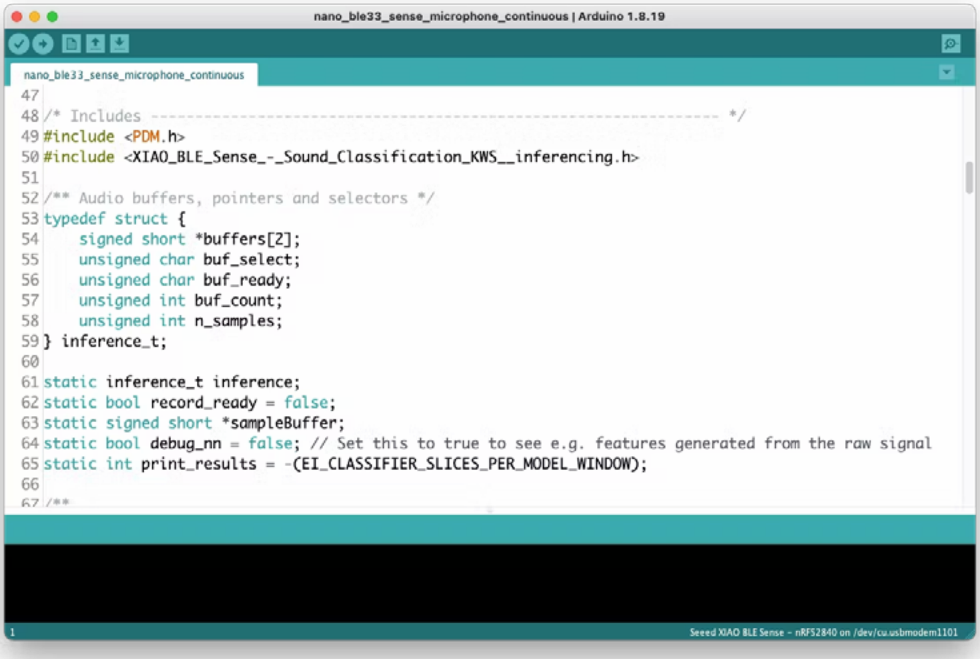
And Choose the.zip file downloaded by the Studio:


Select a zip file or a folder containing the library you'd like to add
Downloads
File Format:
ZIP files or folders
Cancel
Choose

Name	Date Modified
model.h5	Wednesday, August 24, 2022 4:56 PM
saved_model 2	Wednesday, August 24, 2022 4:56 PM
ei-xiao-ble-sense---sound-classification-(kws)-arduino-1.0.3.zip	Wednesday, August 24, 2022 3:32 PM
ei-xiao-ble-sense---sound-classification-(kws)-arduino-1.0.2.zip	Wednesday, August 24, 2022 3:25 PM
IMG_0989.PNG	Wednesday, August 24, 2022 1:25 PM

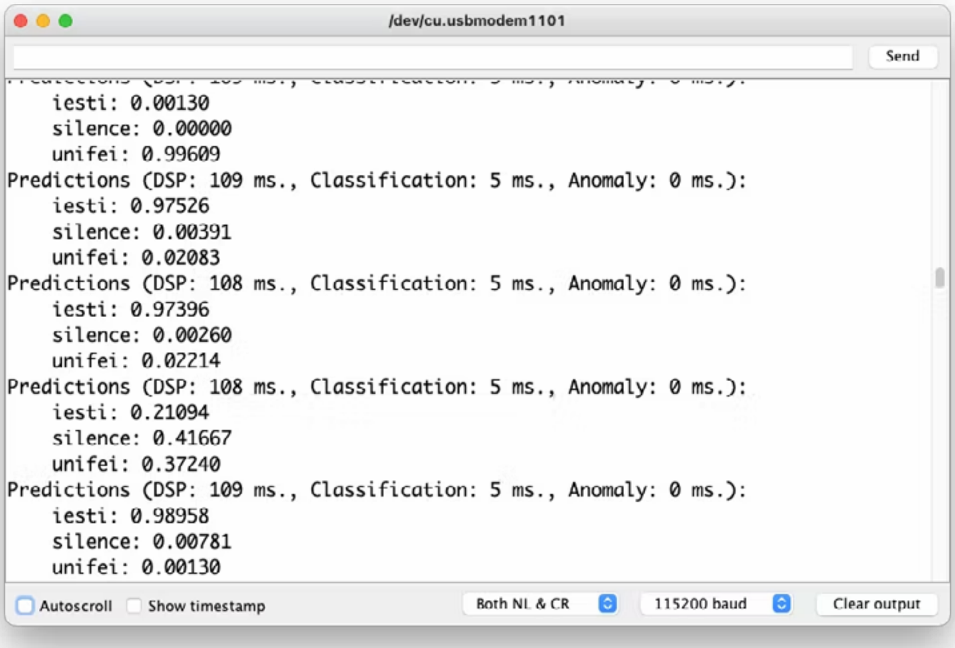
Now, it is time for a real test. We will make inferences wholly disconnected from the Studio. Let's change one of the code examples created when you deploy the Arduino Library.

In your Arduino IDE, go to the `File/Examples` tab and look for your project, and on examples, select `nano_ble33_sense_microphone_continuous`:


nano_ble33_sense_microphone_continuous | Arduino 1.8.19
nano_ble33_sense_microphone_continuous
Seeed XIAO BLE Sense - nRF52840 on /dev/cu.usbmodem1101

```

/* Includes ---------------------------------------------------------------- */
#include <PDM.h>
#include <XIAO_BLE_Sense_-_Sound_Classification_KWS__inferencing.h>

/** Audio buffers, pointers and selectors */
typedef struct {
    signed short *buffers[2];
    unsigned char buf_select;
    unsigned char buf_ready;
    unsigned int buf_count;
    unsigned int n_samples;
} inference_t;

static inference_t inference;
static bool record_ready = false;
static signed short *sampleBuffer;
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal
static int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);

/**
```

Even though the XIAO is not the same as the Arduino, both have the same MPU and PDM microphone, so the code works as it is. Upload the sketch to XIAO and open the Serial Monitor. Start talking about one or another Keyword and confirm that the model is working correctly:



```
    iesti: 0.00130
    silence: 0.00000
    unifei: 0.99609
Predictions (DSP: 109 ms., Classification: 5 ms., Anomaly: 0 ms.):
    iesti: 0.97526
    silence: 0.00391
    unifei: 0.02083
Predictions (DSP: 108 ms., Classification: 5 ms., Anomaly: 0 ms.):
    iesti: 0.97396
    silence: 0.00260
    unifei: 0.02214
Predictions (DSP: 108 ms., Classification: 5 ms., Anomaly: 0 ms.):
    iesti: 0.21094
    silence: 0.41667
    unifei: 0.37240
Predictions (DSP: 109 ms., Classification: 5 ms., Anomaly: 0 ms.):
    iesti: 0.98958
    silence: 0.00781
    unifei: 0.00130
```

Postprocessing

Now that we know that the model is working by detecting our two keywords, let's modify the code so we can see the result with the XIAO nRF52840 Sense completely offline (disconnected from the PC and powered by a battery).

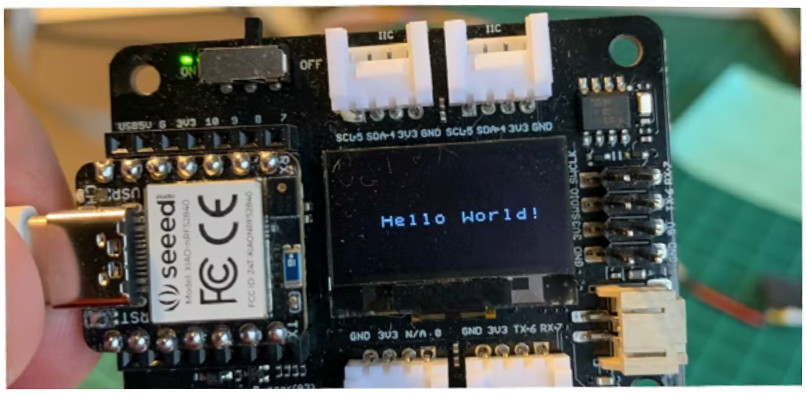
The idea is that whenever the keyword UNIFEI is detected, the LED Red will be ON; if it is IESTI, LED Green will be ON, and if it is SILENCE (No Keyword), both LEDs will be OFF.

> *If you have the XIAO nRF52840 Sense installed on the Expansion Board, we can display the class label and its probability. Otherwise, use only the LEDs.*

Let's go by Parts: Installing and Testing the SSD Display In your Arduino IDE, Install the u8g2 library and run the below code for testing:

```
#include <Arduino.h>
#include <U8x8lib.h>
#include <Wire.h>

U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(PIN_WIRE_SCL, PIN_WIRE_SDA, U8X8_PIN_NONE);

void setup(void) {
    u8x8.begin();
     u8x8.setFlipMode(0);    // set number from 1 to 3, the screen word should rotate 
180
}

void loop(void) {
    u8x8.setFont(u8x8_font_chroma48medium8_r);
    u8x8.setCursor(0, 0);
    u8x8.print(“Hello World!”);
}
```

And you should see the "Hello World" displayed on the SSD:

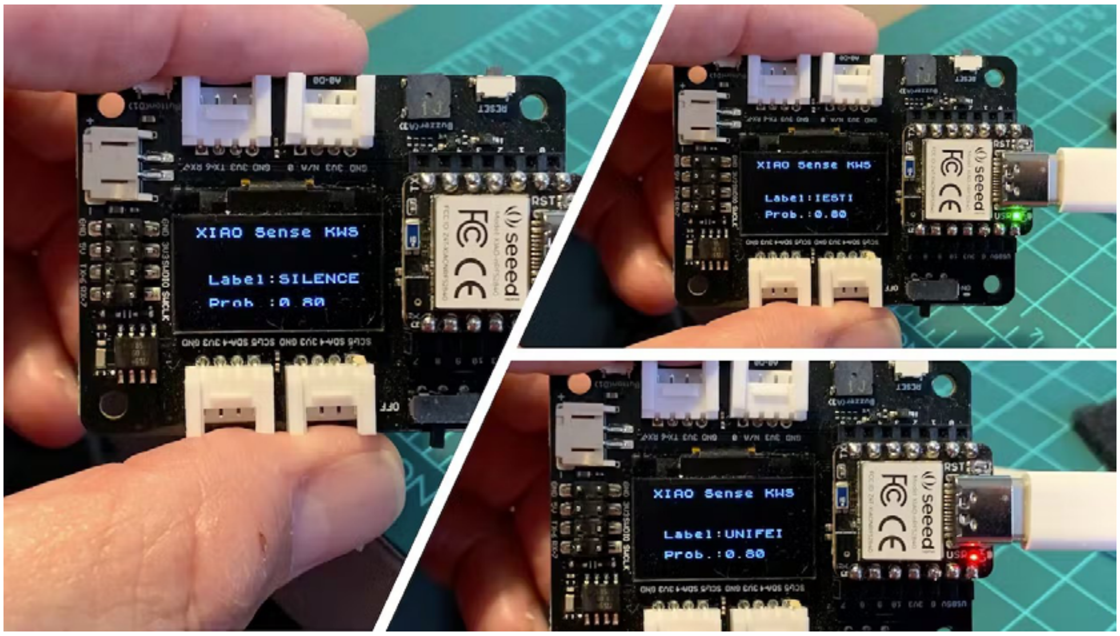


Now, let's create some functions that, depending on the values of pred_index and pred_value, will trigger the proper LED and display the class and probability. The code below will simulate some inference results and present them on display and LEDs:

```
/* Includes ---------------------------------------------------------------- */
#include <Arduino.h>
#include <U8x8lib.h>
#include <Wire.h>

#define NUMBER_CLASSES 3

/** OLED */
U8X8_SSD1306_128X64_NONAME_HW_I2C oled(PIN_WIRE_SCL, PIN_WIRE_SDA, U8X8_PIN_NONE);

Int pred_index = 0;
float pred_value = 0;
String lbl = “ “;
```

```
void setup() {
    pinMode(LEDR, OUTPUT);
    pinMode(LEDG, OUTPUT);
    pinMode(LEDB, OUTPUT);

    digitalWrite(LEDR, HIGH);
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);

    oled.begin();
    oled.setFlipMode(2);
     oled.setFont(u8x8_font_chroma48me-
dium8_r);
    oled.setCursor(0, 0);
    oled.print(“ XIAO Sense KWS”);
}

/**
* @brief      turn_off_leds function -
turn-off all RGB LEDs
*/
void turn_off_leds(){
    digitalWrite(LEDR, HIGH);
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);
}

/**
* @brief       Show Inference Results
on OLED Display
*/
void display_oled(int pred_index, float
pred_value){
    switch (pred_index){
        case 0:
            turn_off_leds();
            digitalWrite(LEDG, LOW);
            lbl = “IESTI  “ ;
            break;

        case 1:
            turn_off_leds();
            lbl = “SILENCE”;
            break;

        case 2:
            turn_off_leds();
            digitalWrite(LEDR, LOW);
            lbl = “UNIFEI “;
            break;
    }
    oled.setCursor(0, 2);
    oled.print(“       “);
    oled.setCursor(2, 4);
    oled.print(“Label:”);
    oled.print(lbl);
    oled.setCursor(2, 6);
    oled.print(“Prob.:”);
    oled.print(pred_value);
}

void loop() {
     for (int i = 0; i < NUMBER_CLASS-
ES; i++) {
        pred_index = i;
        pred_value = 0.8;
         display_oled(pred_index, pred_
value);
        delay(2000);
    }
}
```

Running the above code, you should get the below result:



You should merge the above code (Initialization and functions) with the `nano_ble33_sense_microphone_continuous.ino` you initially used to test your model. Also, you should include the below code on `loop()` between the lines:

```
ei_printf(“: \n”);
...
#if EI_CLASSIFIER_HAS_ANOMALY == 1
```

And replacing the original function to print inference results on the Serial Monitor:

```
int pred_index = 0;     // Initialize pred_index
float pred_value = 0;   // Initialize pred_value

for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
     ei_printf(“     %s: %.5f\n”, result.classification[ix].label, result.classifica-
tion[ix].value);
    if (result.classification[ix].value > pred_value){
        pred_index = ix;
        pred_value = result.classification[ix].value;
    }
}
display_oled(pred_index, pred_value);
```

Here you can see how the final project is: https://youtu.be/1ex88hSqqyI

> *The complete code can be found on the GitHub (3_KWS): Seeed-XIAO-BLE-Sense.*

Conclusion

The Seeed XIAO nRF52840 Sense is really a giant tiny device! However, it is powerful, trustworthy, not expensive, low power, and has suitable sensors to be used on the most common embedded machine learning applications such as movement and sound.

Even though Edge Impulse does not officially support XIAO nRF52840 Sense (yet!), we also realized that it could use Studio for training and deployment.

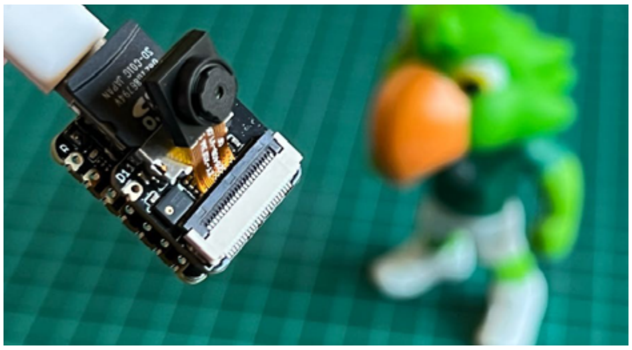
> *On the GitHub repository, you will find the last version of the codes in the 3_KWS folder: Seeed-XIAO-BLE-Sense*

Before we finish, consider that Sound Classification is more than just voice. For example, you can develop TinyML projects around sound in several areas as:

- Security (Broken Glass detection)
- Industry (Anomaly Detection)
- Medical (Snore, Toss, Pulmonary diseases)
- Nature (Beehive control, insect sound)

4.4
Image Classification

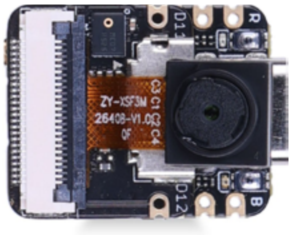
In this section, let's explore Computer Vision ML applications on the XIAO ESP32S3 Sense.



4.4.1 Things used in this project

Hardware components

- Seeed Studio Seeed XIAO ESP32S3 Sense × 1



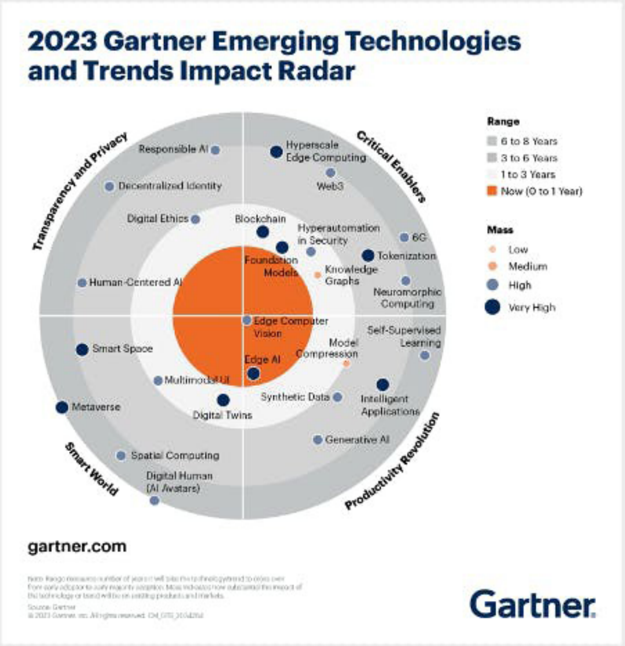
Software apps and online services

- Arduino IDE
- Edge Impulse Studio

4.4.2 Introduction

More and more, we are facing an artificial intelligence (AI) revolution where, as stated by Gartner, **Edge AI** has a very high impact potential, and **it is for now**!

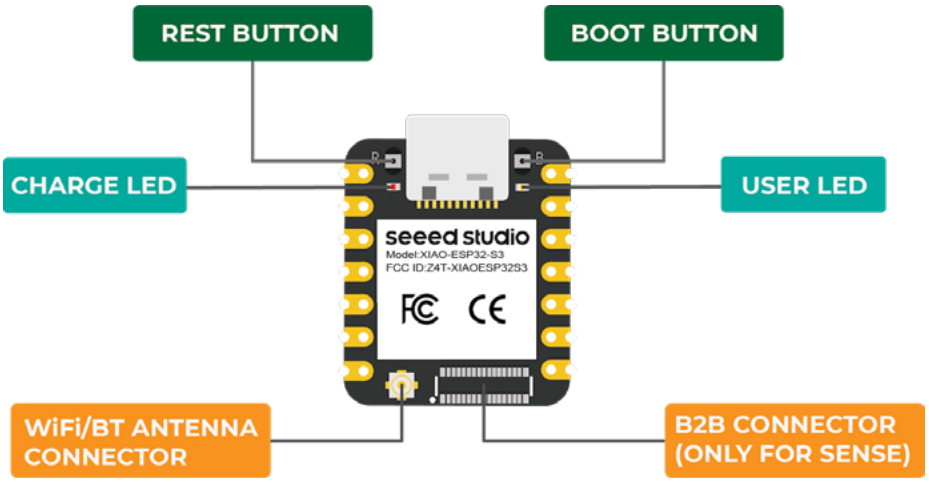
In the "bull-eye" of emerging technologies, radar is the Edge Computer Vision, and when we talk about Machine Learning (ML) applied to vision, the first thing that comes to mind is **Image Classification**, a kind of ML "Hello World"!



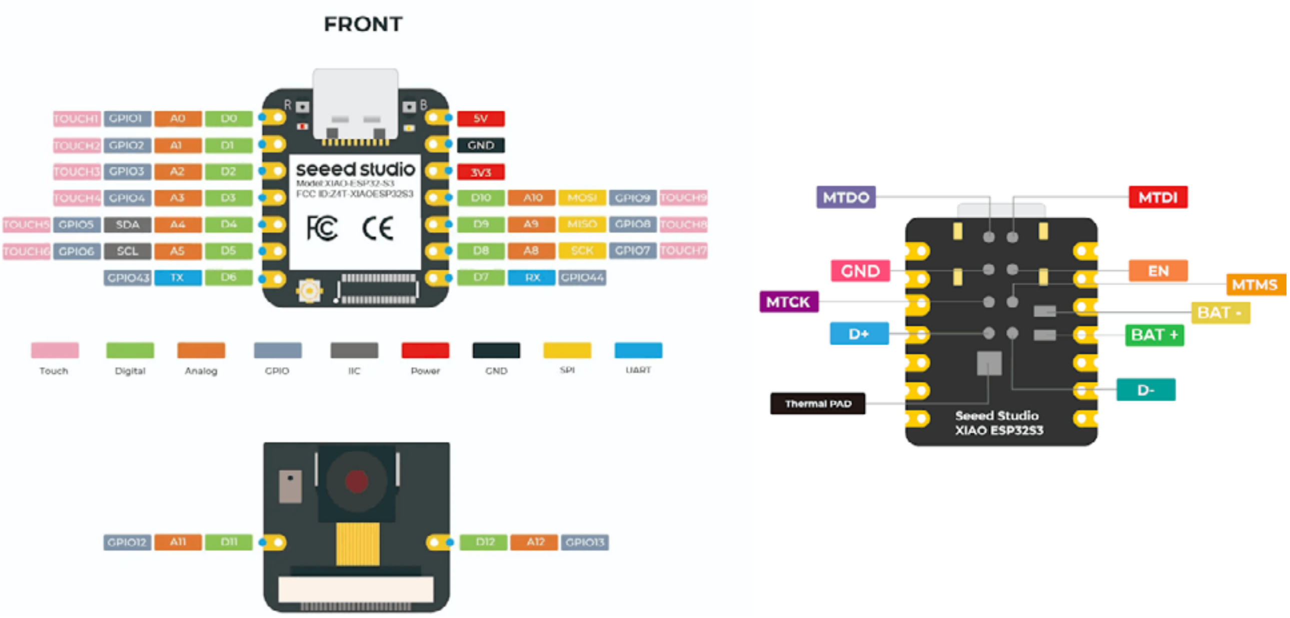
Seeed Studio released a new affordable development board, the XIAO ESP32S3 Sense, which integrates a camera sensor, digital microphone, and SD card support. Combining embedded ML computing power and photography capability, this development board is a great tool to start with TinyML (intelligent voice and vision AI).

XIAO ESP32S3 Sense Main Features

- **Powerful MCU Board:** Incorporate the ESP32S3 32-bit, dual-core, Xtensa processor chip operating up to 240 MHz, mounted multiple development ports, Arduino / MicroPython supported
- **Advanced Functionality:** Detachable OV2640 camera sensor for 1600 * 1200 resolution, compatible with OV5640 camera sensor, integrating an additional digital microphone
- **Elaborate Power Design:** Lithium battery charge management capability offers four power consumption models, which allows for deep sleep mode with power consumption as low as 14μA
- **Great Memory for more Possibilities:** Offer 8MB PSRAM and 8MB FLASH, supporting SD card slot for external 32GB FAT memory
- **Outstanding RF performance:** Support 2.4GHz Wi-Fi and BLE dual wireless communication, support 100m+ remote communication when connected with U.FL antenna
- **Thumb-sized Compact Design:** 21 x 17.5mm, adopting the classic form factor of XIAO, suitable for space-limited projects like wearable devices



Below is the general board pinout:



For more details, please refer to the Seeed Studio WiKi page:

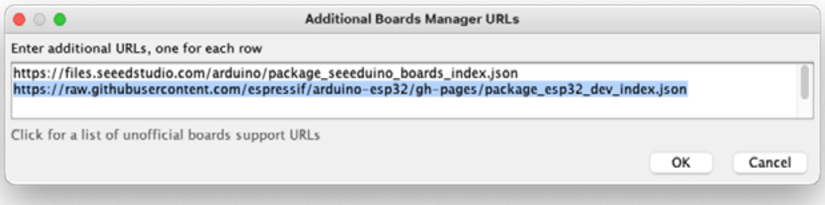
https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/

4.4.3 Installing the XIAO ESP32S3 Sense on Arduino IDE

On Arduino IDE, navigate to **File > Preferences**, and fill in the URL:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

on the field ==> **Additional Boards Manager URLs**



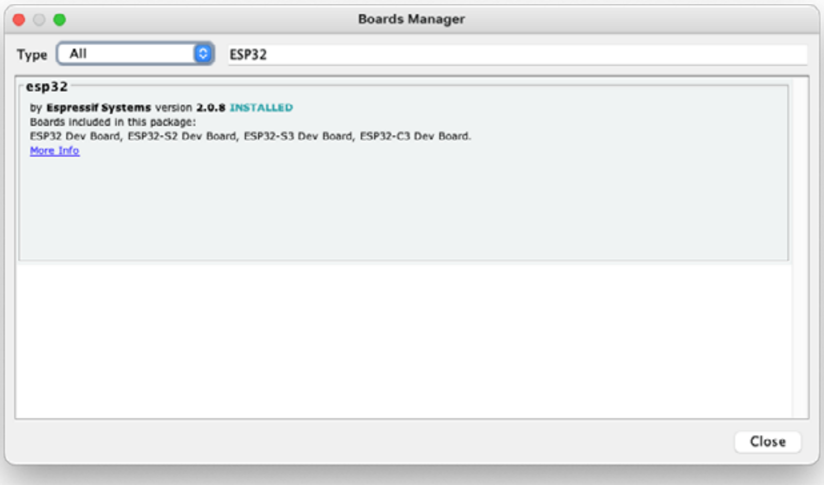
Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
```

Click for a list of unofficial boards support URLs

Next, open boards manager. Go to Tools > Board > Boards Manager… and enter with esp32. Select and install the most updated and stable package (avoid alpha versions) :



Boards Manager

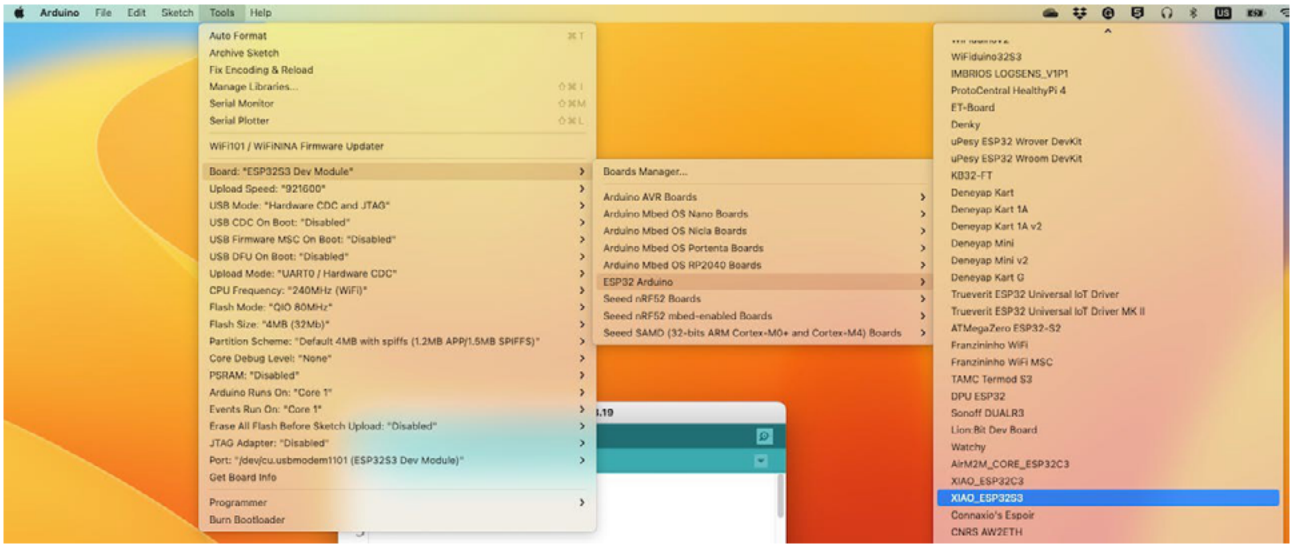
Type: All — ESP32

esp32
by **Espressif Systems** version **2.0.8** INSTALLED
Boards included in this package:
ESP32 Dev Board, ESP32-S2 Dev Board, ESP32-S3 Dev Board, ESP32-C3 Dev Board.
More Info

- Attention -

> *Alpha versions (for example, 3.x-alpha) do not work correctly with the XIAO and Edge Impulse. Use the last stable version (for example, 2.0.11) instead.*

On **Tools**, select the Board (**XIAO ESP32S3**):



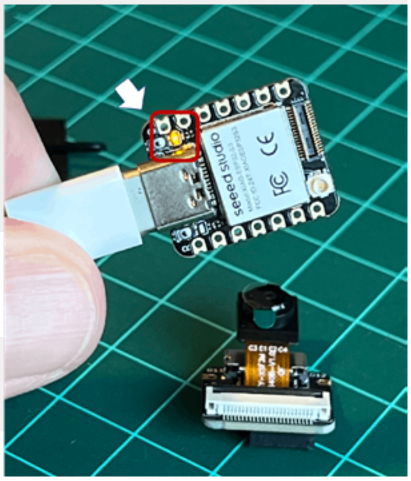
Last but not least, choose the **Port** where the ESP32S3 is connected.

That is it! The device should be OK. Let's do some tests.

4.4.4 Testing the board with BLINK

The XIAO ESP32S3 Sense has a built-in LED that is connected to GPIO21. So, you can run the blink sketch as it is (using the `LED_BUILTIN` Arduino constant) or by changing the Blink sketch accordingly:

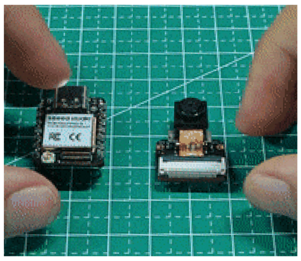
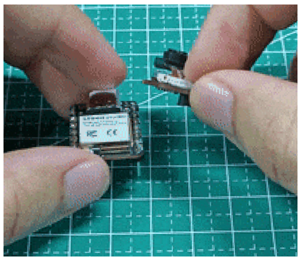
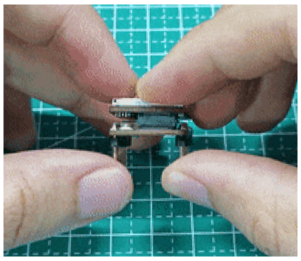
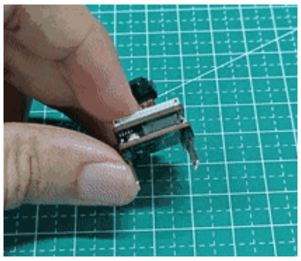
```
#define LED_BUILT_IN 21

void setup() {
  pinMode(LED_BUILT_IN, OUTPUT); // Set the pin as output
}

// Remember that the pin work with inverted logic
// LOW to Turn on and HIGH to turn off
void loop() {
  digitalWrite(LED_BUILT_IN, LOW); //Turn on
  delay (1000); //Wait 1 sec
  digitalWrite(LED_BUILT_IN, HIGH); //Turn off
  delay (1000); //Wait 1 sec
}
```



Note that the pins work with inverted logic: LOW to Turn on and HIGH to turn off.

4.4.5 Connecting Sense module (Expansion Board)

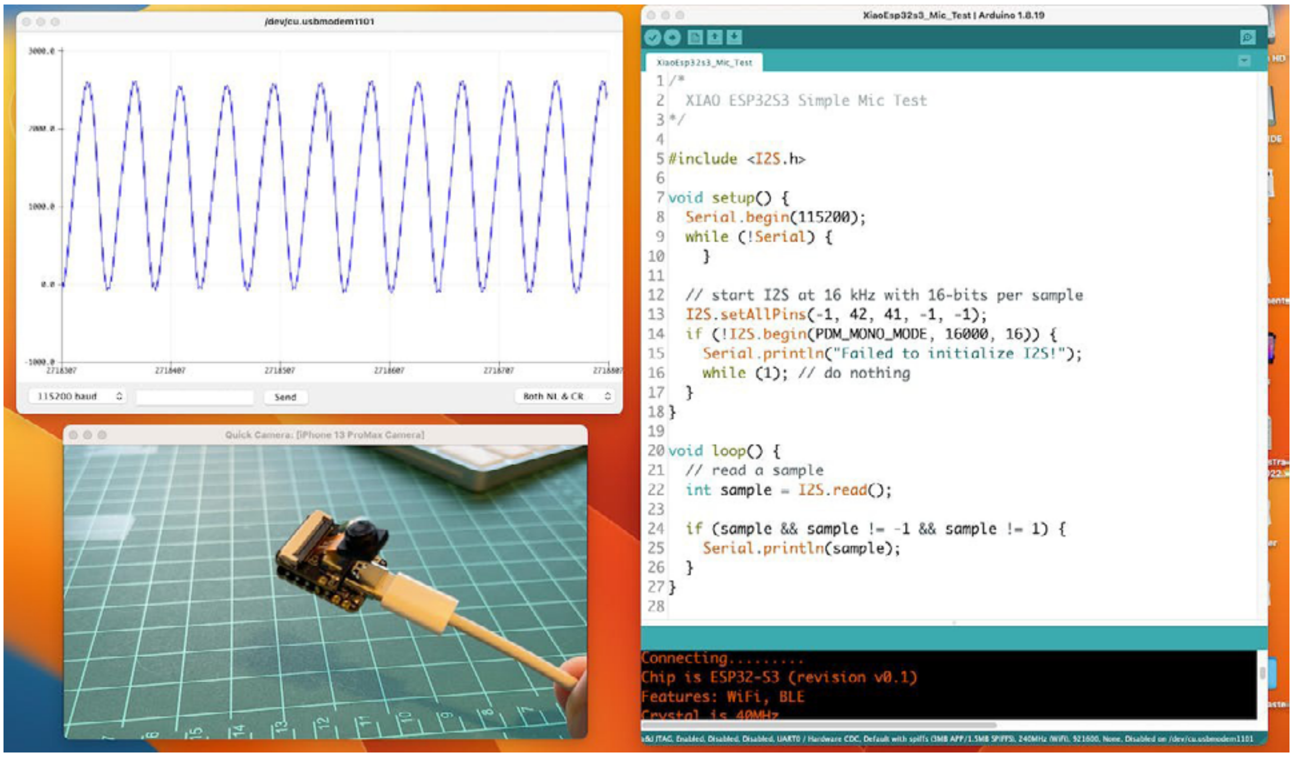
When purchased, the expansion board is separated from the main board, but installing the expansion board is very simple. You need to align the connector on the expansion board with the B2B connector on the XIAO ESP32S3, press it hard, and when you hear a "click," the installation is complete.

As commented in the introduction, the expansion board, or the "sense" part of the device, has a 1600x1200 OV2640 camera, an SD card slot, and a digital microphone.

4.4.6 Microphone Test

Let's start with sound detection. Go to the GitHub project and download the sketch: XIAOEsp2s3_Mic_Test and run it on the Arduino IDE:



```
/*
  XIAO ESP32S3 Simple Mic Test
*/

#include <I2S.h>

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    }

  // start I2S at 16 kHz with 16-bits per sample
  I2S.setAllPins(-1, 42, 41, -1, -1);
  if (!I2S.begin(PDM_MONO_MODE, 16000, 16)) {
    Serial.println("Failed to initialize I2S!");
    while (1); // do nothing
  }
}

void loop() {
  // read a sample
  int sample = I2S.read();

  if (sample && sample != -1 && sample != 1) {
    Serial.println(sample);
  }
}

```

```
Connecting.........
Chip is ESP32-S3 (revision v0.1)
Features: WiFi, BLE
Crystal is 40MHz
```

When producing sound, you can verify it on the Serial Plotter.

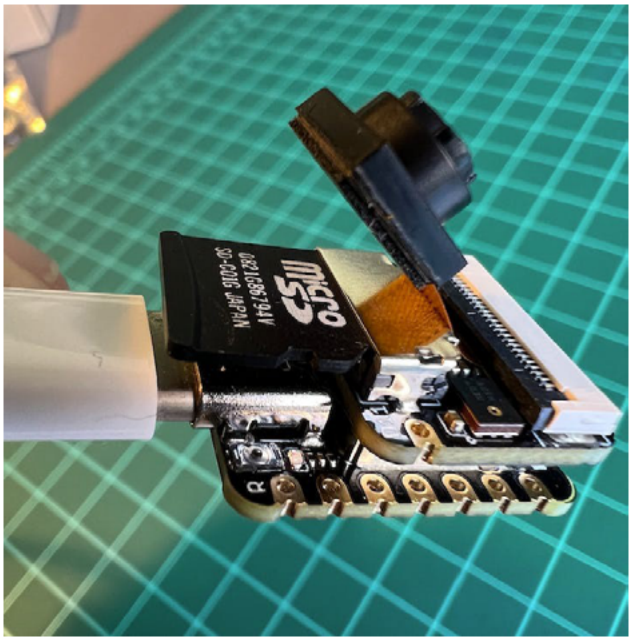
Save recorded sound (.wav audio files) to a microSD card.

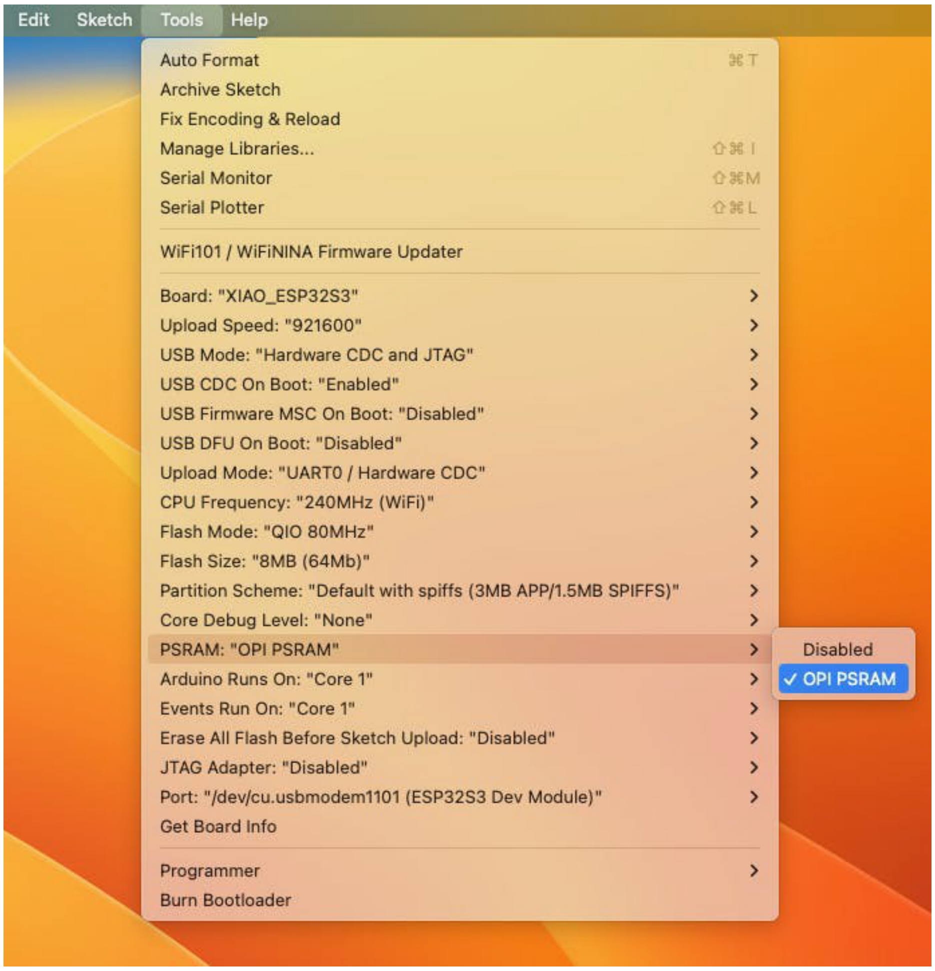
Now, the onboard SD Card reader can save .wav audio files. For that, we need to habilitate the XIAO PSRAM.

> *ESP32-S3 has only a few hundred kilobytes of internal RAM on the MCU chip. It can be insufficient for some purposes so that ESP32-S3 can use up to 16 MB of external PSRAM (Psuedostatic RAM) connected with the SPI flash chip. The external memory is incorporated in the memory map and, with certain restrictions, is usable in the same way as internal data RAM.*

For a start, Insert the SD Card on the XIAO as shown in the photo below (the SD Card should be formatted to **FAT32**).

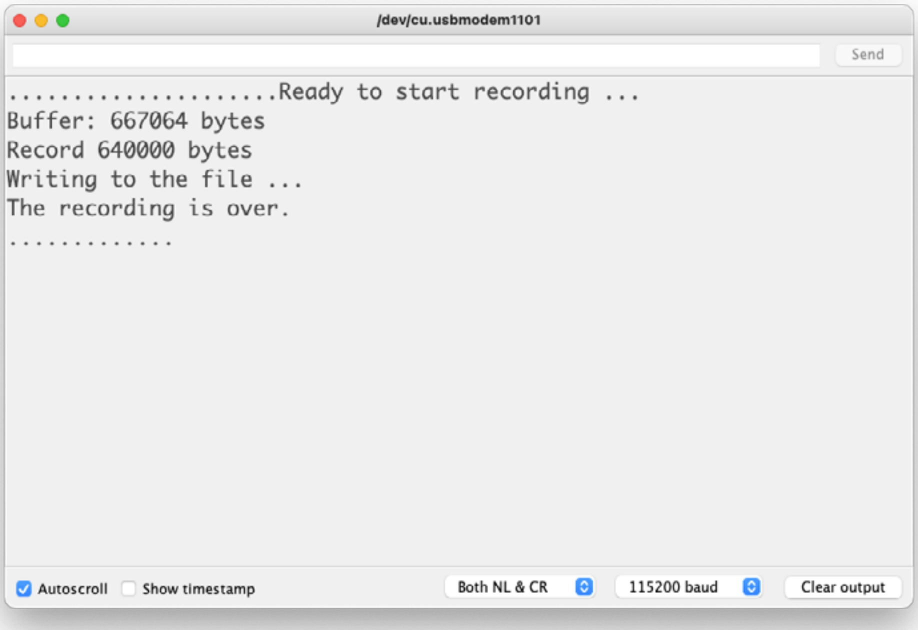
- Download the sketch Wav_Record, which you can find on GitHub.
- To execute the code (Wav Record), it is necessary to use the PSRAM function of the ESP-32 chip, so turn it on before uploading.: Tools>PSRAM: “OPI PSRAM”>OPI PSRAM
- Download the sketch Wav_Record, which you can find on GitHub.
- To execute the code (Wav Record), it is necessary to use the PSRAM function of the ESP-32 chip, so turn it on before uploading.: Tools>PSRAM: “OPI PSRAM”>OPI PSRAM





```
Board: "XIAO_ESP32S3"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
USB CDC On Boot: "Enabled"
USB Firmware MSC On Boot: "Disabled"
USB DFU On Boot: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
CPU Frequency: "240MHz (WiFi)"
Flash Mode: "QIO 80MHz"
Flash Size: "8MB (64Mb)"
Partition Scheme: "Default with spiffs (3MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "OPI PSRAM"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Erase All Flash Before Sketch Upload: "Disabled"
JTAG Adapter: "Disabled"
Port: "/dev/cu.usbmodem1101 (ESP32S3 Dev Module)"
```

- Run the code `Wav_Record.ino`
- This program is executed only once after the user turns on the serial monitor, recording for 20 seconds and saving the recording file to a microSD card as “arduino_rec.wav.”
- When the “.” is output every 1 second in the serial monitor, the program execution is finished, and you can play the recorded sound file with the help of a card reader.



```
.....................Ready to start recording ...
Buffer: 667064 bytes
Record 640000 bytes
Writing to the file ...
The recording is over.
.............
```

The sound quality is excellent!

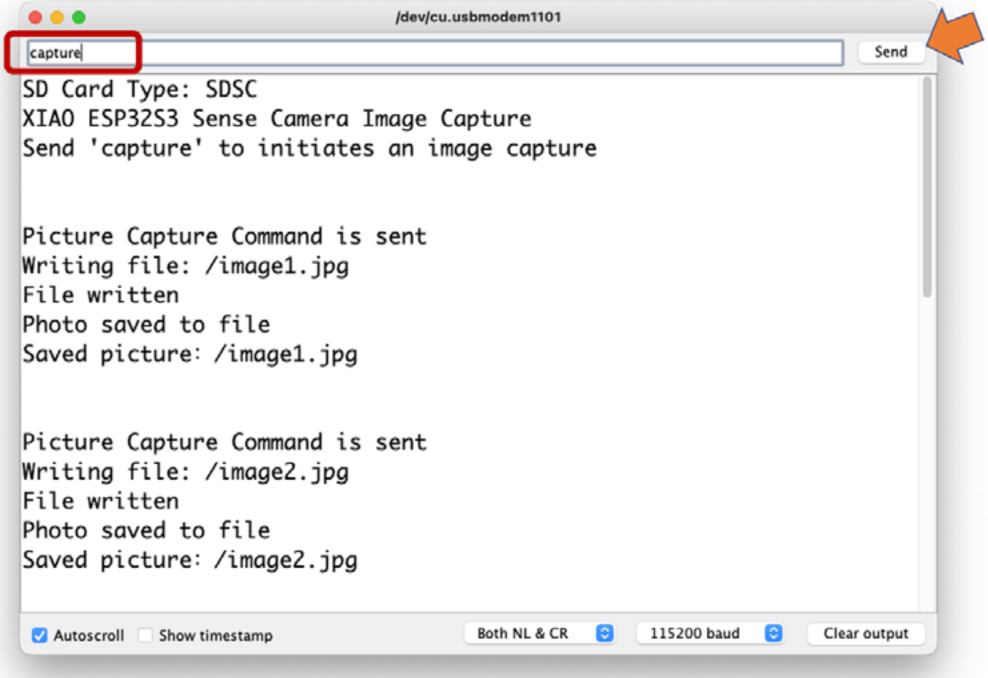
> *The explanation of how the code works is beyond the scope of this tutorial, but you can find an excellent description on the wiki page.*

4.4.7 Testing the Camera

To test the camera, you should download the folder take_photos_command from GitHub. The folder contains the sketch (.ino) and two .h files with camera details.

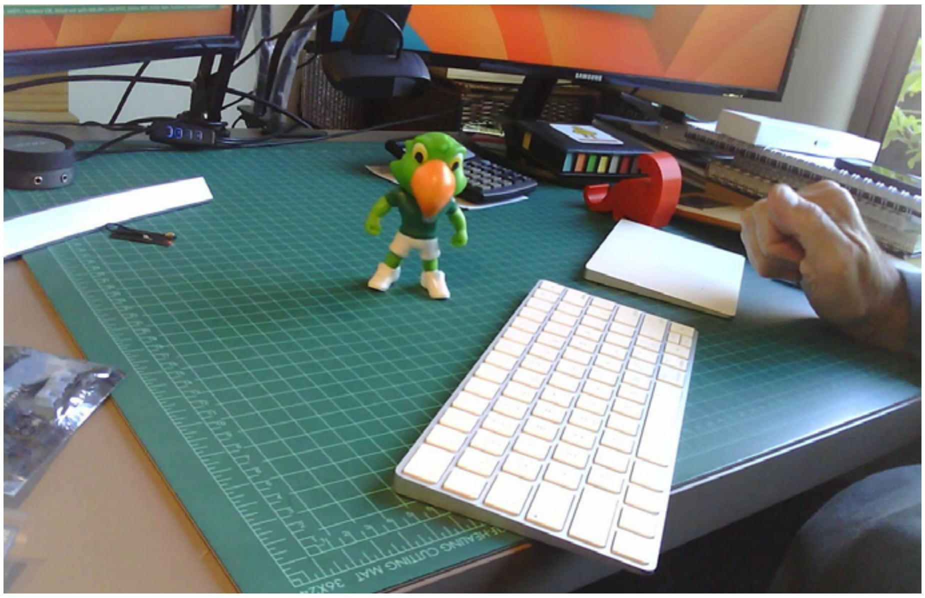
- Run the code: `take_photos_command.ino`. Open the Serial Monitor and send the command capture to `capture` and save the image on the SD Card:

Verify that `[Both NL & CR]` is selected on Serial Monitor.



```
SD Card Type: SDSC
XIAO ESP32S3 Sense Camera Image Capture
Send 'capture' to initiates an image capture


Picture Capture Command is sent
Writing file: /image1.jpg
File written
Photo saved to file
Saved picture: /image1.jpg


Picture Capture Command is sent
Writing file: /image2.jpg
File written
Photo saved to file
Saved picture: /image2.jpg
```

Here is an example of a taken photo:

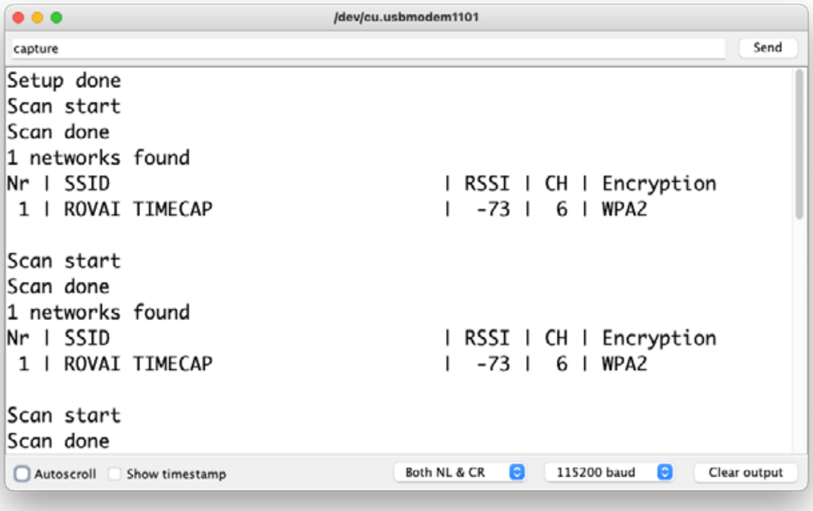


4.4.8 Testing WiFi

One of the differentiators of the XIAO ESP32S3 is its WiFi capability. So, let's test its radio, scanning the wifi networks around it. You can do it by running one of the code examples on the board.

Go to Arduino IDE Examples and look for **WiFI ==> WiFIScan**

On the Serial monitor, you should see the wifi networks (SSIDs and RSSIs) in the range of your device. Here is what I got in the lab:



```
Setup done
Scan start
Scan done
1 networks found
Nr | SSID                             | RSSI | CH | Encryption
 1 | ROVAI TIMECAP                    |  -73 |  6 | WPA2

Scan start
Scan done
1 networks found
Nr | SSID                             | RSSI | CH | Encryption
 1 | ROVAI TIMECAP                    |  -73 |  6 | WPA2

Scan start
Scan done
```

Simple WiFi Server (Turning LED ON/OFF)

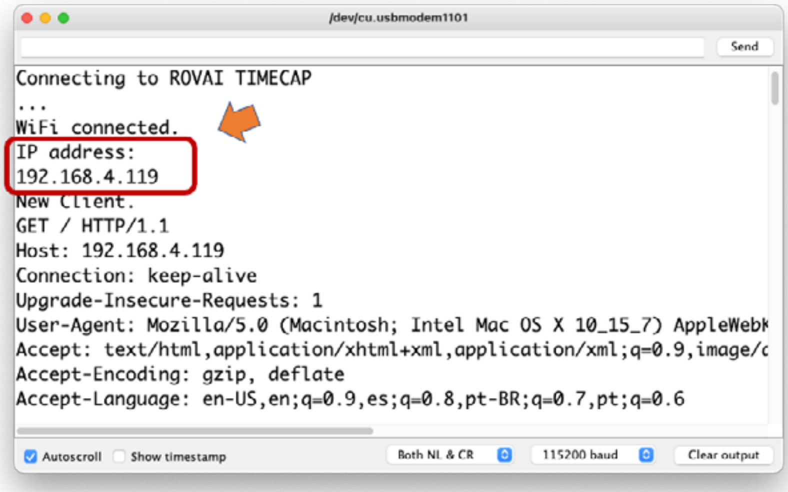
Let's test the device's capability to behave as a WiFi Server. We will host a simple page on the device that sends commands to turn the XIAO built-in LED ON and OFF.

Like before, go to GitHub to download the folder with the sketch SimpleWiFiServer.

Before running the sketch, you should enter your network credentials:

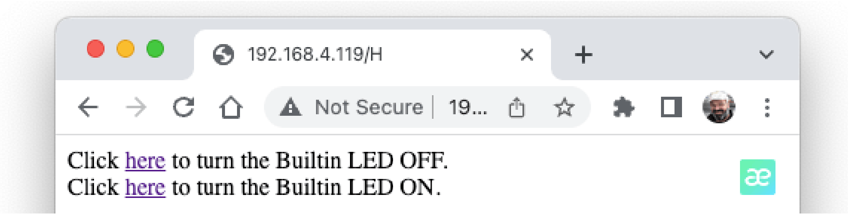
```
const char* ssid     = "Your credentials here";
const char* password = "Your credentials here";
```

You can monitor how your server is working with the Serial Monitor.



```
Connecting to ROVAI TIMECAP
...
WiFi connected.
IP address:
192.168.4.119
New Client.
GET / HTTP/1.1
Host: 192.168.4.119
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebk
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/a
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,es;q=0.8,pt-BR;q=0.7,pt;q=0.6
```

Take the IP address and enter it on your browser:



Click here to turn the Builtin LED OFF.
Click here to turn the Builtin LED ON.

You will see a page with links that can turn the built-in LED of your XIAO ON and OFF.

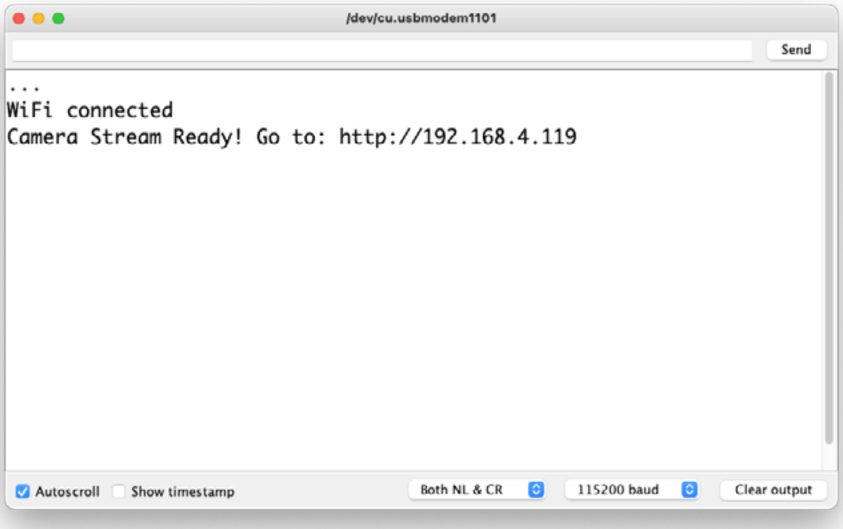
Streaming video to Web

Now that you know that you can send commands from the webpage to your device, let's do the reverse. Let's take the image captured by the camera and stream it to a webpage:

Download from GitHub the folder that contains the code: XIAO-ESP32S3-Streeming_Video.ino.

> *Remember that the folder contains the.ino file and a couple of .h files necessary to handle the camera.*

Enter your credentials and run the sketch. On the Serial monitor, you can find the page address to enter in your browser:

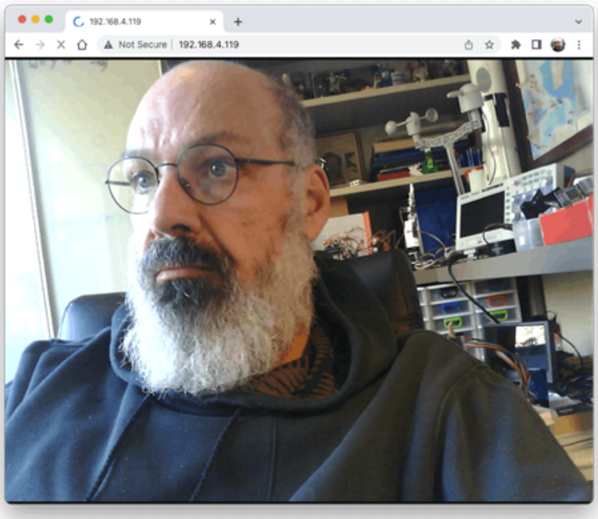


```
...
WiFi connected
Camera Stream Ready! Go to: http://192.168.4.119
```

Open the page on your browser (wait a few seconds to start the streaming). That's it.

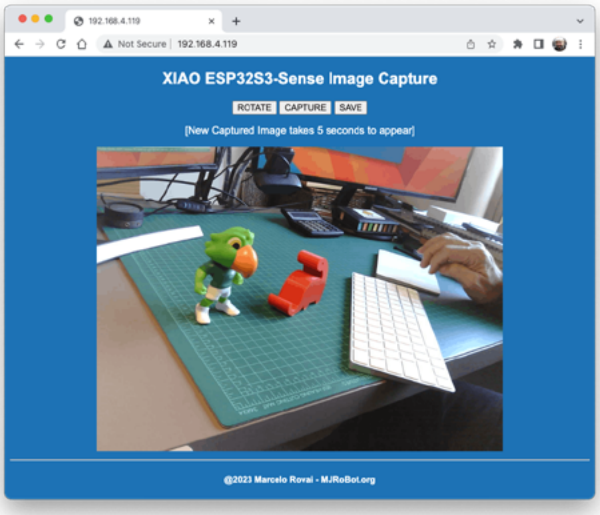
Streamlining what your camera is "seen" can be important when you position it to capture a dataset for an ML project (for example, using the code "take_phots_commands.ino".

Of course, we can do both things simultaneously: show what the camera sees on the page and send a command to capture and save the image on the SD card. For that, you can use the code Camera_HTTP_Server_STA, which can be downloaded from GitHub.



The program will do the following tasks:

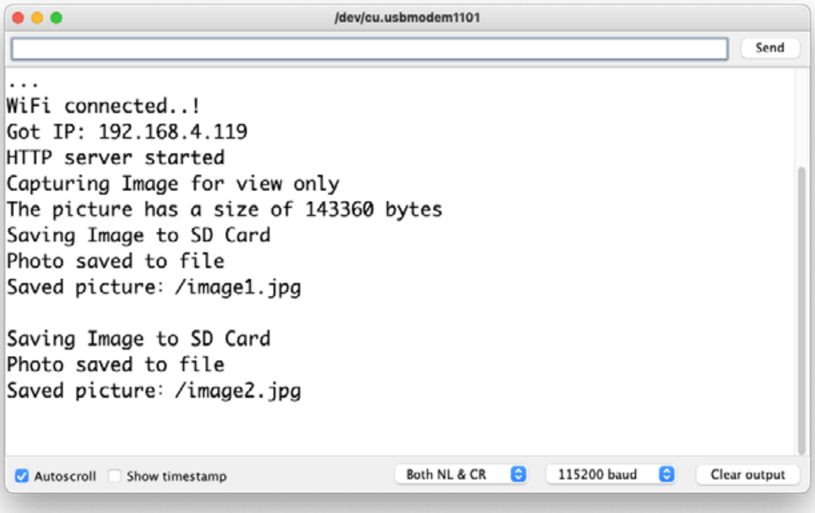
- Set the camera to JPEG output mode.
- Create a web page (for example ==> http://192.168.4.119//). The correct address will be displayed on the Serial Monitor.
- If server.on ("/capture", HTTP_GET, serverCapture), the program takes a photo and sends it to the Web.
- It is possible to rotate the image on webPage using the button [ROTATE]
- The command [CAPTURE] only will preview the image on the webpage, showing its size on the Serial Monitor
- The [SAVE] command will save an image on the SD Card and show the image on the browser.
- Saved images will follow a sequential naming (image1.jpg, image2.jpg.



XIAO ESP32S3-Sense Image Capture

[New Captured Image takes 5 seconds to appear]

@2023 Marcelo Rovai - MJRoBot.org



```
...
WiFi connected..!
Got IP: 192.168.4.119
HTTP server started
Capturing Image for view only
The picture has a size of 143360 bytes
Saving Image to SD Card
Photo saved to file
Saved picture: /image1.jpg

Saving Image to SD Card
Photo saved to file
Saved picture: /image2.jpg
```

☑ Autoscroll □ Show timestamp Both NL & CR 115200 baud

> *This program can be used for an image dataset capture with an Image Classification project.*

Inspect the code; it will be easier to understand how the camera works. This code was developed based on the great Rui Santos Tutorial ESP32-CAM Take Photo and Display in Web Server, which I invite all of you to visit.

Using the CameraWebServer

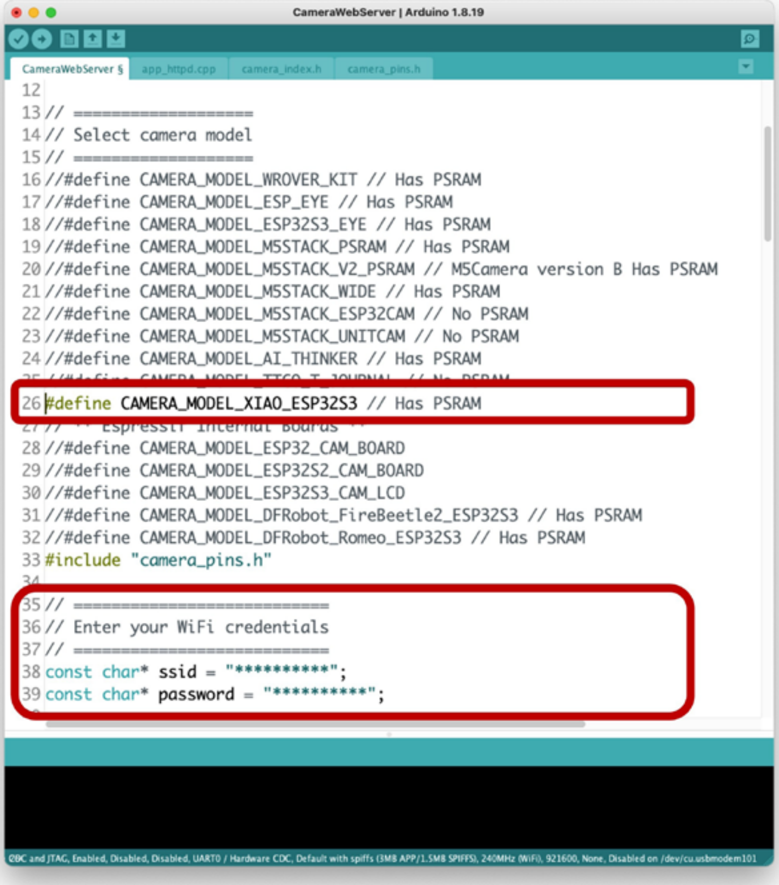
In the Arduino IDE, go to `File > Examples > ESP32 > Camera`, and select `CameraWebServer`

You also should comment on all cameras' models, except the XIAO model pins:

`#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM`

And do not forget the `Tools` to enable the PSRAM.

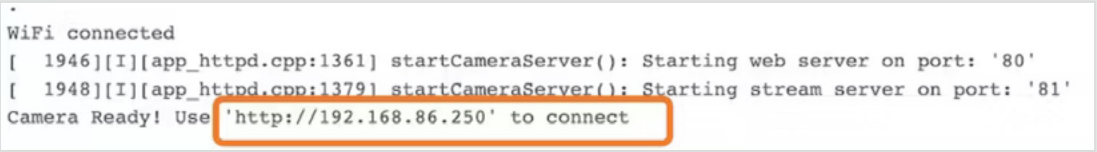
Enter your wifi credentials and upload the code to the device:



```

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
//#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid = "**********";
const char* password = "**********";
```

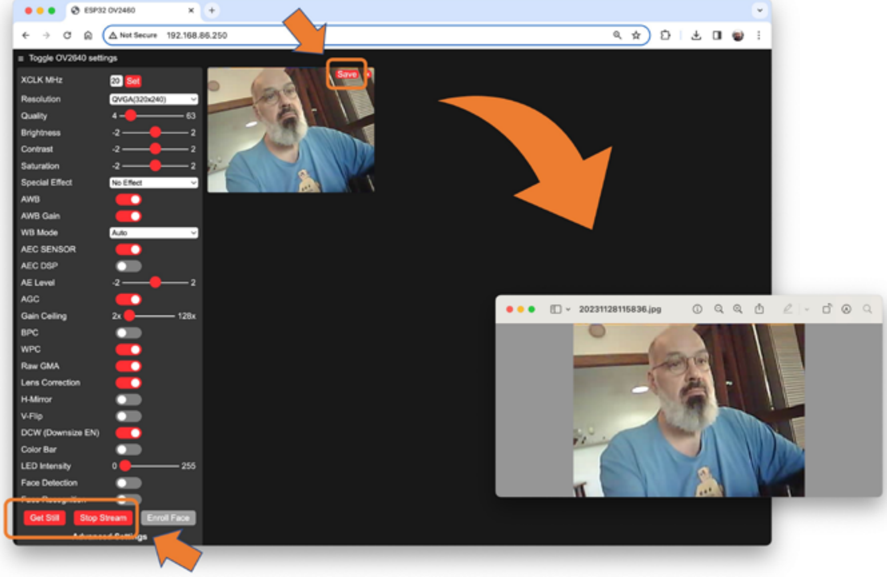
ØBC and JTAG, Enabled, Disabled, Disabled, UART0 / Hardware CDC, Default with spiffs (3MB APP/1.5MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on /dev/cu.usbmodem101

If the code is executed correctly, you should see the address on the Serial Monitor:



```
.
WiFi connected
[  1946][I][app_httpd.cpp:1361] startCameraServer(): Starting web server on port: '80'
[  1948][I][app_httpd.cpp:1379] startCameraServer(): Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.86.250' to connect
```

Copy the address on your browser and wait for the page to be uploaded. Select the camera resolution (for example, QVGA) and select [START STREAM]. Wait for a few seconds/minutes, depending on your connection. You can save an image on your computer download area using the [Save] button.



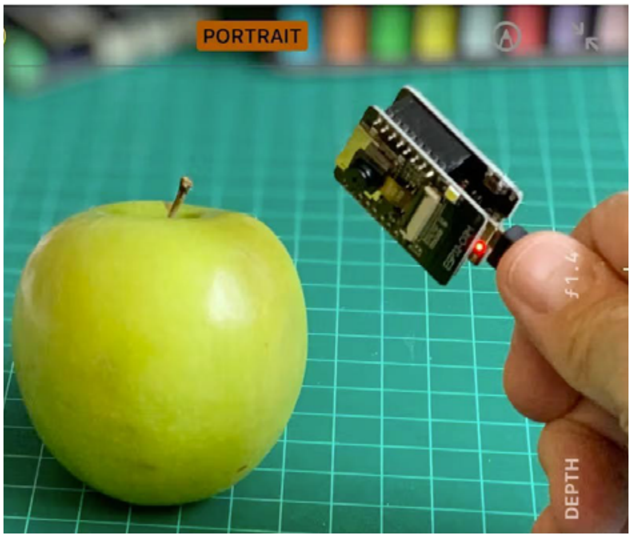
XCLK MHz 20 Set
Resolution QVGA(320x240)
Quality 4 – 63
Brightness -2 – 2
Contrast -2 – 2
Saturation -2 – 2
Special Effect No Effect
AWB
AWB Gain
WB Mode Auto
AEC SENSOR
AEC DSP
AE Level -2 – 2
AGC
Gain Ceiling 2x – 128x
BPC
WPC
Raw GMA
Lens Correction
H-Mirror
V-Flip
DCW (Downsize EN)
Color Bar
LED Intensity 0 – 255
Face Detection
Face Recognition

That's it! You can save the images directly on your computer to be used on projects.

4.4.9 Fruits versus Veggies - A TinyML Image Classification Project

Now that we have an embedded camera running, it is time to try image classification. For comparative motive, we will replicate the same image classification project developed to be used with an old ESP2-CAM:

ESP32-CAM: TinyML Image Classification - Fruits vs Veggies



The whole idea of our project will be to train a model and proceed with inference on the XIAO ESP32S3 Sense. For training, we should find some data (**in fact, tons of data!**).

But first of all, we need a goal! What do we want to classify?

With TinyML, a set of techniques associated with machine learning inference on embedded devices, we should limit the classification to three or four categories due to limitations (mainly memory). We will differentiate **apples** from **bananas** and **potatoes** (you can try other categories).

So, let's find a specific dataset that includes images from those categories. Kaggle is a good start:

https://www.kaggle.com/kritikseth/fruit-and-vegetable-image-recognition

This dataset contains images of the following food items:

- **Fruits** - banana, apple, pear, grapes, orange, kiwi, watermelon, pomegranate, pineapple, mango.
- **Vegetables** - cucumber, carrot, capsicum, onion, potato, lemon, tomato, radish, beetroot, cabbage, lettuce, spinach, soybean, cauliflower, bell pepper, chili pepper, turnip, corn, sweetcorn, sweet potato, paprika, jalepeño, ginger, garlic, peas, eggplant.

Each category is split into the **train** (100 images), **test** (10 images), and **validation** (10 images).

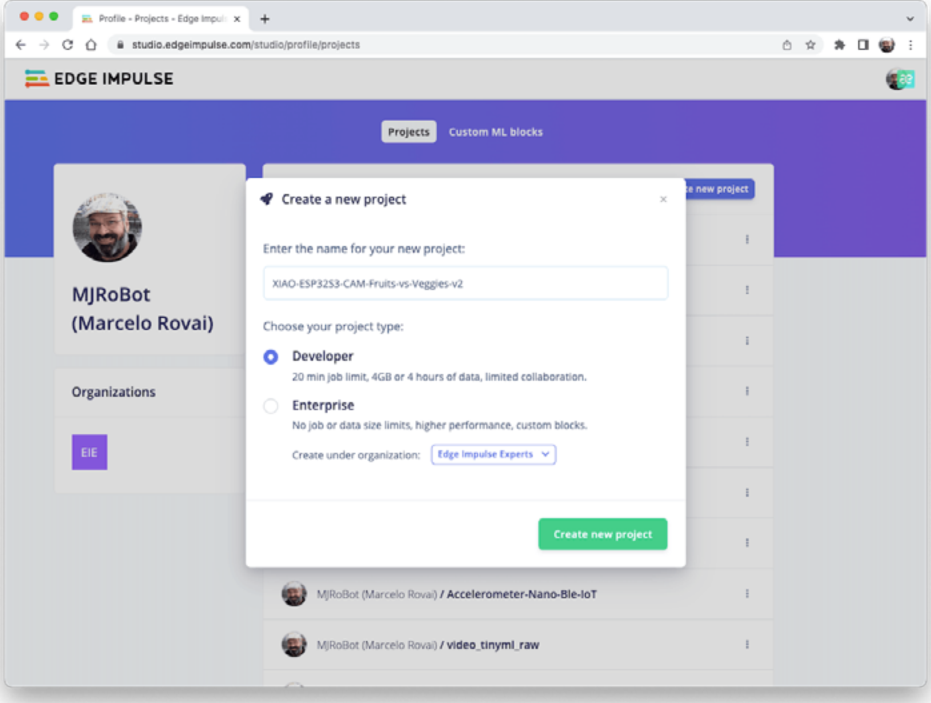
- Download the dataset from the Kaggle website to your computer.

> *Optionally, you can add some fresh photos of bananas, apples, and potatoes from your home kitchen, using, for example, the codes discussed in the last section.*

4.4.10 Training the model with Edge Impulse Studio

We will use the Edge Impulse Studio to train our model. As you know, Edge Impulse is a leading development platform for machine learning on edge devices.

Enter your account credentials (or create a free account) at Edge Impulse. Next, create a new project:


studio.edgeimpulse.com/studio/profile/projects
EDGE IMPULSE
Projects
Custom ML blocks
MJRoBot (Marcelo Rovai)
Organizations
EIE
MJRoBot (Marcelo Rovai) / Accelerometer-Nano-Ble-IoT
MJRoBot (Marcelo Rovai) / video_tinyml_raw

Create a new project

Enter the name for your new project:

XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v2

Choose your project type:

● Developer
20 min job limit, 4GB or 4 hours of data, limited collaboration.

○ Enterprise
No job or data size limits, higher performance, custom blocks.

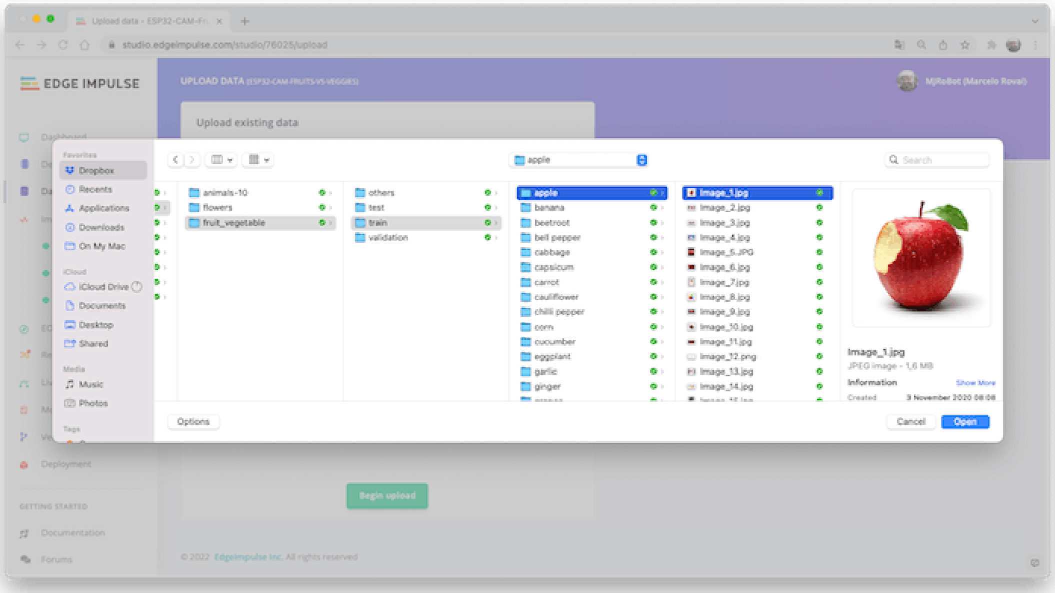
Create under organization: Edge Impulse Experts

Create new project

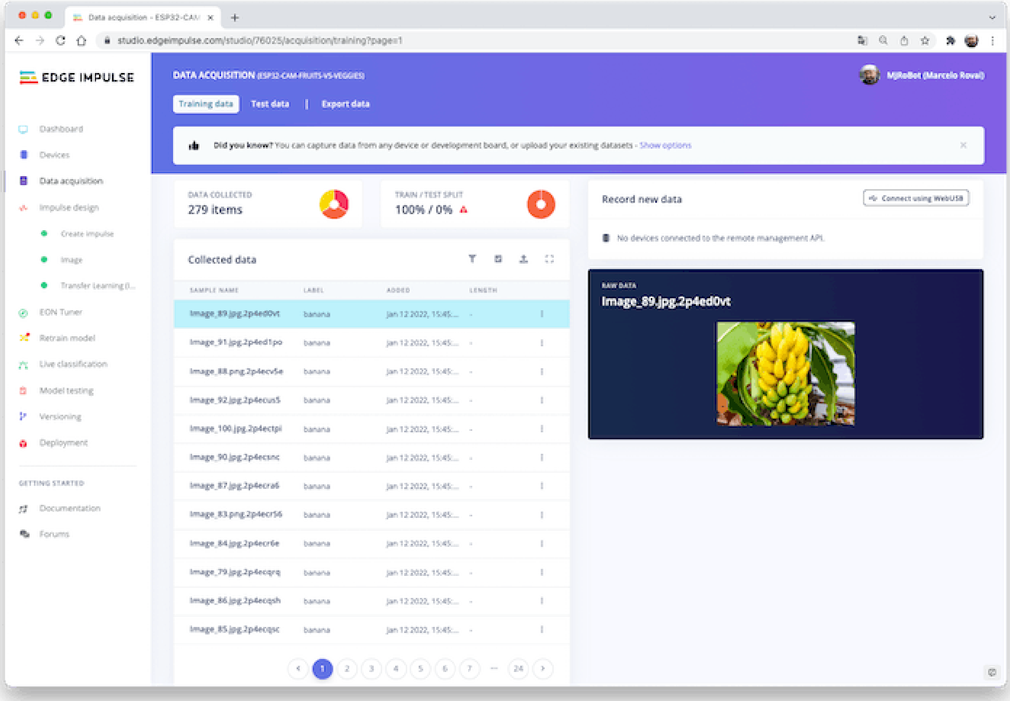
Data Acquisition

Next, on the UPLOAD DATA section, upload from your computer the files from chosen categories:

It would be best if you now had your training dataset split into three classes of data:



It would be best if you now had your training dataset split into three classes of data:



DATA ACQUISITION (ESP32-CAM-FRUITS-VS-VEGGIES)

Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLECTED
279 items

TRAIN / TEST SPLIT
100% / 0%

Record new data

No devices connected to the remote management API.

RAW DATA
Image_89.jpg.2p4ed0vt

Collected data

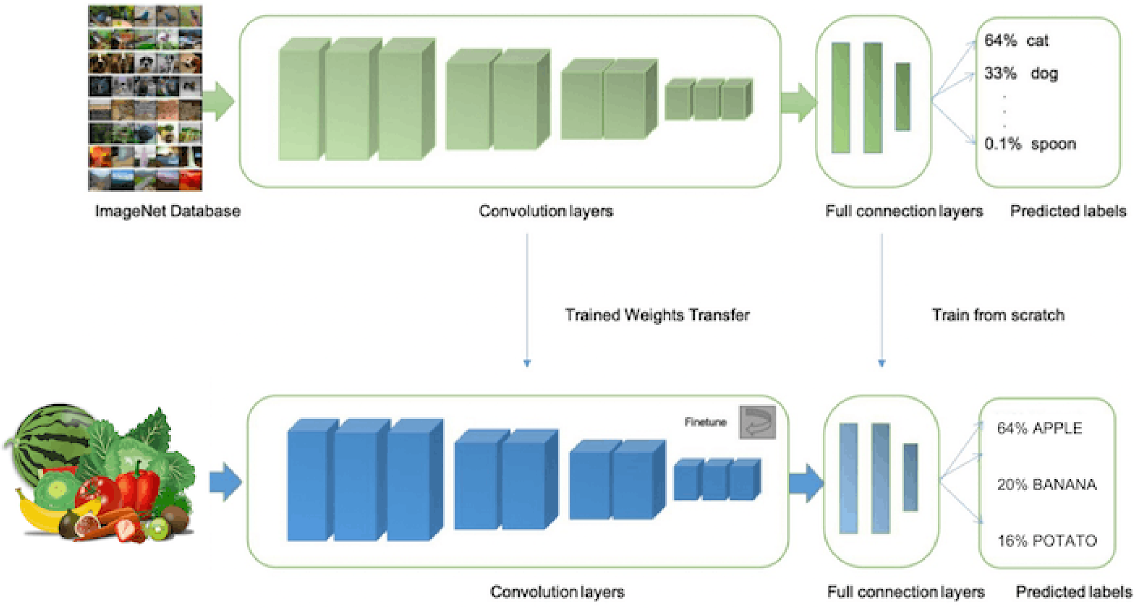
SAMPLE NAME	LABEL	ADDED	LENGTH
Image_89.jpg.2p4ed0vt	banana	Jan 12 2022, 15:45:...	-
Image_91.jpg.2p4ed1po	banana	Jan 12 2022, 15:45:...	-
Image_88.png.2p4ecv5e	banana	Jan 12 2022, 15:45:...	-
Image_92.jpg.2p4ecus5	banana	Jan 12 2022, 15:45:...	-
Image_100.jpg.2p4ectpi	banana	Jan 12 2022, 15:45:...	-
Image_90.jpg.2p4ecsnc	banana	Jan 12 2022, 15:45:...	-
Image_87.jpg.2p4ecra6	banana	Jan 12 2022, 15:45:...	-
Image_83.png.2p4ecr56	banana	Jan 12 2022, 15:45:...	-
Image_84.jpg.2p4ecr6e	banana	Jan 12 2022, 15:45:...	-
Image_79.jpg.2p4ecqrq	banana	Jan 12 2022, 15:45:...	-
Image_86.jpg.2p4ecqsh	banana	Jan 12 2022, 15:45:...	-
Image_85.jpg.2p4ecqsc	banana	Jan 12 2022, 15:45:...	-

You can upload extra data for further model testing or split the training data. I will leave it as it is to use the most data possible.

Impulse Design

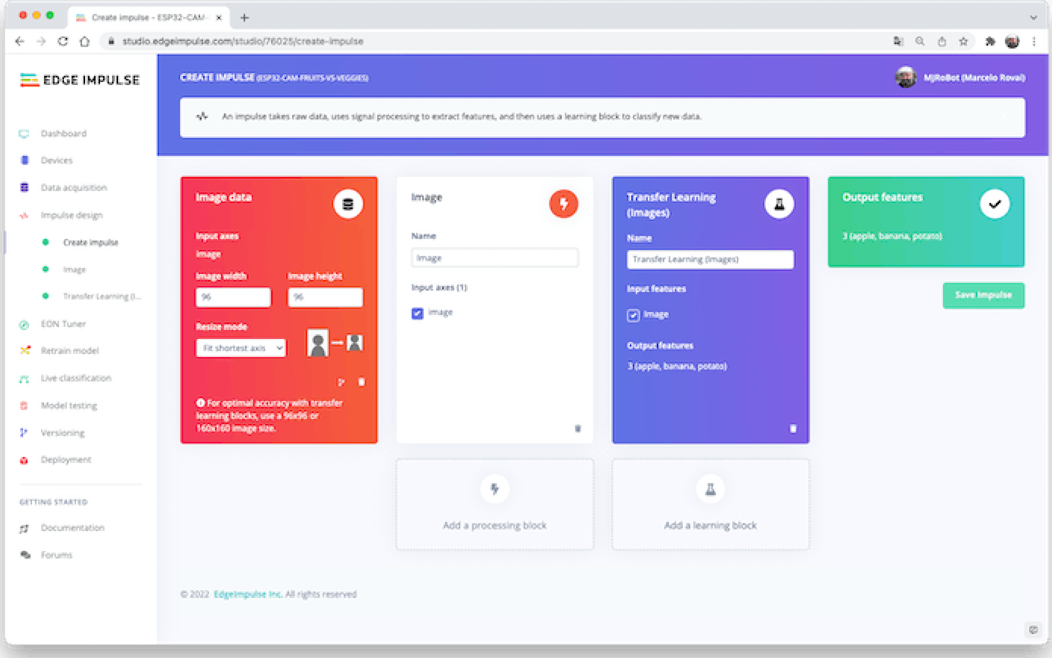
An impulse takes raw data (in this case, images), extracts features (resize pictures), and then uses a learning block to classify new data.

Classifying images is the most common use of deep learning, but much data should be used to accomplish this task. We have around 90 images for each category. Is this number enough? Not at all! We will need thousands of images to "teach or model" to differentiate an apple from a banana. But, we can solve this issue by re-training a previously trained model with thousands of images. We call this technique "Transfer Learning" (TL).



With TL, we can fine-tune a pre-trained image classification model on our data, performing well even with relatively small image datasets (our case).

So, starting from the raw images, we will resize them (96x96) pixels and feed them to our Transfer Learning block:



An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Image data
Input axes: image
Image width: 96
Image height: 96
Resize mode: Fit shortest axis
For optimal accuracy with transfer learning blocks, use a 96x96 or 160x160 image size.

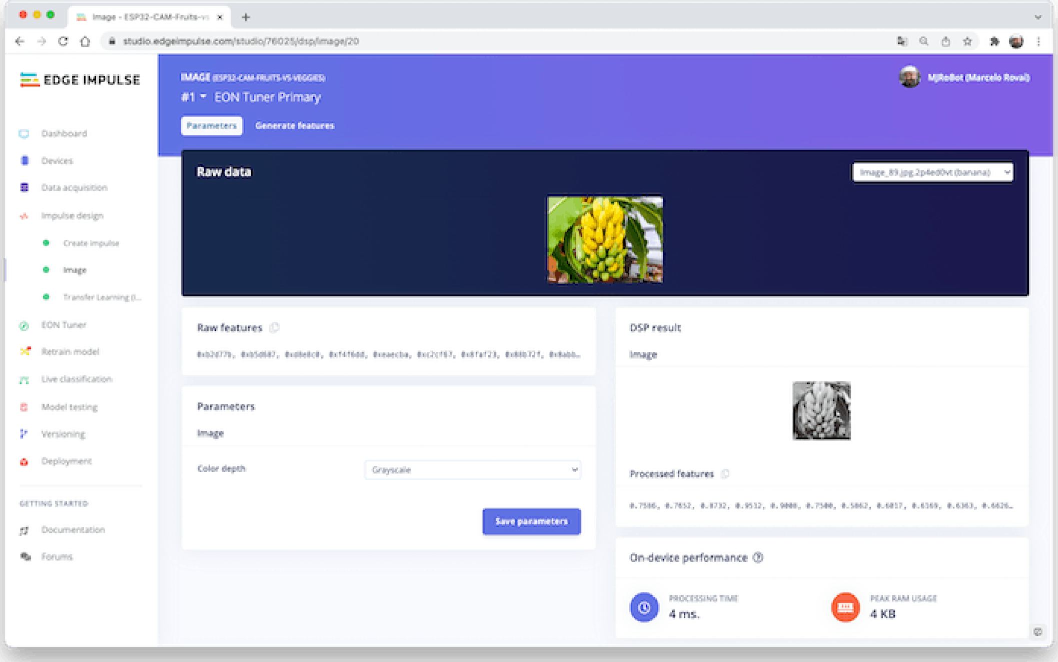
Image
Name: Image
Input axes (1): ☑ image

Transfer Learning (Images)
Name: Transfer Learning (Images)
Input features: ☑ Image
Output features: 3 (apple, banana, potato)

Output features
3 (apple, banana, potato)

Pre-processing (Feature generation)

Besides resizing the images, we can change them to Grayscale or keep the actual RGB color depth. Let's start selecting `Grayscale`. Doing that, each one of our data samples will have dimension 9, 216 features (96x96x1). Keeping RGB, this dimension would be three times bigger. Working with Grayscale helps to reduce the amount of final memory needed for inference.



#1 EON Tuner Primary

image_89.jpg.2p4ed0vt (banana)

Raw features: 0xb2d77b, 0xb5d687, 0xd8e8c8, 0xf4f6dd, 0xeaecba, 0xc2cf67, 0x8faf23, 0x88b72f, 0x8abb…

Parameters
Image
Color depth: Grayscale

Processed features: 0.7586, 0.7652, 0.8732, 0.9512, 0.9008, 0.7580, 0.5862, 0.6017, 0.6169, 0.6363, 0.6626…

PROCESSING TIME: 4 ms.
PEAK RAM USAGE: 4 KB

Do not forget to `[Save parameters]`." This will generate the features to be used in training.

Training (Transfer Learning & Data Augmentation)

In 2007, Google introduced MobileNetV1, a family of general-purpose computer vision neural networks designed with mobile devices in mind to support classification, detection, and more. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of various use cases.

Although the base MobileNet architecture is already tiny and has low latency, many times, a specific use case or application may require the model to be smaller and faster. MobileNet introduces a straightforward parameter α (alpha) called width multiplier to construct these smaller, less computationally expensive models. The role of the width multiplier α is to thin a network uniformly at each layer.

Edge Impulse Studio has available MobileNet V1 (96x96 images) and V2 (96x96 and 160x160 images), with several different **α** values (from 0.05 to 1.0). For example, you will get the highest accuracy with V2, 160x160 images, and α=1.0. Of course, there is a trade-off. The higher the accuracy, the more memory (around 1.3M RAM and 2.6M ROM) will be needed to run the model, implying more latency.

The smaller footprint will be obtained at another extreme with **MobileNet V1** and α=0.10 (around 53.2K RAM and 101K ROM).

When we first published this project to be running on an ESP32-CAM, we stayed at the lower side of possibilities, which guaranteed the inference with small latency but not with high accuracy. For this first pass, we will keep this model design (**MobileNet V1** and α=0.10).

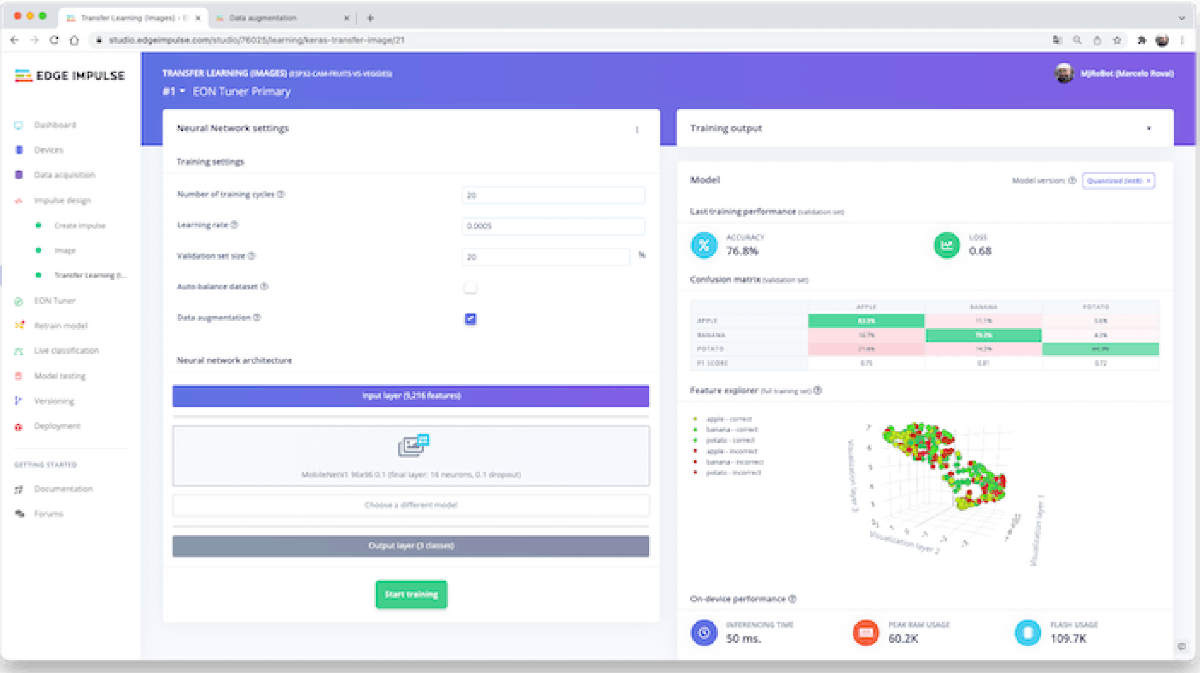
Another necessary technique to use with deep learning is **data augmentation**. Data augmentation is a method that can help improve the accuracy of machine learning models, creating additional artificial data. A data augmentation system makes small, random changes to your training data during the training process (such as flipping, cropping, or rotating the images).

Under the rood, here you can see how Edge Impulse implements a data Augmentation policy on your data:

```
# Implements the data augmentation policy
def augment_image(image, label):
    # Flips the image randomly
    image = tf.image.random_flip_left_right(image)

    # Increase the image size, then randomly crop it down to
    # the original dimensions
    resize_factor = random.uniform(1, 1.2)
    new_height = math.floor(resize_factor * INPUT_SHAPE[0])
    new_width = math.floor(resize_factor * INPUT_SHAPE[1])
    image = tf.image.resize_with_crop_or_pad(image, new_height, new_width)
    image = tf.image.random_crop(image, size=INPUT_SHAPE)

    # Vary the brightness of the image
    image = tf.image.random_brightness(image, max_delta=0.2)

    return image, label
```

Exposure to these variations during training can help prevent your model from taking shortcuts by “memorizing” superficial clues in your training data, meaning it may better reflect the deep underlying patterns in your dataset.

The final layer of our model will have 16 neurons with a 10% dropout for overfitting prevention. Here is the Training output:



TRANSFER LEARNING (IMAGES) (ESP32-CAM-FRUITS-VS-VEGGIES)

#1 EON Tuner Primary

Neural Network settings

Training settings

- Number of training cycles: 20
- Learning rate: 0.0005
- Validation set size: 20 %
- Auto-balance dataset: □
- Data augmentation: ☑

Neural network architecture

Input layer (9,216 features)

MobileNetV1 96x96 0.1 (final layer: 16 neurons, 0.1 dropout)

Choose a different model

Output layer (3 classes)

Start training

Training output

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 76.8%

LOSS 0.68

Confusion matrix (validation set)

	APPLE	BANANA	POTATO
APPLE	83.3%	11.1%	5.6%
BANANA	16.7%	79.2%	4.2%
POTATO	21.4%	14.3%	64.3%
F1 SCORE	0.75	0.81	0.72

Feature explorer (full training set)

On-device performance

INFERENCING TIME 50 ms.

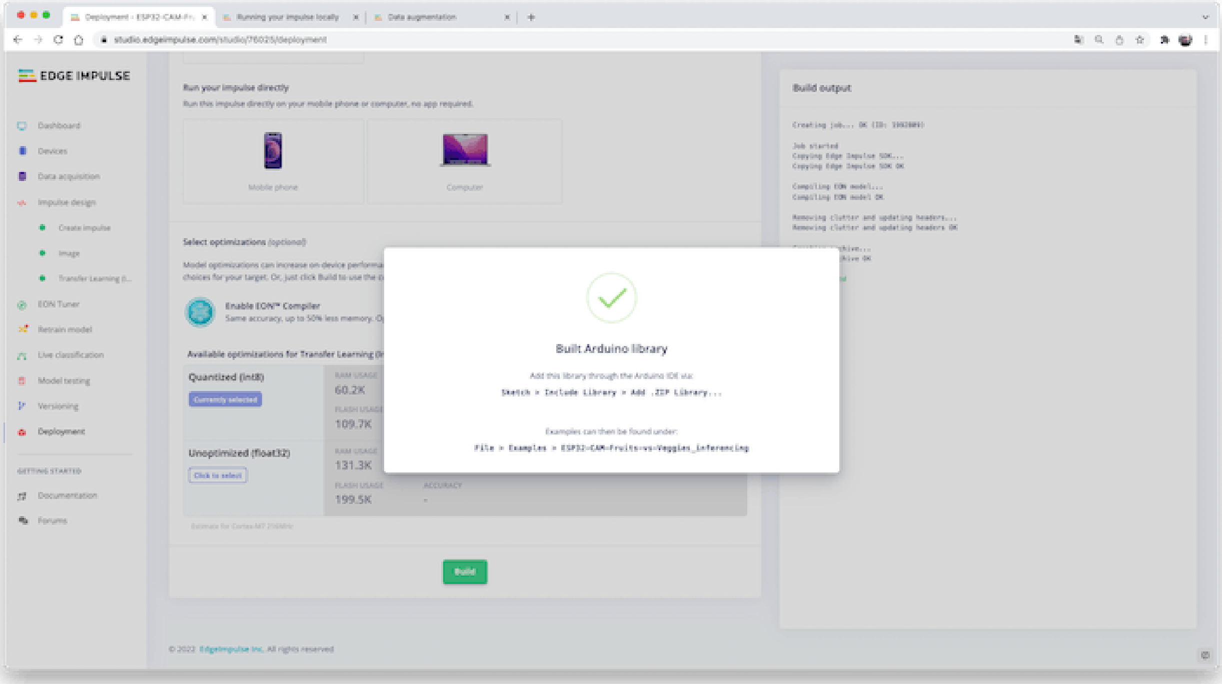
PEAK RAM USAGE 60.2K

FLASH USAGE 109.7K

The result could be better. The model reached around 77% accuracy, but the amount of RAM expected to be used during the inference is relatively tiny (about 60 KBytes), which is very good.

Deployment

The trained model will be deployed as a .zip Arduino library:



Run your impulse directly

Run this impulse directly on your mobile phone or computer, no app required.

Select optimizations (optional)

Model optimizations can increase on-device performa... choices for your target. Or, just click Build to use the ...

Enable EON™ Compiler
Same accuracy, up to 50% less memory. O...

Available optimizations for Transfer Learning (I...

Quantized (int8) — Currently selected — RAM USAGE 60.2K — FLASH USAGE 109.7K

Unoptimized (float32) — Click to select — RAM USAGE 131.3K — FLASH USAGE 199.5K — ACCURACY -

Built Arduino library

Add this library through the Arduino IDE via:

```
Sketch > Include Library > Add .ZIP Library...
```

Examples can then be found under:

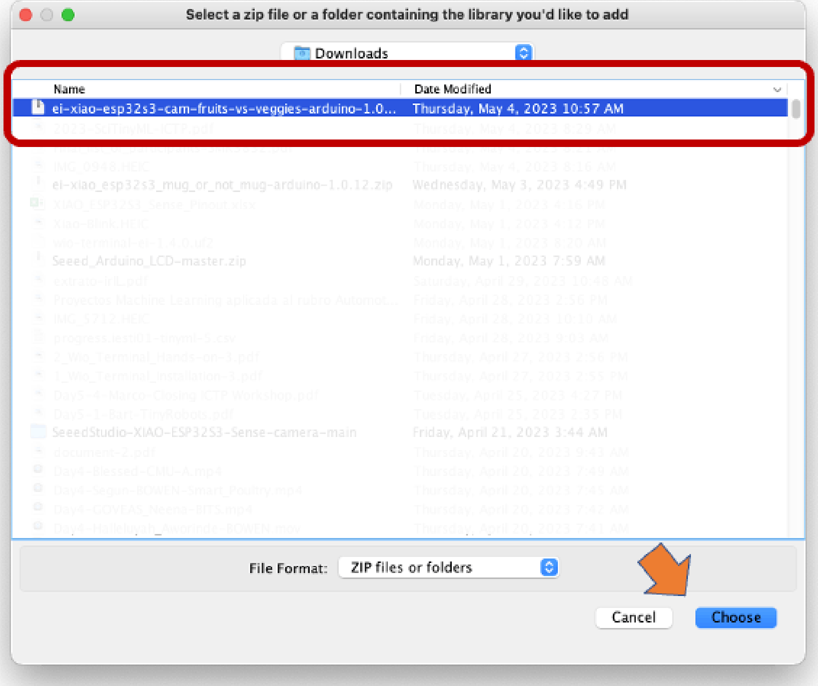
```
File > Examples > ESP32-CAM-Fruits-vs-Veggies_inferencing
```

Build output

```
Creating job... OK (ID: [illegible])

Job started
Copying Edge Impulse SDK...
Copying Edge Impulse SDK OK

Compiling EON model...
Compiling EON model OK

Removing clutter and updating headers...
Removing clutter and updating headers OK

...chive...
...chive OK
```

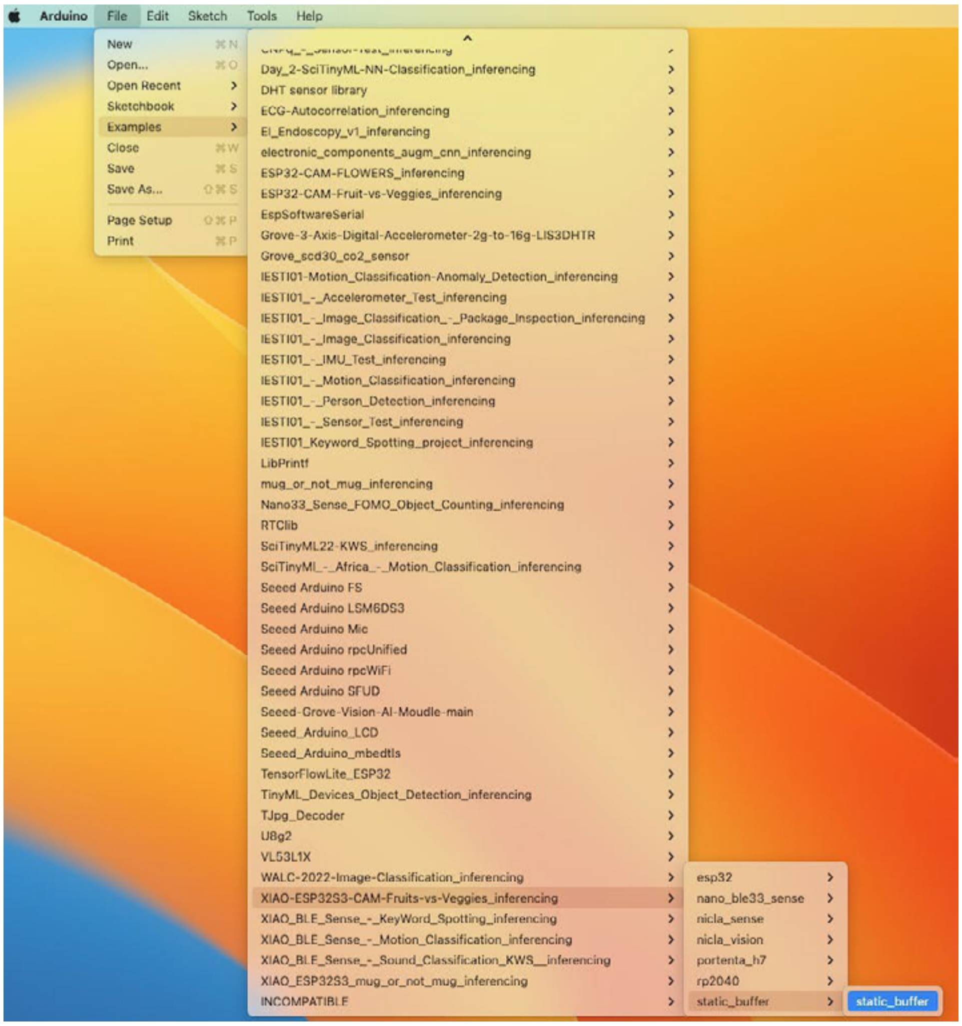
© 2022 EdgeImpulse Inc. All rights reserved

Open your Arduino IDE, and under **Sketch**, go to **Include Library** and **add.ZIP Library**. Please select the file you download from Edge Impulse Studio, and that's it!

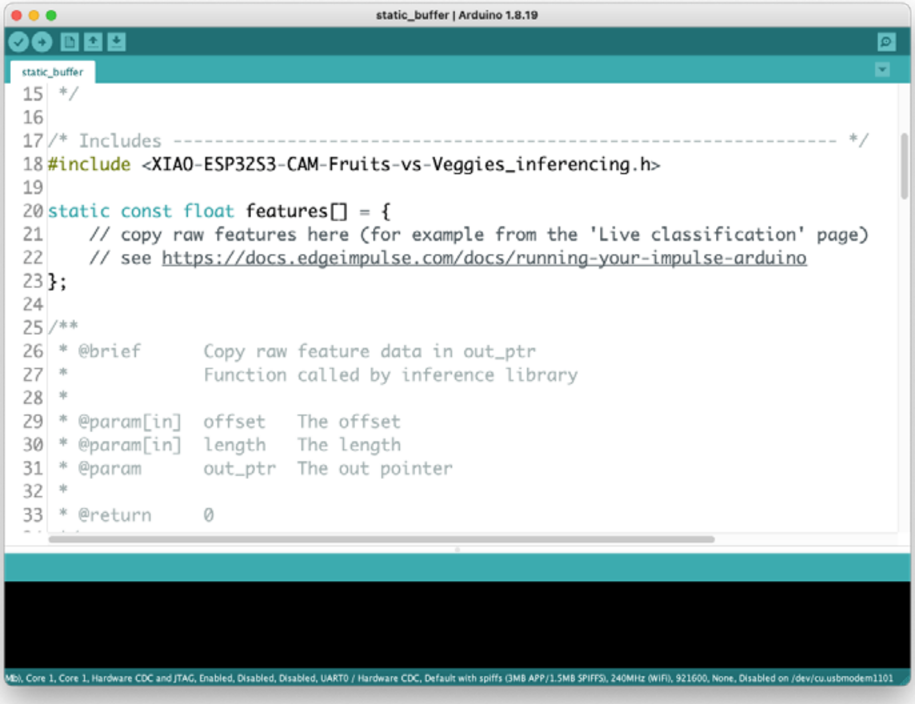

Select a zip file or a folder containing the library you'd like to add
Downloads
File Format:
ZIP files or folders
Cancel
Choose

Name	Date Modified
ei-xiao-esp32s3-cam-fruits-vs-veggies-arduino-1.0...	Thursday, May 4, 2023 10:57 AM
2023-SciTinyML-ICTP.pdf	Thursday, May 4, 2023 8:29 AM
[illegible]	Thursday, May 4, 2023 8:21 AM
IMG_0948.HEIC	Thursday, May 4, 2023 8:16 AM
ei-xiao_esp32s3_mug_or_not_mug-arduino-1.0.12.zip	Wednesday, May 3, 2023 4:49 PM
XIAO_ESP32S3_Sense_Pinout.xlsx	Monday, May 1, 2023 4:16 PM
Xiao-Blink.HEIC	Monday, May 1, 2023 4:12 PM
wio-terminal-ei-1.4.0.uf2	Monday, May 1, 2023 8:20 AM
Seeed_Arduino_LCD-master.zip	Monday, May 1, 2023 7:59 AM
extrato-irIL.pdf	Saturday, April 29, 2023 10:48 AM
Proyectos Machine Learning aplicada al rubro Automot...	Friday, April 28, 2023 2:56 PM
IMG_5712.HEIC	Friday, April 28, 2023 10:10 AM
progress.iesti01-tinyml-5.csv	Friday, April 28, 2023 9:03 AM
2_Wio_Terminal_Hands-on-3.pdf	Thursday, April 27, 2023 2:56 PM
1_Wio_Terminal_installation-3.pdf	Thursday, April 27, 2023 2:55 PM
Day5-4-Marco-Closing ICTP Workshop.pdf	Tuesday, April 25, 2023 4:27 PM
Day5-1-Bart-TinyRobots.pdf	Tuesday, April 25, 2023 2:35 PM
SeeedStudio-XIAO-ESP32S3-Sense-camera-main	Friday, April 21, 2023 3:44 AM
document-2.pdf	Thursday, April 20, 2023 9:43 AM
Day4-Blessed-CMU-A.mp4	Thursday, April 20, 2023 7:49 AM
Day4-Segun-BOWEN-Smart_Poultry.mp4	Thursday, April 20, 2023 7:45 AM
Day4-GOVEAS_Neena-BITS.mp4	Thursday, April 20, 2023 7:42 AM
Day4-Halleluyah_Aworinde-BOWEN.mov	Thursday, April 20, 2023 7:41 AM

Under the **Examples** tab on Arduino IDE, you should find a sketch code under your project name.


Arduino
File
Edit
Sketch
Tools
Help
New ⌘N
Open... ⌘O
Open Recent
Sketchbook
Examples
Close ⌘W
Save ⌘S
Save As... ⇧⌘S
Page Setup ⇧⌘P
Print ⌘P
Day_2-SciTinyML-NN-Classification_inferencing
DHT sensor library
ECG-Autocorrelation_inferencing
EI_Endoscopy_v1_inferencing
electronic_components_augm_cnn_inferencing
ESP32-CAM-FLOWERS_inferencing
ESP32-CAM-Fruit-vs-Veggies_inferencing
EspSoftwareSerial
Grove-3-Axis-Digital-Accelerometer-2g-to-16g-LIS3DHTR
Grove_scd30_co2_sensor
IESTI01-Motion_Classification-Anomaly_Detection_inferencing
IESTI01_-_Accelerometer_Test_inferencing
IESTI01_-_Image_Classification_-_Package_Inspection_inferencing
IESTI01_-_Image_Classification_inferencing
IESTI01_-_IMU_Test_inferencing
IESTI01_-_Motion_Classification_inferencing
IESTI01_-_Person_Detection_inferencing
IESTI01_-_Sensor_Test_inferencing
IESTI01_Keyword_Spotting_project_inferencing
LibPrintf
mug_or_not_mug_inferencing
Nano33_Sense_FOMO_Object_Counting_inferencing
RTClib
SciTinyML22-KWS_inferencing
SciTinyML_-_Africa_-_Motion_Classification_inferencing
Seeed Arduino FS
Seeed Arduino LSM6DS3
Seeed Arduino Mic
Seeed Arduino rpcUnified
Seeed Arduino rpcWiFi
Seeed Arduino SFUD
Seeed-Grove-Vision-AI-Moudle-main
Seeed_Arduino_LCD
Seeed_Arduino_mbedtls
TensorFlowLite_ESP32
TinyML_Devices_Object_Detection_inferencing
TJpg_Decoder
U8g2
VL53L1X
WALC-2022-Image-Classification_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing
XIAO_BLE_Sense_-_KeyWord_Spotting_inferencing
XIAO_BLE_Sense_-_Motion_Classification_inferencing
XIAO_BLE_Sense_-_Sound_Classification_KWS__inferencing
XIAO_ESP32S3_mug_or_not_mug_inferencing
INCOMPATIBLE
esp32
nano_ble33_sense
nicla_sense
nicla_vision
portenta_h7
rp2040
static_buffer
static_buffer

Open the Static Buffer example:



```
*/

/* Includes ---------------------------------------------------------------- */
#include <XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing.h>

static const float features[] = {
    // copy raw features here (for example from the 'Live classification' page)
    // see https://docs.edgeimpulse.com/docs/running-your-impulse-arduino
};

/**
 * @brief      Copy raw feature data in out_ptr
 *             Function called by inference library
 *
 * @param[in]  offset   The offset
 * @param[in]  length   The length
 * @param      out_ptr  The out pointer
 *
 * @return     0
```

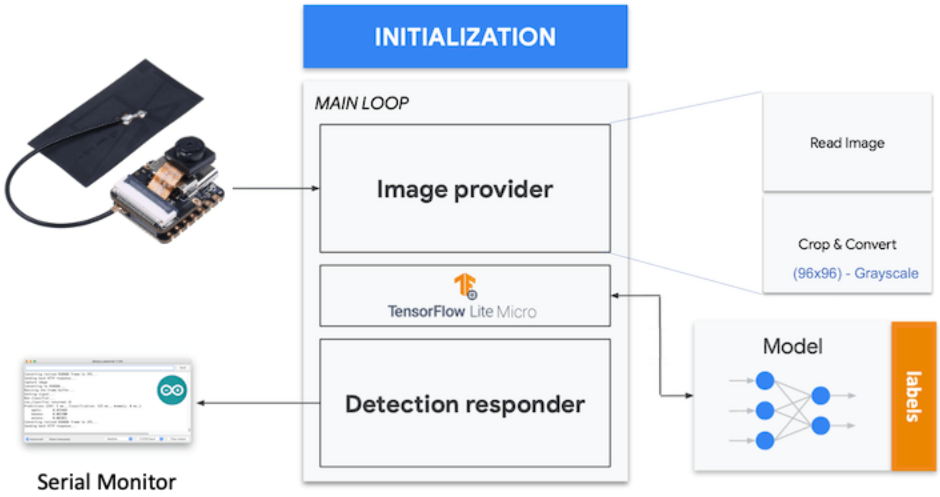
```
Mb), Core 1, Core 1, Hardware CDC and JTAG, Enabled, Disabled, Disabled, UART0 / Hardware CDC, Default with spiffs (3MB APP/1.5MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on /dev/cu.usbmodem1101
```

You can see that the first line of code is exactly the calling of a library with all the necessary stuff for running inference on your device.

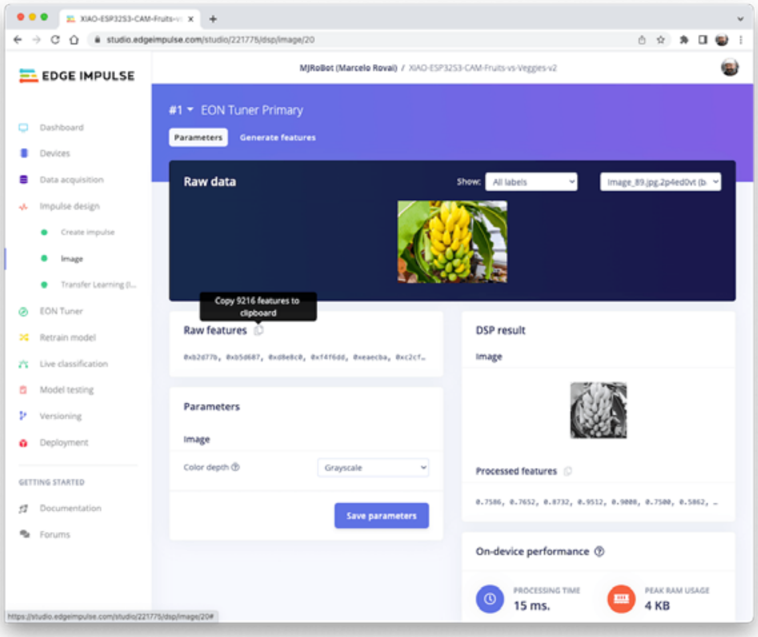
```
#include <XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing.h>
```

Of course, this is a generic code (a "template") that only gets one sample of raw data (stored on the variable: features = {} and runs the classifier, doing the inference. The result is shown on the Serial Monitor.

We should get the sample (image) from the camera and pre-process it (resizing to 96x96, converting to grayscale, and flatting it). This will be the input tensor of our model. The output tensor will be a vector with three values (labels), showing the probabilities of each one of the classes.



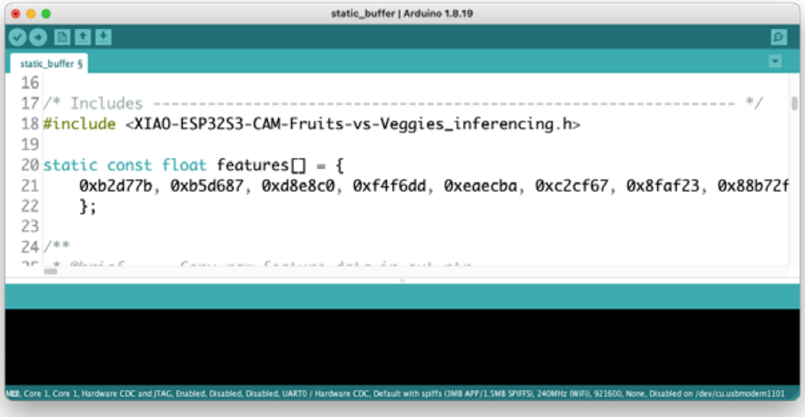
Returning to your project (Tab Image), copy one of the Raw Data Sample:



```
0xb2d77b, 0xb5d687, 0xd8e8c0, 0xf4f6dd, 0xeaecba, 0xc2cf…
```

```
0.7586, 0.7652, 0.8732, 0.9512, 0.9008, 0.7500, 0.5862, …
```

9, 216 features will be copied to the clipboard. This is the input tensor (a flattened image of 96x96x1), in this case, bananas. Past this Input tensor on features[] = {0xb2d77b, 0xb5d687, 0xd8e8c0, 0xeaecba, 0xc2cf67, ...}

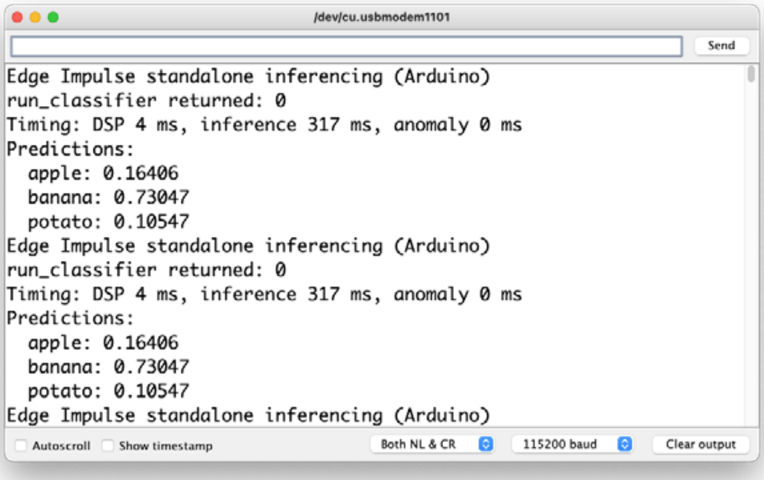


```

/* Includes ---------------------------------------------------------------- */
#include <XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing.h>

static const float features[] = {
    0xb2d77b, 0xb5d687, 0xd8e8c0, 0xf4f6dd, 0xeaecba, 0xc2cf67, 0xe8faf23, 0x88b72f
    };

/**
```

MB), Core 1, Core 1, Hardware CDC and JTAG, Enabled, Disabled, Disabled, UART0 / Hardware CDC, Default with spiffs (3MB APP/1.5MB SPIFFS), 240MHz (WiFi), 921600, None, Disabled on /dev/cu.usbmodem1101

Edge Impulse included the library ESP NN in its SDK, which contains optimized NN (Neural Network) functions for various Espressif chips, including the ESP32S3 (running at Arduino IDE).

When running the inference, you should get the highest score for "banana."

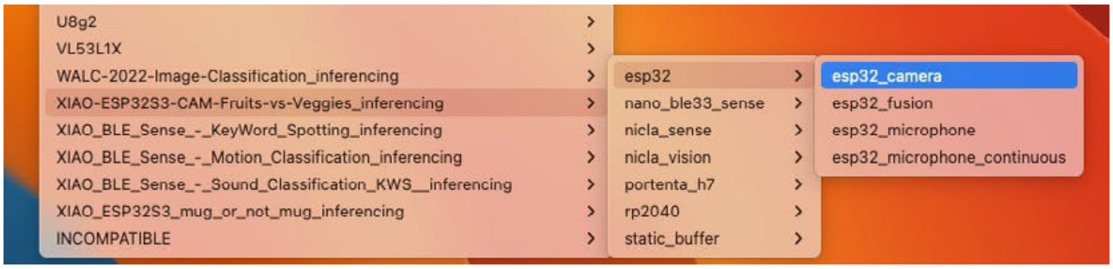


```
Edge Impulse standalone inferencing (Arduino)
run_classifier returned: 0
Timing: DSP 4 ms, inference 317 ms, anomaly 0 ms
Predictions:
  apple: 0.16406
  banana: 0.73047
  potato: 0.10547
Edge Impulse standalone inferencing (Arduino)
run_classifier returned: 0
Timing: DSP 4 ms, inference 317 ms, anomaly 0 ms
Predictions:
  apple: 0.16406
  banana: 0.73047
  potato: 0.10547
Edge Impulse standalone inferencing (Arduino)
```

Great news! Our device handles an inference, discovering that the input image is a banana. Also, note that the inference time was around 317ms, resulting in a maximum of 3 fps if you tried to classify images from a video. It is a better result than the ESP32 CAM (525ms of latency).

Now, we should incorporate the camera and classify images in real time.

Go to the Arduino IDE Examples and download from your project the sketch esp32_camera:

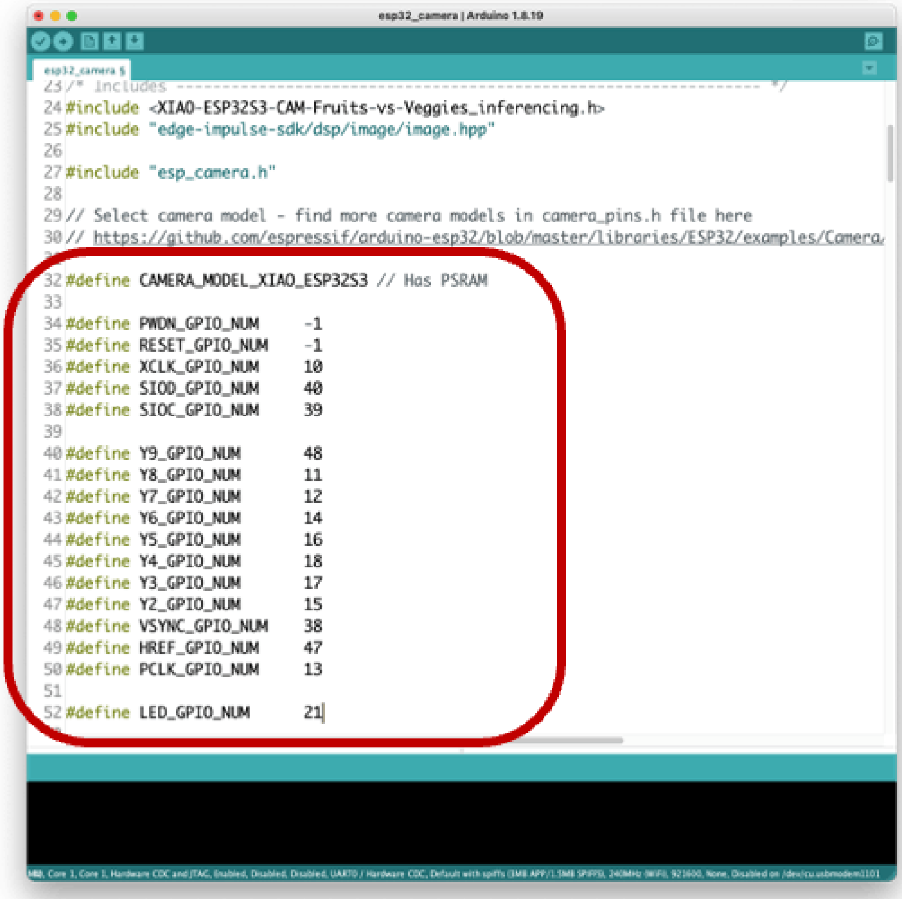

U8g2
VL53L1X
WALC-2022-Image-Classification_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing
XIAO_BLE_Sense_-_KeyWord_Spotting_inferencing
XIAO_BLE_Sense_-_Motion_Classification_inferencing
XIAO_BLE_Sense_-_Sound_Classification_KWS__inferencing
XIAO_ESP32S3_mug_or_not_mug_inferencing
INCOMPATIBLE
esp32
nano_ble33_sense
nicla_sense
nicla_vision
portenta_h7
rp2040
static_buffer
esp32_camera
esp32_fusion
esp32_microphone
esp32_microphone_continuous

You should change lines 32 to 75, which define the camera model and pins, using the data related to our model. Copy and paste the below lines, replacing the lines 32-75:

```
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
#define Y9_GPIO_NUM       48
#define Y8_GPIO_NUM       11
#define Y7_GPIO_NUM       12
```

```
#define Y6_GPIO_NUM       14
#define Y5_GPIO_NUM       16
#define Y4_GPIO_NUM       18
#define Y3_GPIO_NUM       17
#define Y2_GPIO_NUM       15
#define VSYNC_GPIO_NUM    38
#define HREF_GPIO_NUM     47
#define PCLK_GPIO_NUM     13
```

Here you can see the resulting code:



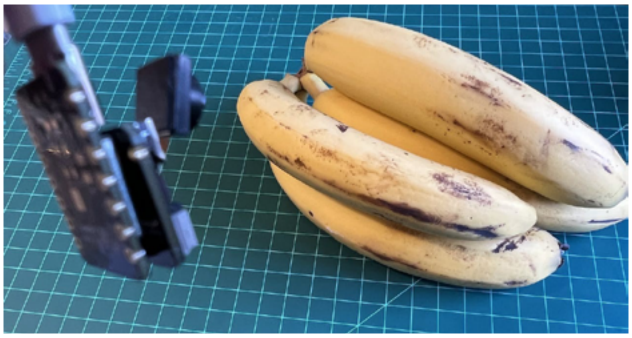
```
23 /* Includes ---------------------------------------------------------------- */
24 #include <XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing.h>
25 #include "edge-impulse-sdk/dsp/image/image.hpp"
26
27 #include "esp_camera.h"
28
29 // Select camera model - find more camera models in camera_pins.h file here
30 // https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera
31
32 #define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
33
34 #define PWDN_GPIO_NUM     -1
35 #define RESET_GPIO_NUM    -1
36 #define XCLK_GPIO_NUM     10
37 #define SIOD_GPIO_NUM     40
38 #define SIOC_GPIO_NUM     39
39
40 #define Y9_GPIO_NUM       48
41 #define Y8_GPIO_NUM       11
42 #define Y7_GPIO_NUM       12
43 #define Y6_GPIO_NUM       14
44 #define Y5_GPIO_NUM       16
45 #define Y4_GPIO_NUM       18
46 #define Y3_GPIO_NUM       17
47 #define Y2_GPIO_NUM       15
48 #define VSYNC_GPIO_NUM    38
49 #define HREF_GPIO_NUM     47
50 #define PCLK_GPIO_NUM     13
51
52 #define LED_GPIO_NUM      21
```

The modified sketch can be downloaded from GitHub: xiao_esp32s3_camera.

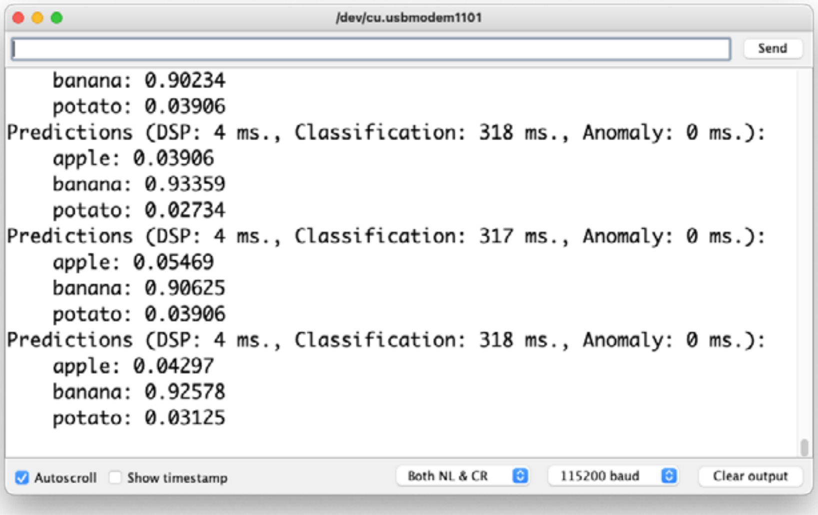
> *Note that you can optionally keep the pins as a .h file as we did in previous sections.*

Upload the code to your XIAO ESP32S3 Sense, and you should be OK to start classifying your fruits and vegetables! You can check the result on Serial Monitor.

4.4.11 Testing the Model (Inference)

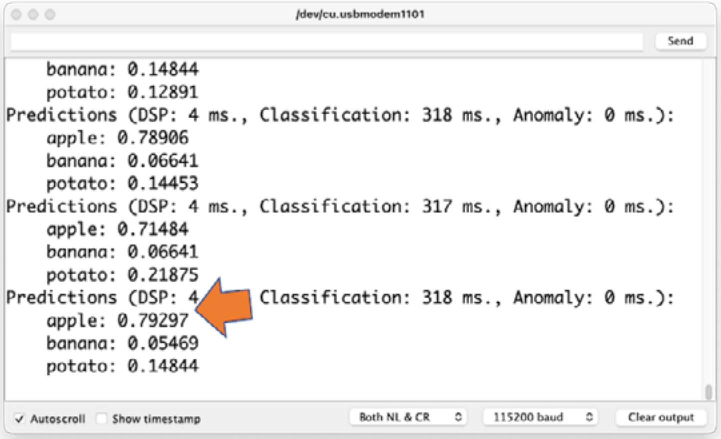


Getting a photo with the camera, the classification result will appear on the Serial Monitor:



```
    banana: 0.90234
    potato: 0.03906
Predictions (DSP: 4 ms., Classification: 318 ms., Anomaly: 0 ms.):
    apple: 0.03906
    banana: 0.93359
    potato: 0.02734
Predictions (DSP: 4 ms., Classification: 317 ms., Anomaly: 0 ms.):
    apple: 0.05469
    banana: 0.90625
    potato: 0.03906
Predictions (DSP: 4 ms., Classification: 318 ms., Anomaly: 0 ms.):
    apple: 0.04297
    banana: 0.92578
    potato: 0.03125
```

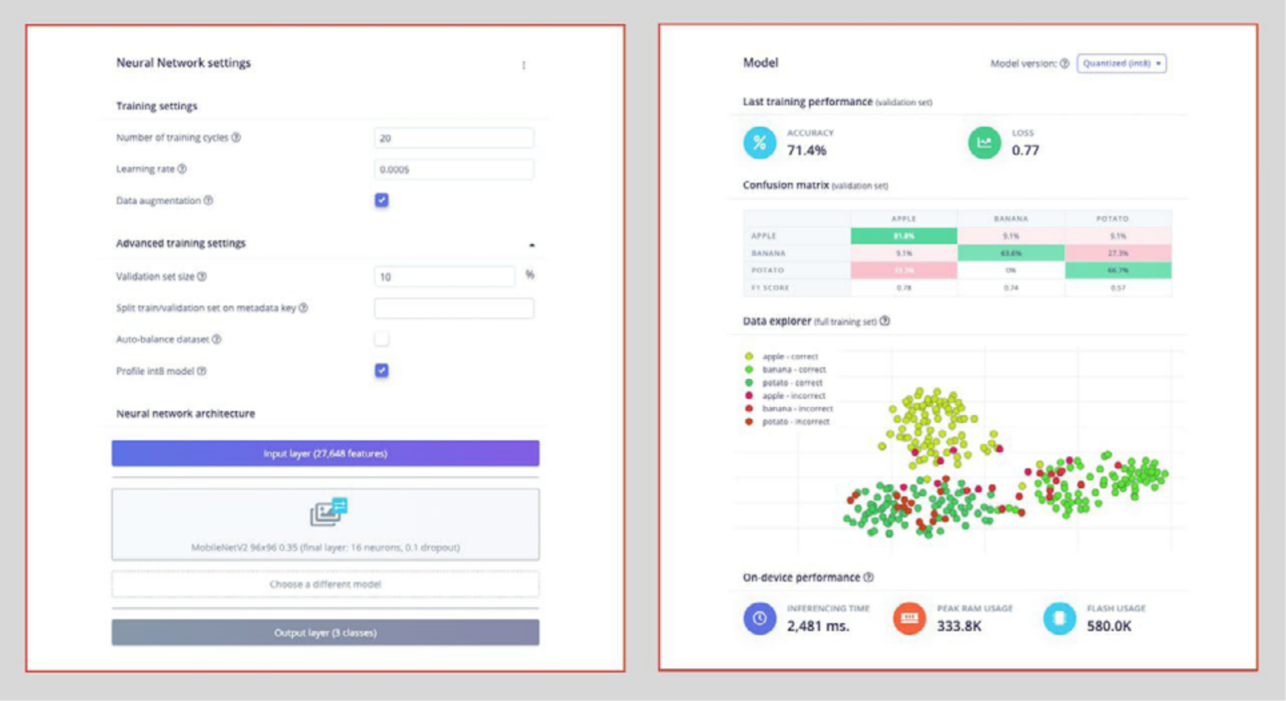
Other tests:



```
    banana: 0.14844
    potato: 0.12891
Predictions (DSP: 4 ms., Classification: 318 ms., Anomaly: 0 ms.):
    apple: 0.78906
    banana: 0.06641
    potato: 0.14453
Predictions (DSP: 4 ms., Classification: 317 ms., Anomaly: 0 ms.):
    apple: 0.71484
    banana: 0.06641
    potato: 0.21875
Predictions (DSP: 4     Classification: 318 ms., Anomaly: 0 ms.):
    apple: 0.79297
    banana: 0.05469
    potato: 0.14844
```

4.4.12 Testing with a Bigger Model

Now, let's go to the other side of the model size. Let's select a MobilinetV2 96x96 0.35, having as input RGB images.



Neural Network settings

Training settings

Number of training cycles ⓘ 20

Learning rate ⓘ 0.0005

Data augmentation ⓘ ☑

Advanced training settings

Validation set size ⓘ 10 %

Split train/validation set on metadata key ⓘ

Auto-balance dataset ⓘ □

Profile int8 model ⓘ ☑

Neural network architecture

Input layer (27,648 features)

MobileNetV2 96x96 0.35 (final layer: 16 neurons, 0.1 dropout)

Choose a different model

Output layer (3 classes)

Model — Model version: ⓘ Quantized (int8)

Last training performance (validation set)

ACCURACY 71.4%

LOSS 0.77

Confusion matrix (validation set)

	APPLE	BANANA	POTATO
APPLE	81.8%	9.1%	9.1%
BANANA	9.1%	63.6%	27.3%
POTATO	33.3%	0%	66.7%
F1 SCORE	0.78	0.74	0.57

Data explorer (full training set) ⓘ

On-device performance ⓘ

INFERENCING TIME 2,481 ms.

PEAK RAM USAGE 333.8K

FLASH USAGE 580.0K

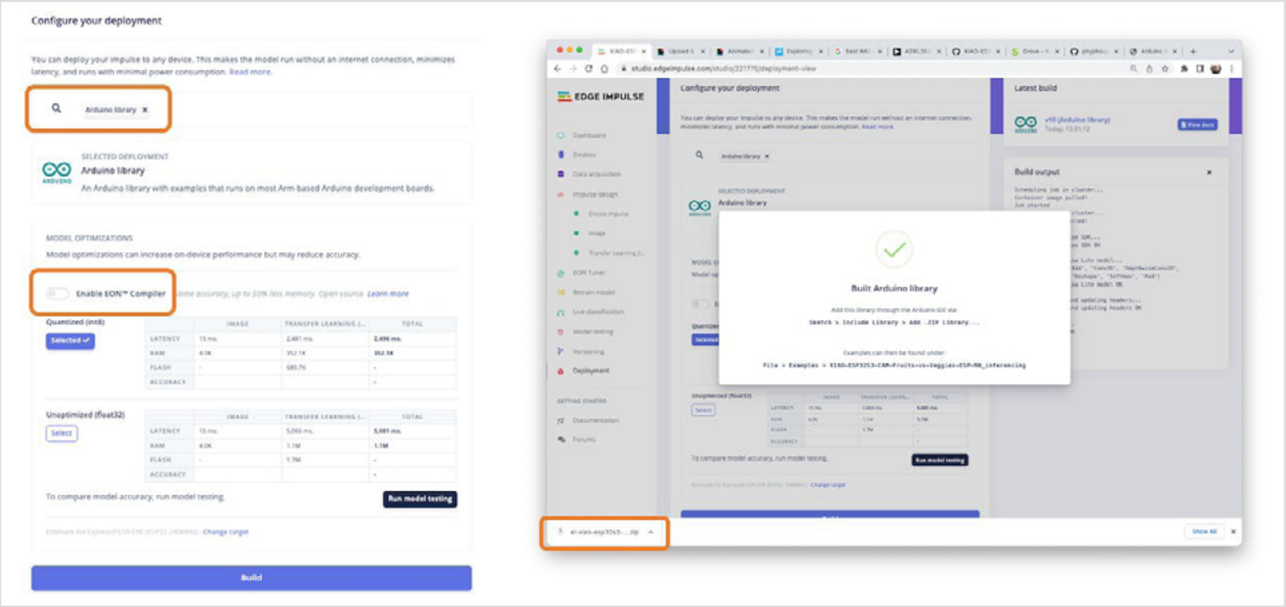
Even with a bigger model, the accuracy could be better, and the amount of memory necessary to run the model increases five times, with latency increasing seven times.

> *Note that the performance here is estimated with a smaller device, the ESP-EYE. The actual inference with the ESP32S3 should be better.*

To improve our model, we will need to train more images.

Even though our model did not improve accuracy, let's test whether the XIAO can handle such a bigger model. We will do a simple inference test with the Static Buffer sketch.

Let's redeploy the model. If the EON Compiler is enabled when you generate the library, the total memory needed for inference should be reduced, but it does not influence accuracy.



Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Arduino library ×

SELECTED DEPLOYMENT

Arduino library

An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

○ Enable EON™ Compiler ...ome accuracy, up to 50% less memory. Open source. Learn more

Quantized (int8) — Selected ✓

	IMAGE	TRANSFER LEARNING (...	TOTAL
LATENCY	15 ms.	2,481 ms.	2,496 ms.
RAM	4.0K	352.1K	352.1K
FLASH	-	580.7K	-
ACCURACY			-

Unoptimized (float32) — Select

	IMAGE	TRANSFER LEARNING (...	TOTAL
LATENCY	15 ms.	5,066 ms.	5,081 ms.
RAM	4.0K	1.1M	1.1M
FLASH	-	1.7M	-
ACCURACY			-

To compare model accuracy, run model testing. Run model testing

Estimate for Espressif ESP-EYE (ESP32 240MHz) · Change target

Build

Built Arduino library

Add this library through the Arduino IDE via:

`Sketch > Include Library > Add .ZIP Library...`

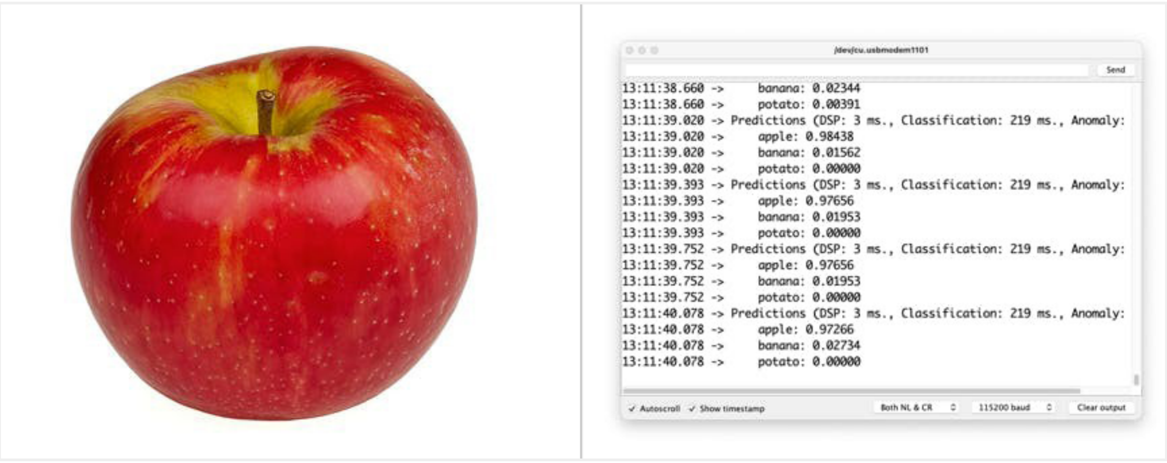
Examples can then be found under:

`File > Examples > XIAO-ESP32S3-CAM-Fruits-vs-Veggies-ESP-NN_inferencing`

Build output

```
Scheduling job in cluster...
Container image pulled!
Job started
```

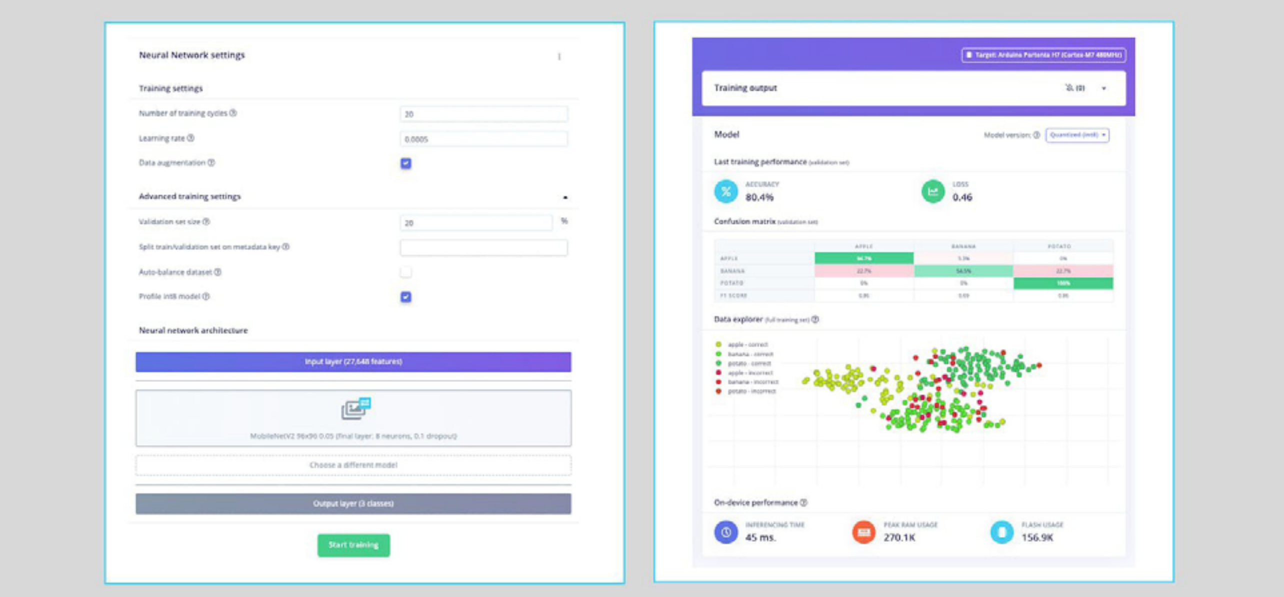
Doing an inference with MobilinetV2 96x96 0.35, having as input RGB images, the latency was 219ms, which is great for such a bigger model.



```
13:11:38.660 ->     banana: 0.02344
13:11:38.660 ->     potato: 0.00391
13:11:39.020 -> Predictions (DSP: 3 ms., Classification: 219 ms., Anomaly:
13:11:39.020 ->     apple: 0.98438
13:11:39.020 ->     banana: 0.01562
13:11:39.020 ->     potato: 0.00000
13:11:39.393 -> Predictions (DSP: 3 ms., Classification: 219 ms., Anomaly:
13:11:39.393 ->     apple: 0.97656
13:11:39.393 ->     banana: 0.01953
13:11:39.393 ->     potato: 0.00000
13:11:39.752 -> Predictions (DSP: 3 ms., Classification: 219 ms., Anomaly:
13:11:39.752 ->     apple: 0.97656
13:11:39.752 ->     banana: 0.01953
13:11:39.752 ->     potato: 0.00000
13:11:40.078 -> Predictions (DSP: 3 ms., Classification: 219 ms., Anomaly:
13:11:40.078 ->     apple: 0.97266
13:11:40.078 ->     banana: 0.02734
13:11:40.078 ->     potato: 0.00000
```

✓ Autoscroll ✓ Show timestamp | Both NL & CR | 115200 baud | Clear output

For the test, I trained the model again, using the smallest version of MobileNet V2, with an alpha of 0.05. Interesting that the result in accuracy was higher.



Neural Network settings

Training settings

- Number of training cycles: 20
- Learning rate: 0.0005
- Data augmentation: ☑

Advanced training settings

- Validation set size: 20 %
- Split train/validation set on metadata key:
- Auto-balance dataset: □
- Profile int8 model: ☑

Neural network architecture

- Input layer (27,648 features)
- MobileNetV2 96x96 0.05 (final layer: 8 neurons, 0.1 dropout)
- Choose a different model
- Output layer (3 classes)

Target: Arduino Portenta H7 (Cortex-M7 480MHz)

Training output

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 80.4%

LOSS 0.46

Confusion matrix (validation set)

	APPLE	BANANA	POTATO
APPLE	94.7%	5.3%	0%
BANANA	22.7%	54.5%	22.7%
POTATO	0%	0%	100%
F1 SCORE	0.86	0.69	0.86

Data explorer (full training set)

On-device performance

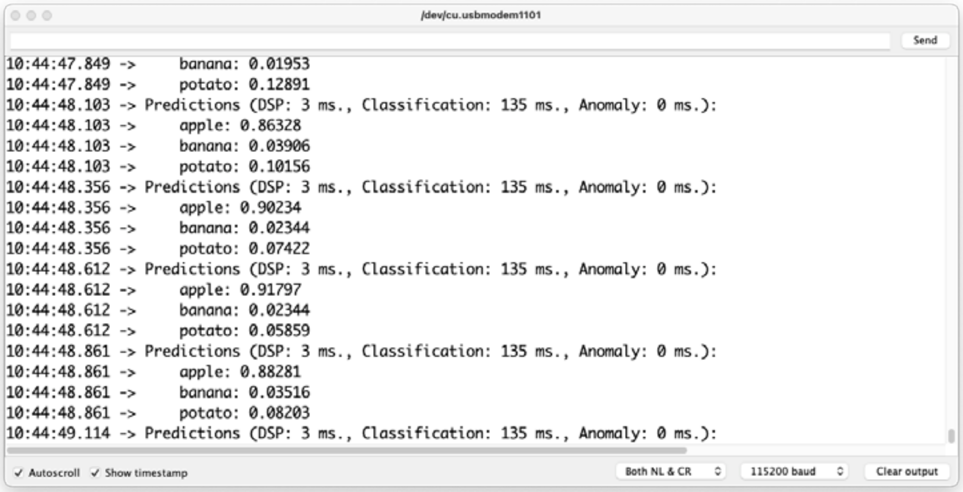
INFERENCING TIME 45 ms.

PEAK RAM USAGE 270.1K

FLASH USAGE 156.9K

Note that the estimated latency for an Arduino Portenta (ou Nicla), running with a clock of 480MHz is 45ms.

Deploying the model, I got an inference of only 135ms, remembering that the XIAO runs with half of the clock used by the Portenta/Nicla (240MHz):



```
10:44:47.849 ->     banana: 0.01953
10:44:47.849 ->     potato: 0.12891
10:44:48.103 -> Predictions (DSP: 3 ms., Classification: 135 ms., Anomaly: 0 ms.):
10:44:48.103 ->     apple: 0.86328
10:44:48.103 ->     banana: 0.03906
10:44:48.103 ->     potato: 0.10156
10:44:48.356 -> Predictions (DSP: 3 ms., Classification: 135 ms., Anomaly: 0 ms.):
10:44:48.356 ->     apple: 0.90234
10:44:48.356 ->     banana: 0.02344
10:44:48.356 ->     potato: 0.07422
10:44:48.612 -> Predictions (DSP: 3 ms., Classification: 135 ms., Anomaly: 0 ms.):
10:44:48.612 ->     apple: 0.91797
10:44:48.612 ->     banana: 0.02344
10:44:48.612 ->     potato: 0.05859
10:44:48.861 -> Predictions (DSP: 3 ms., Classification: 135 ms., Anomaly: 0 ms.):
10:44:48.861 ->     apple: 0.88281
10:44:48.861 ->     banana: 0.03516
10:44:48.861 ->     potato: 0.08203
10:44:49.114 -> Predictions (DSP: 3 ms., Classification: 135 ms., Anomaly: 0 ms.):
```

✓ Autoscroll ✓ Show timestamp | Both NL & CR | 115200 baud | Clear output

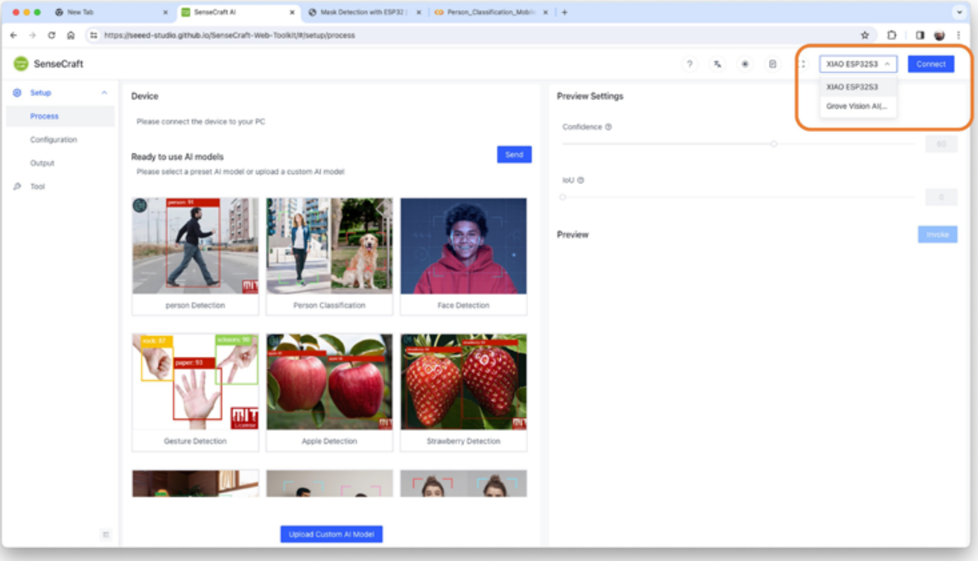
4.4.13 Running inference on the SenseCraft-Web-Toolkit

One significant limitation of viewing inference on Arduino IDE is that we can not see what the camera focuses on. A good alternative is the **SenseCraft-Web-Toolkit**, a visual model deployment tool provided by SSCMA(Seeed SenseCraft Model Assistant). This tool allows you to deploy models to various platforms easily through simple operations. The tool offers a user-friendly interface and does not require any coding.

Follow the following steps to start the SenseCraft-Web-Toolkit:

1. Open the SenseCraft-Web-Toolkit website.
2. Connect the XIAO to your computer:

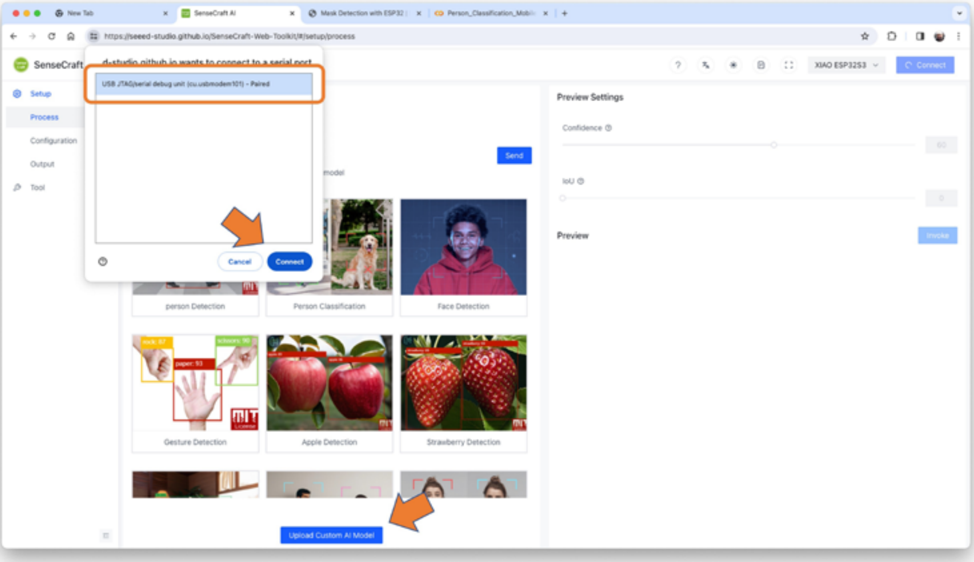
- Having the XIAO connected, select it as below:



Please connect the device to your PC

Please select a preset AI model or upload a custom AI model

- Select the device/Port and press `[Connect]`:



d-studio.github.io wants to connect to a serial port

USB JTAG/serial debug unit (cu.usbmodem101) - Paired

You can try several Computer Vision models previously uploaded by Seeed Studio. Try them and have fun!

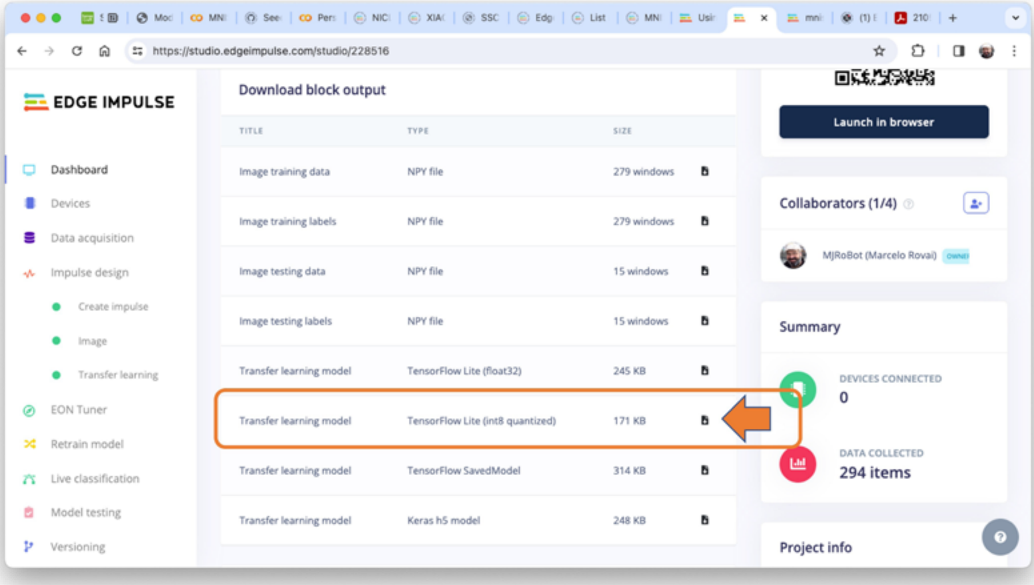
In our case, we will use the blue button at the bottom of the page: `[Upload Custom AI Model]`.

But first, we must download from Edge Impulse Studio our **quantized .tflite** model.

3. Go to your project at Edge Impulse Studio, or clone this one:

- XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v1-ESP-NN

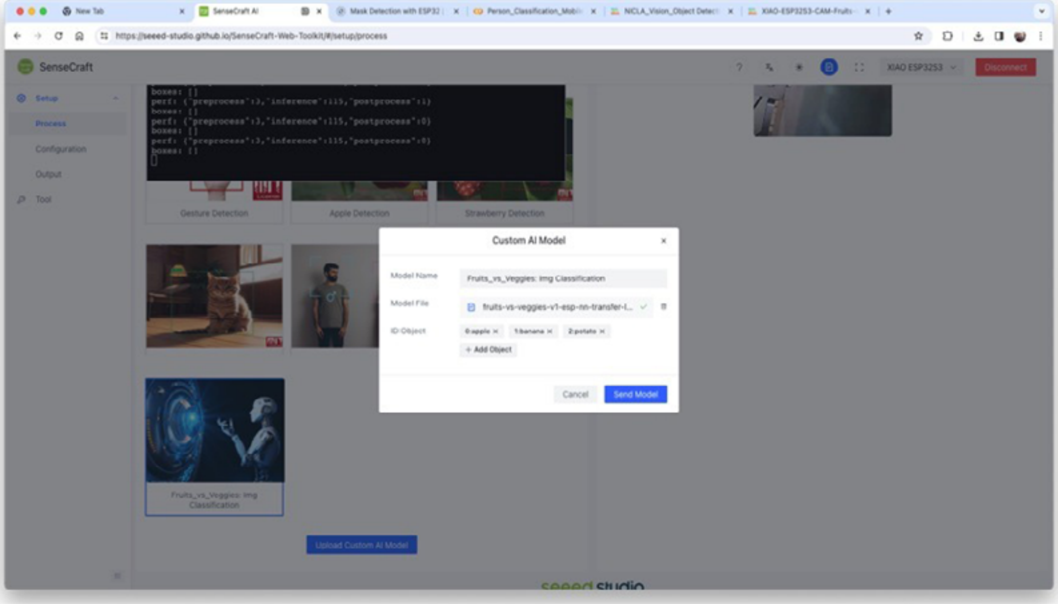
4. On the `Dashboard`, download the model ("block output"): `Transfer learning model - TensorFlow Lite (int8 quantized).`



Download block output

TITLE	TYPE	SIZE
Image training data	NPY file	279 windows
Image training labels	NPY file	279 windows
Image testing data	NPY file	15 windows
Image testing labels	NPY file	15 windows
Transfer learning model	TensorFlow Lite (float32)	245 KB
Transfer learning model	TensorFlow Lite (int8 quantized)	171 KB
Transfer learning model	TensorFlow SavedModel	314 KB
Transfer learning model	Keras h5 model	248 KB

5. On SenseCraft-Web-Toolkit, use the blue button at the bottom of the page: `[Upload Custom AI Model]`. A window will pop up. Enter the Model file that you downloaded to your computer from Edge Impulse Studio, choose a Model Name, and enter with labels (ID: Object):



```
boxes: []
perf: {"preprocess":3,"inference":115,"postprocess":1}
boxes: []
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: []
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: []
```

Custom AI Model

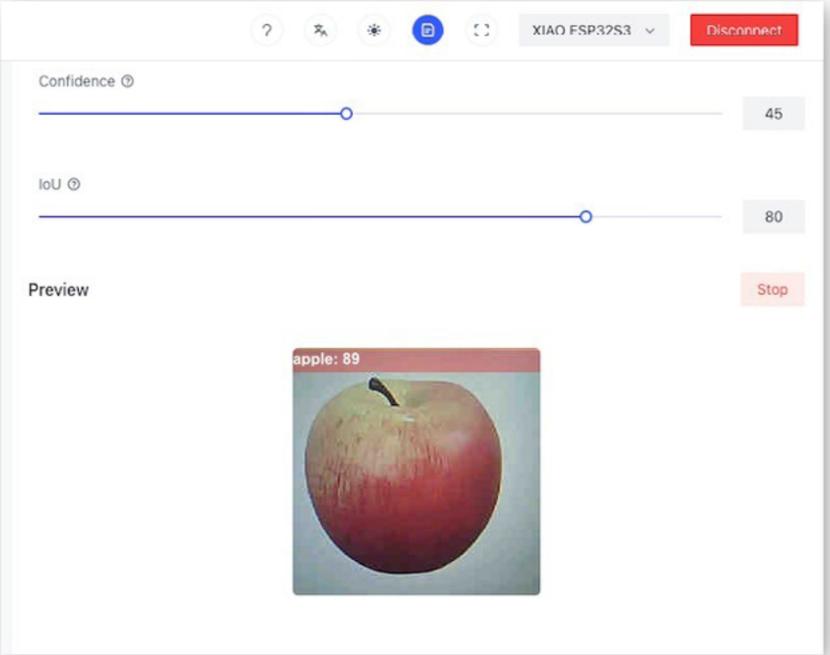
Model Name: Fruits_vs_Veggies: Img Classification

Model File: fruits-vs-veggies-v1-esp-nn-transfer-l...

ID:Object: 0:apple 1:banana 2:potato

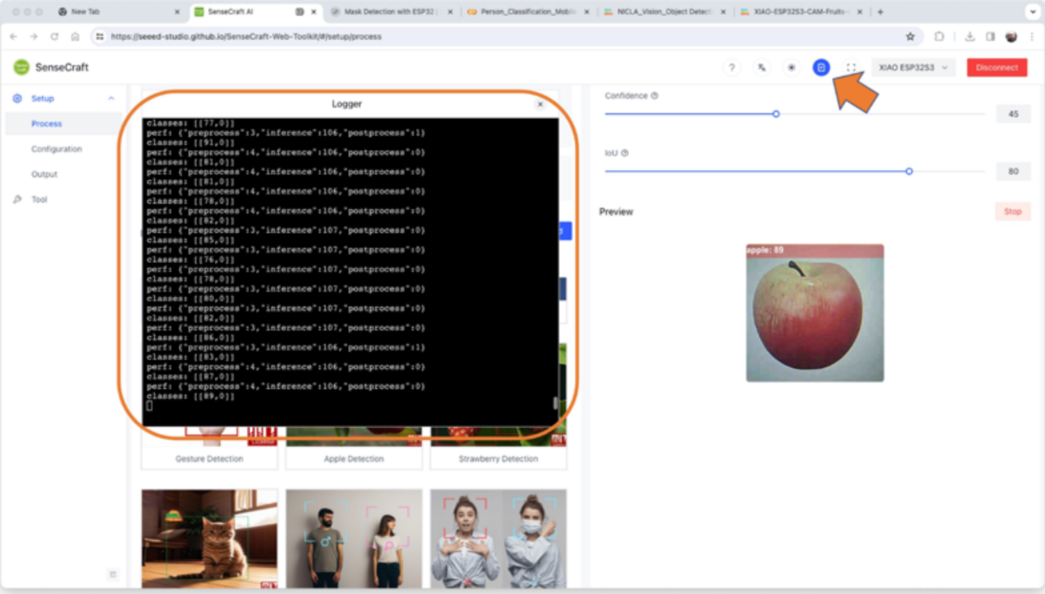
> *Note that you should use the labels trained on EI Studio, entering them in alphabetic order (in our case: apple, banana, potato).*

After a few seconds (or minutes), the model will be uploaded to your device, and the camera image will appear in real-time on the Preview Sector:



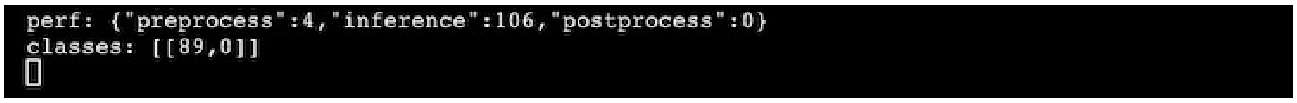
The Classification result will be at the top of the image. You can also select the Confidence of your inference cursor Confidence.

Clicking on the top button (Device Log), you can open a Serial Monitor to follow the inference, the same that we have done with the Arduino IDE:



```
classes: [[77,0]]
perf: {"preprocess":3,"inference":106,"postprocess":1}
classes: [[91,0]]
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[81,0]]
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[81,0]]
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[78,0]]
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[82,0]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[85,0]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[76,0]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[78,0]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[80,0]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[82,0]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[86,0]]
perf: {"preprocess":3,"inference":106,"postprocess":1}
classes: [[83,0]]
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[87,0]]
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[89,0]]
```

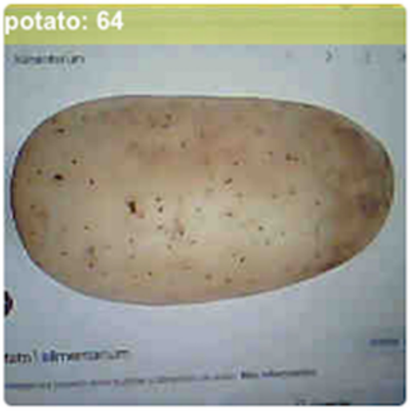
On Device Log, you will get information as:

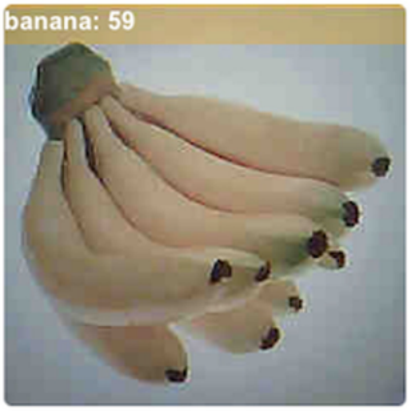


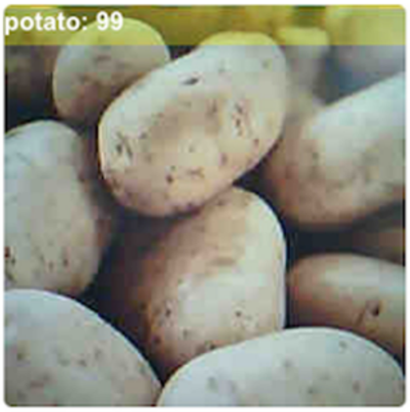
```
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[89,0]]
```

- Preprocess time (image capture and Crop): 4ms;
- Inference time (model latency): 106ms,
- Postprocess time (display of the image and inclusion of data): 0ms.
- Output tensor (classes), for example: [[89,0]]; where 0 is Apple (and 1is banana and 2 is potato)

Here are other screenshots:





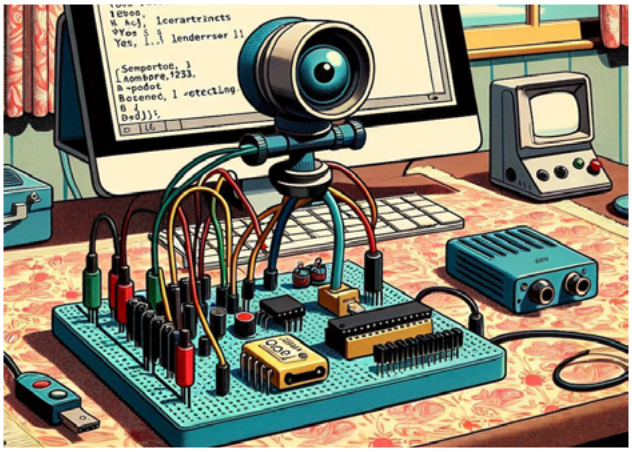


4.4.14 Conclusion

The XIAO ESP32S3 Sense is very flexible, inexpensive, and easy to program. The project proves the potential of TinyML. Memory is not an issue; the device can handle many post-processing tasks, including communication.

You will find the last version of the codes on the GitHub repository: XIAO-ESP32S3-Sense.

4.5
Object Detection

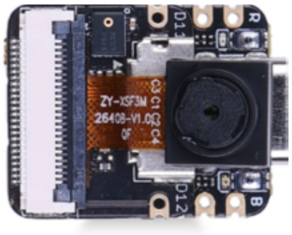


This section will cover other critical computer vision applications, such as object detection using the XIAO ESP32S3 Sense, Edge Impulse Studio, and Arduino IDE.

4.5.1 Things used in this project

Hardware components

- Seeed Studio Seeed XIAO ESP32S3 Sense × 1



Software apps and online services

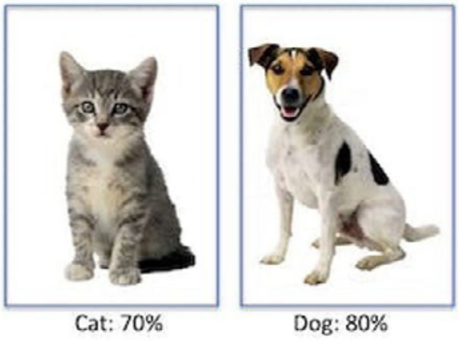
- Arduino IDE
- Edge Impulse Studio

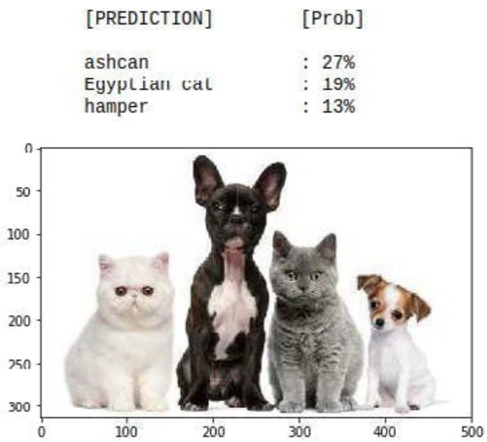
4.5.2 Introduction

In the last section regarding Computer Vision (CV) and the XIAO ESP32S3, Image Classification, we learned how to set up and classify images with this remarkable development board. Continuing our CV journey, we will explore **Object Detection** on microcontrollers.

Object Detection versus Image Classification

The main task with Image Classification models is to identify the most probable object category present on an image, for example, to classify between a cat or a dog, dominant "objects" in an image:





```
[PREDICTION]        [Prob]

ashcan              : 27%
Egyptian cat        : 19%
hamper              : 13%
```

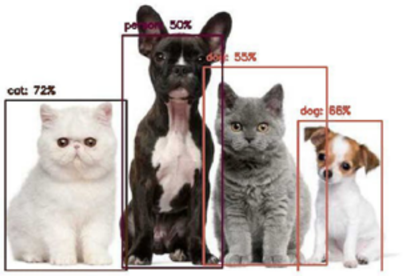
But what happens if there is no dominant category in the image?

An image classification model identifies the above image utterly wrong as an "ashcan," possibly due to the color tonalities.

> *The model used in the previous example is the MobileNet, trained with a large dataset, the ImageNet, running on a Raspberry Pi.*

To solve this issue, we need another type of model, where not only **multiple categories** (or labels) can be found but also **where** the objects are located on a given image.

As we can imagine, such models are much more complicated and bigger, for example, the **MobileNetV2 SSD FPN-Lite 320x320, trained with the COCO dataset**. This pre-trained object detection model is designed to locate up to 10 objects within an image, outputting a bounding box for each object detected. The below image is the result of such a model running on a Raspberry Pi:



Those models used for object detection (such as the MobileNet SSD or YOLO) usually have several MB in size, which is OK for use with Raspberry Pi but unsuitable for use with embedded devices, where the RAM usually is lower than 1M Bytes or at least a few MB as in the case of the XIAO ESP32S3.

An Innovative Solution for Object Detection: FOMO

Edge Impulse launched in 2022, **FOMO** (Faster Objects, More Objects), a novel solution to perform object detection on embedded devices, such as the Nicla Vision and Portenta (Cortex M7), on Cortex M4F CPUs (Arduino Nano33 and OpenMV M4 series) as well the Espressif ESP32 devices (ESP-CAM, ESP-EYE and XIAO ESP32S3 Sense).

In this Hands-On project, we will explore Object Detection using FOMO.

> *To understand more about FOMO, you can go into the official FOMO announcement by Edge Impulse, where Louis Moreau and Mat Kelcey explain in detail how it works.*

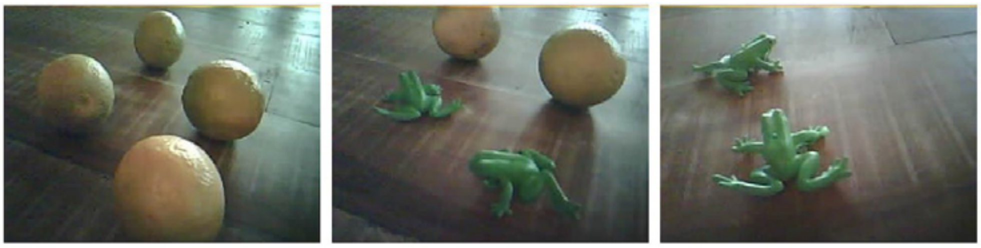
The Object Detection Project Goal

All Machine Learning projects need to start with a detailed goal. Let's assume we are in an industrial or rural facility and must sort and count **oranges (fruits)** and particular **frogs (bugs)**.

In other words, we should perform a multi-label classification, where each image can have three classes:

- Background (No objects)
- Fruit
- Bug

Here are some not labeled image samples that we should use to detect the objects (fruits and bugs):



We are interested in which object is in the image, its location (centroid), and how many we can find on it. The object's size is not detected with FOMO, as with MobileNet SSD or YOLO, where the Bounding Box is one of the model outputs.

We will develop the project using the XIAO ESP32S3 for image capture and model inference. The ML project will be developed using the Edge Impulse Studio. But before starting the object detection project in the Studio, let's create a raw dataset (not labeled) with images that contain the objects to be detected.

Data Collection

You can use the XIAO, your phone, or other devices for the image capture. Here, we will use the XIAO with a code in the ESP32 library.

Collecting Dataset with the XIAO ESP32S3

Open the Arduino IDE and select the XIAO_ESP32S3 board (and the port where it is connected). On `File > Examples > ESP32 > Camera, select CameraWebServer`.

On the BOARDS MANAGER panel, confirm that you have installed the latest "stable" package.

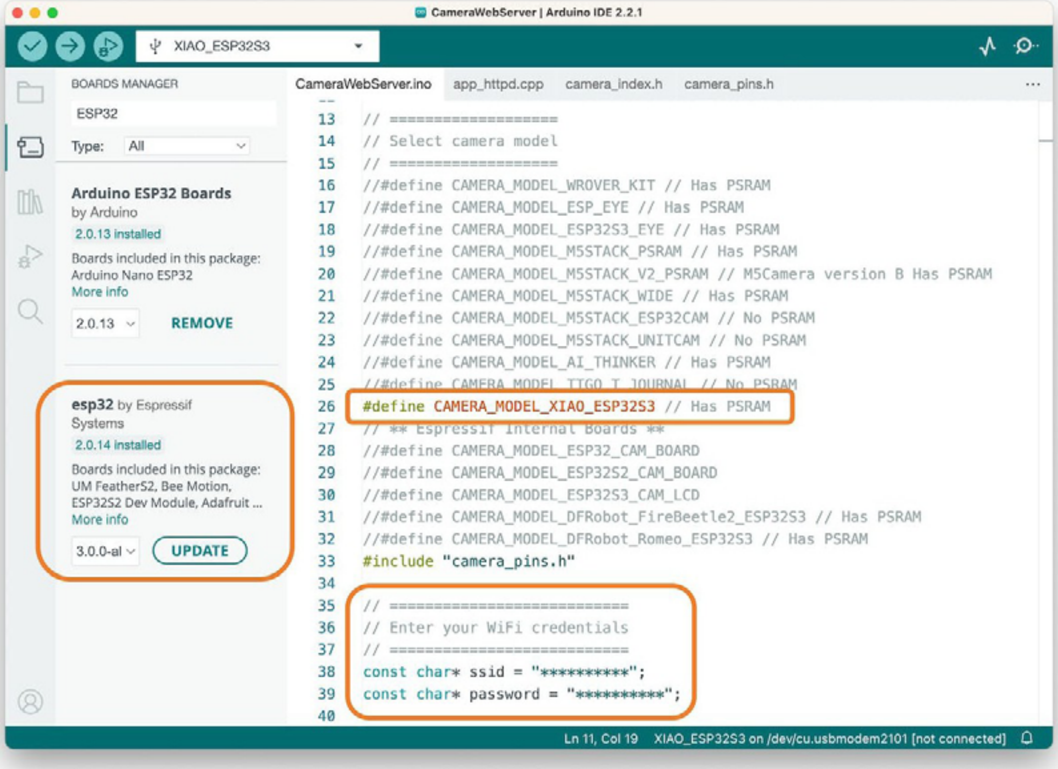
- Attention -

> *Alpha versions (for example, 3.x-alpha) do not work correctly with the XIAO and Edge Impulse. Use the last stable version (for example, 2.0.11) instead.*

You also should comment on all cameras' models, except the XIAO model pins:

```
#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
```

And on `Tools`, enable the PSRAM. Enter your wifi credentials and upload the code to the device:

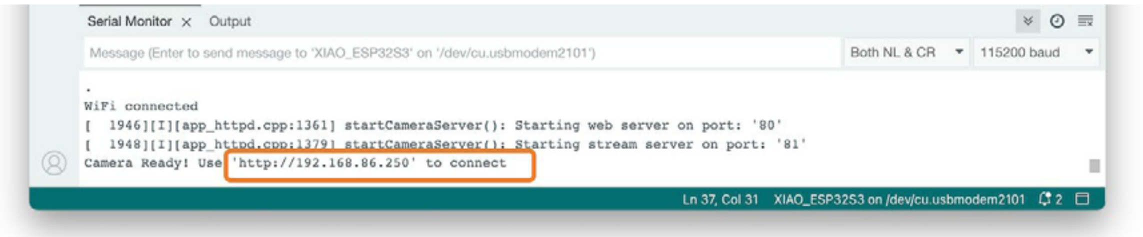


Arduino ESP32 Boards by Arduino
2.0.13 installed
Boards included in this package:
Arduino Nano ESP32
More info
2.0.13

esp32 by Espressif Systems
2.0.14 installed
Boards included in this package:
UM FeatherS2, Bee Motion, ESP32S2 Dev Module, Adafruit ...
More info
3.0.0-al

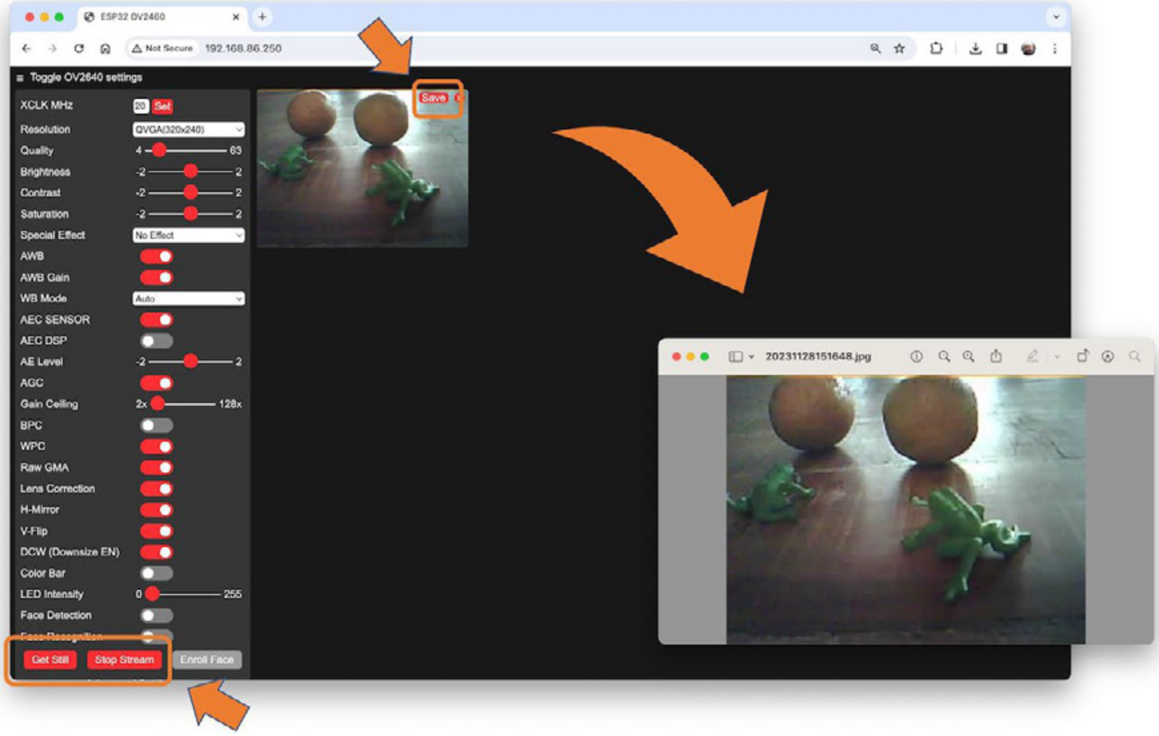
```
// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
//#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD
//#define CAMERA_MODEL_DFRobot_FireBeetle2_ESP32S3 // Has PSRAM
//#define CAMERA_MODEL_DFRobot_Romeo_ESP32S3 // Has PSRAM
#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid = "**********";
const char* password = "**********";

```

If the code is executed correctly, you should see the address on the Serial Monitor:


Serial Monitor
Output
Message (Enter to send message to 'XIAO_ESP32S3' on '/dev/cu.usbmodem2101')
Both NL & CR
115200 baud
Ln 37, Col 31
XIAO_ESP32S3 on /dev/cu.usbmodem2101
2

```
.
WiFi connected
[  1946][I][app_httpd.cpp:1361] startCameraServer(): Starting web server on port: '80'
[  1948][I][app_httpd.cpp:1379] startCameraServer(): Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.86.250' to connect
```

Copy the address on your browser and wait for the page to be uploaded. Select the camera resolution (for example, QVGA) and select `[START STREAM]`. Wait for a few seconds/minutes, depending on your connection. You can save an image on your computer download area using the [Save] button.



XCLK MHz 20 Set
Resolution QVGA(320x240)
Quality 4 – 63
Brightness -2 – 2
Contrast -2 – 2
Saturation -2 – 2
Special Effect No Effect
AWB
AWB Gain
WB Mode Auto
AEC SENSOR
AEC DSP
AE Level -2 – 2
AGC
Gain Ceiling 2x – 128x
BPC
WPC
Raw GMA
Lens Correction
H-Mirror
V-Flip
DCW (Downsize EN)
Color Bar
LED Intensity 0 – 255
Face Detection

Edge impulse suggests that the objects should be of similar size and not overlapping for better performance. This is OK in an industrial facility, where the camera should be fixed, keeping the same distance from the objects to be detected. Despite that, we will also try using mixed sizes and positions to see the result.

> *We do not need to create separate folders for our images because each contains multiple labels.*

We suggest around 50 images mixing the objects and varying the number of each appearing on the scene. Try to capture different angles, backgrounds, and light conditions.

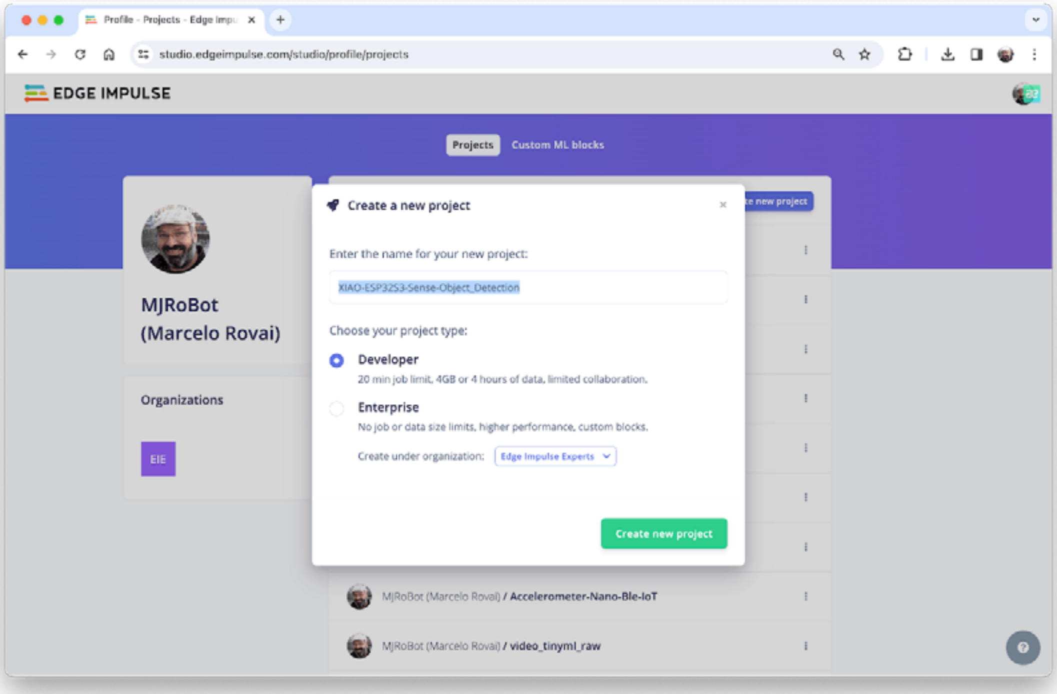
> *The stored images use a QVGA frame size 320x240 and RGB565 (color pixel format).*

After capturing your dataset, [Stop Stream] and move your images to a folder.

Edge Impulse Studio

Setup the project

Go to Edge Impulse Studio, enter your credentials at **Login** (or create an account), and start a new project.


Profile - Projects - Edge Impu
studio.edgeimpulse.com/studio/profile/projects
EDGE IMPULSE
Projects
Custom ML blocks
MJRoBot (Marcelo Rovai)
Organizations
EIE
MJRoBot (Marcelo Rovai) / Accelerometer-Nano-Ble-IoT
MJRoBot (Marcelo Rovai) / video_tinyml_raw

Create a new project

Enter the name for your new project: XIAO-ESP32S3-Sense-Object_Detection

Choose your project type:

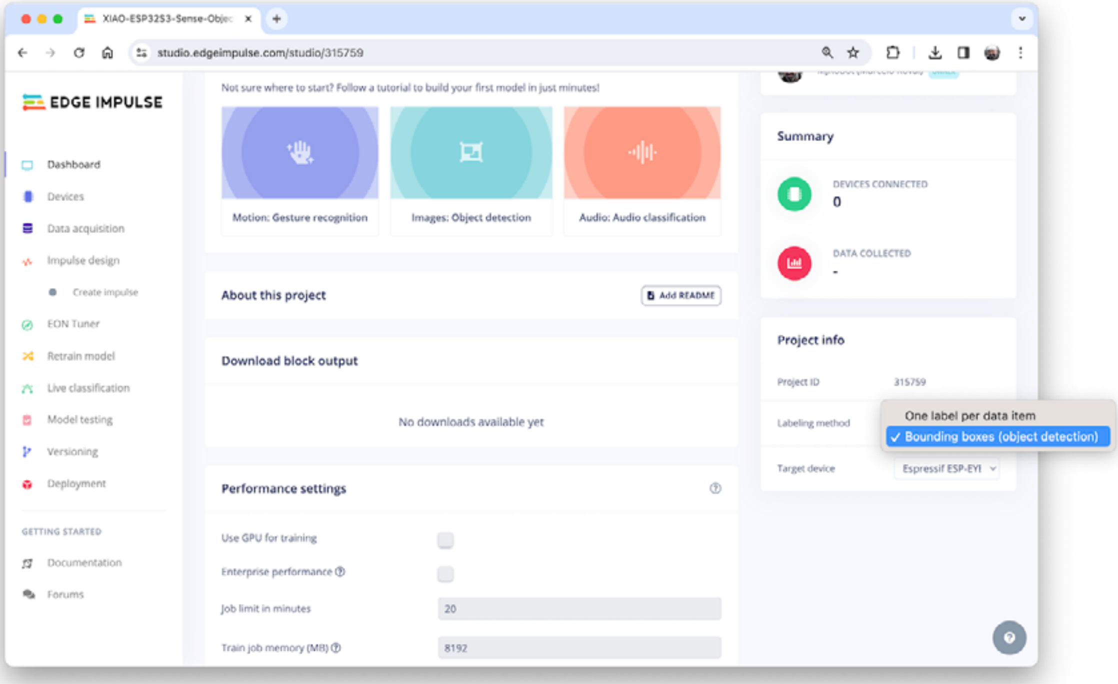
● Developer
20 min job limit, 4GB or 4 hours of data, limited collaboration.

○ Enterprise
No job or data size limits, higher performance, custom blocks.
Create under organization: Edge Impulse Experts

Create new project

Here, you can clone the project developed for this hands-on: XIAO-ESP32S3-Sense-Object_Detection

On your Project Dashboard, go down and on **Project info** and select **Bounding boxes (object detection)** and **Espressif ESP-EYE** (most similar to our board) as your Target Device:



DEVICES CONNECTED 0

DATA COLLECTED -

Project info

Project ID 315759

Labeling method: One label per data item / ✓ Bounding boxes (object detection)

Target device: Espressif ESP-EYE

Use GPU for training □

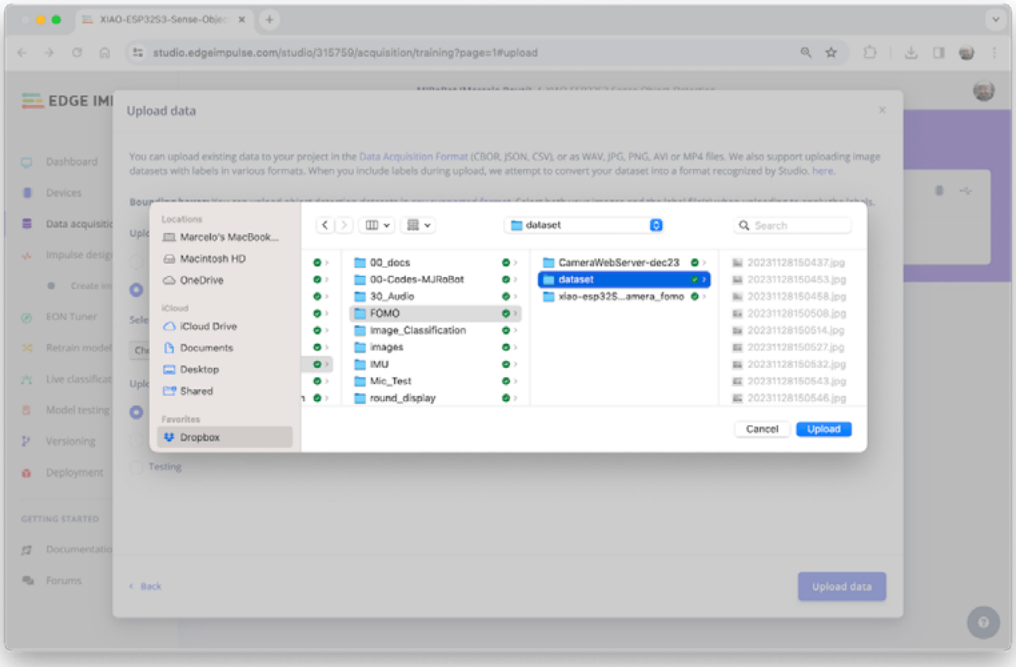
Enterprise performance □

Job limit in minutes 20

Train job memory (MB) 8192

Uploading the unlabeled data

On Studio, go to the `Data acquisition tab`, and on the `UPLOAD DATA` section, upload files captured as a folder from your computer.

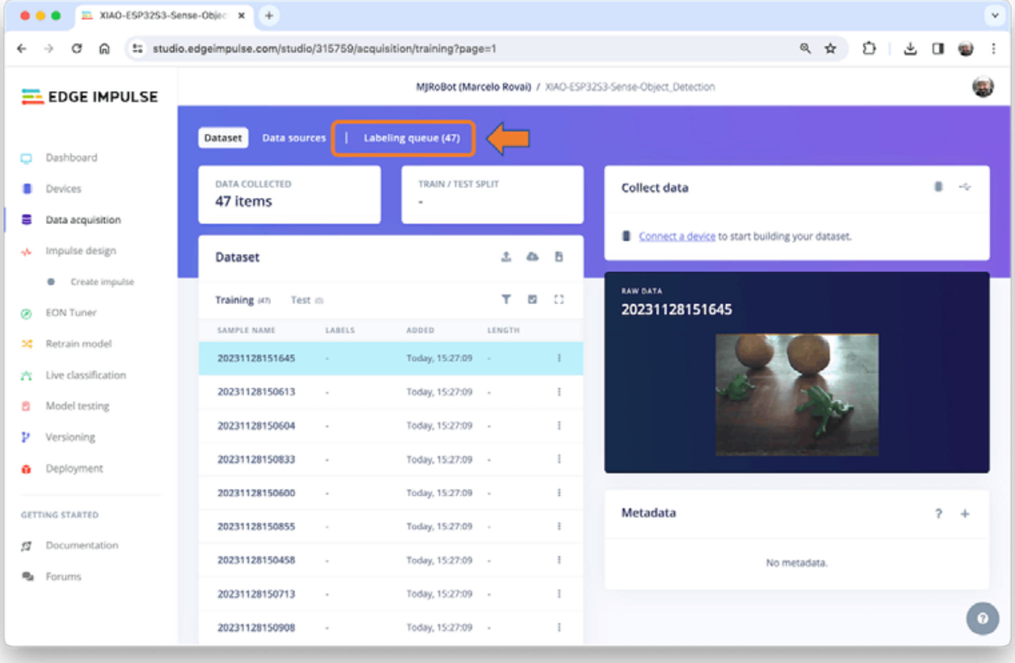


studio.edgeimpulse.com/studio/315759/acquisition/training?page=1#upload

Upload data

You can upload existing data to your project in the Data Acquisition Format (CBOR, JSON, CSV), or as WAV, JPG, PNG, AVI or MP4 files. We also support uploading image datasets with labels in various formats. When you include labels during upload, we attempt to convert your dataset into a format recognized by Studio. here.

You can leave for the Studio to split your data automatically between Train and Test or do it manually. We will upload all of them as training.



studio.edgeimpulse.com/studio/315759/acquisition/training?page=1

MJRoBot (Marcelo Rovai) / XIAO-ESP32S3-Sense-Object_Detection

DATA COLLECTED
47 items

TRAIN / TEST SPLIT
-

Collect data

Connect a device to start building your dataset.

Dataset

Training (47) Test (0)

SAMPLE NAME	LABELS	ADDED	LENGTH
20231128151645	-	Today, 15:27:09	-
20231128150613	-	Today, 15:27:09	-
20231128150604	-	Today, 15:27:09	-
20231128150833	-	Today, 15:27:09	-
20231128150600	-	Today, 15:27:09	-
20231128150855	-	Today, 15:27:09	-
20231128150458	-	Today, 15:27:09	-
20231128150713	-	Today, 15:27:09	-
20231128150908	-	Today, 15:27:09	-

RAW DATA
20231128151645

Metadata

No metadata.

All the not-labeled images (47) were uploaded but must be labeled appropriately before being used as a project dataset. The Studio has a tool for that purpose, which you can find in the link Labeling queue (47).

There are two ways you can use to perform AI-assisted labeling on the Edge Impulse Studio (free version):

- Using yolov5
- Tracking objects between frames

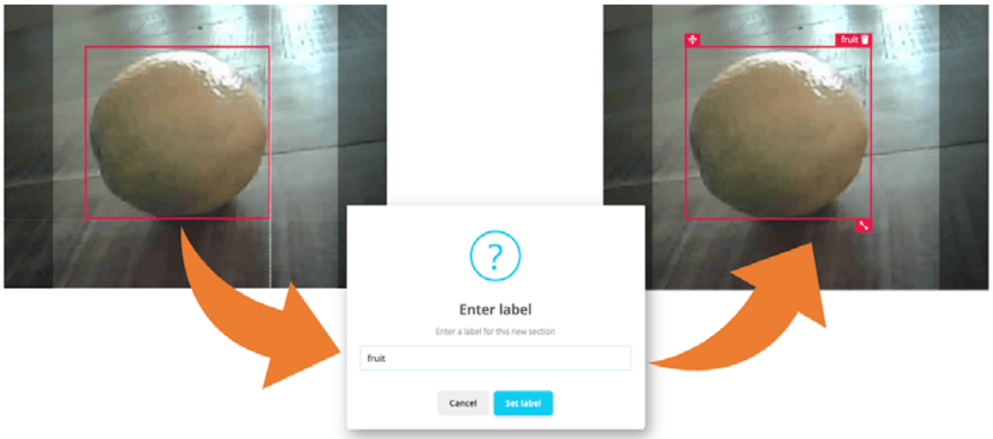
> *Edge Impulse launched an auto-labeling feature for Enterprise customers, easing labeling tasks in object detection projects.*

Ordinary objects can quickly be identified and labeled using an existing library of pre-trained object detection models from YOLOv5 (trained with the COCO dataset). But since, in our case, the objects are not part of COCO datasets, we should select the option of tracking objects. With this option, once you draw bounding boxes and label the images in one frame, the objects will be tracked automatically from frame to frame, partially labeling the new ones (not all are correctly labeled).

> *You can use the EI uploader to import your data if you already have a labeled dataset containing bounding boxes.*

Labeling the Dataset

Starting with the first image of your unlabeled data, use your mouse to drag a box around an object to add a label. Then click Save labels to advance to the next item.

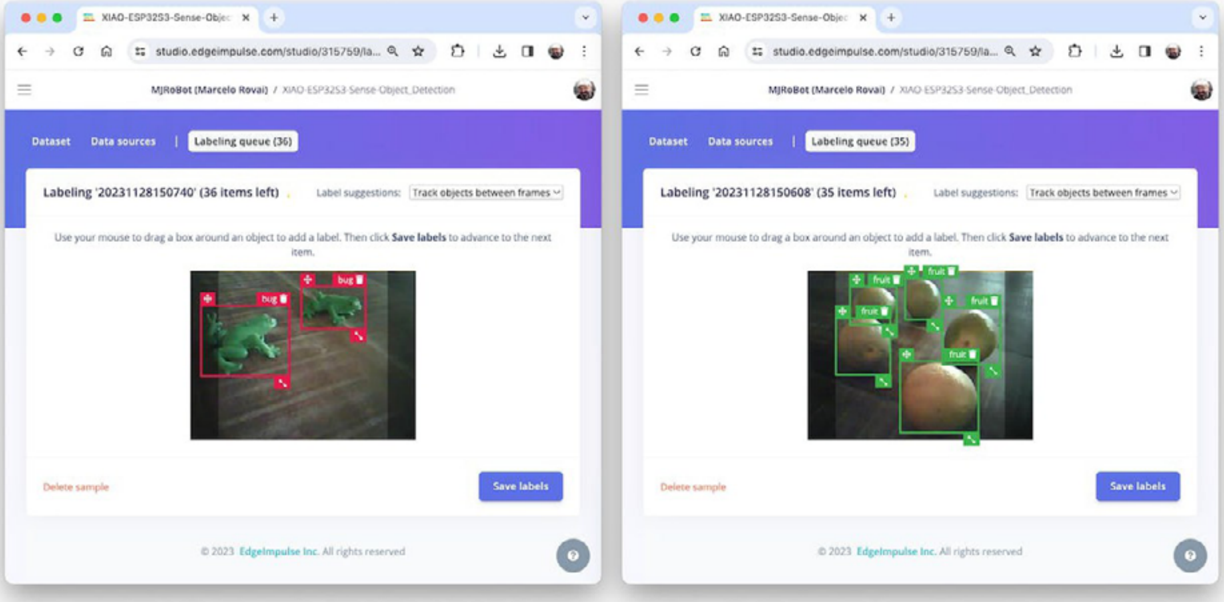


Enter label

Enter a label for this new section

fruit

Continue with this process until the queue is empty. At the end, all images should have the objects labeled as those samples below:



XIAO-ESP32S3-Sense-Objec

studio.edgeimpulse.com/studio/315759/la...

MJRoBot (Marcelo Rovai) / XIAO-ESP32S3-Sense-Object_Detection

Labeling '20231128150740' (36 items left)

Label suggestions: Track objects between frames

Use your mouse to drag a box around an object to add a label. Then click **Save labels** to advance to the next item.

© 2023 EdgeImpulse Inc. All rights reserved

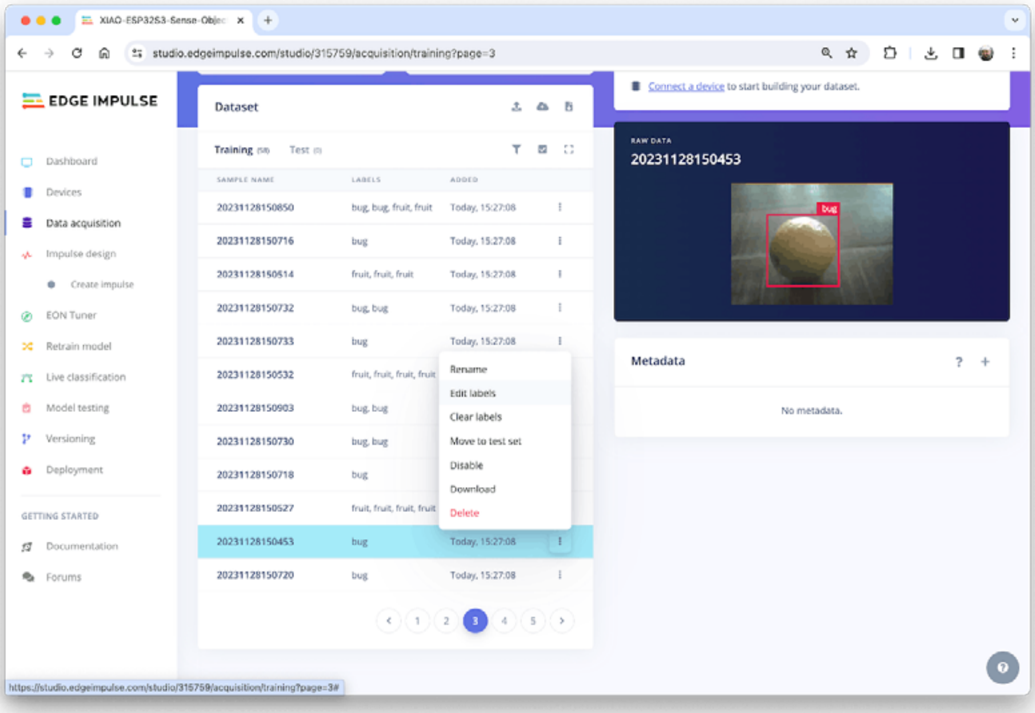
Labeling '20231128150608' (35 items left)

Label suggestions: Track objects between frames

Use your mouse to drag a box around an object to add a label. Then click **Save labels** to advance to the next item.

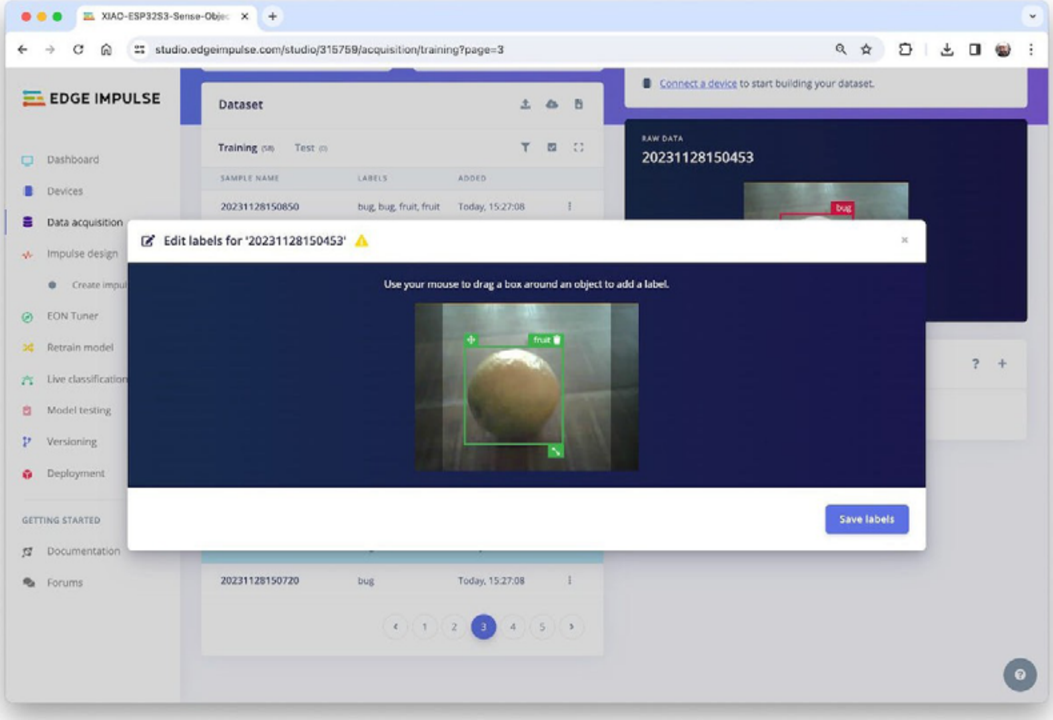
© 2023 EdgeImpulse Inc. All rights reserved

Next, review the labeled samples on the `Data acquisition` tab. If one of the labels is wrong, you can edit it using the three dots menu after the sample name:



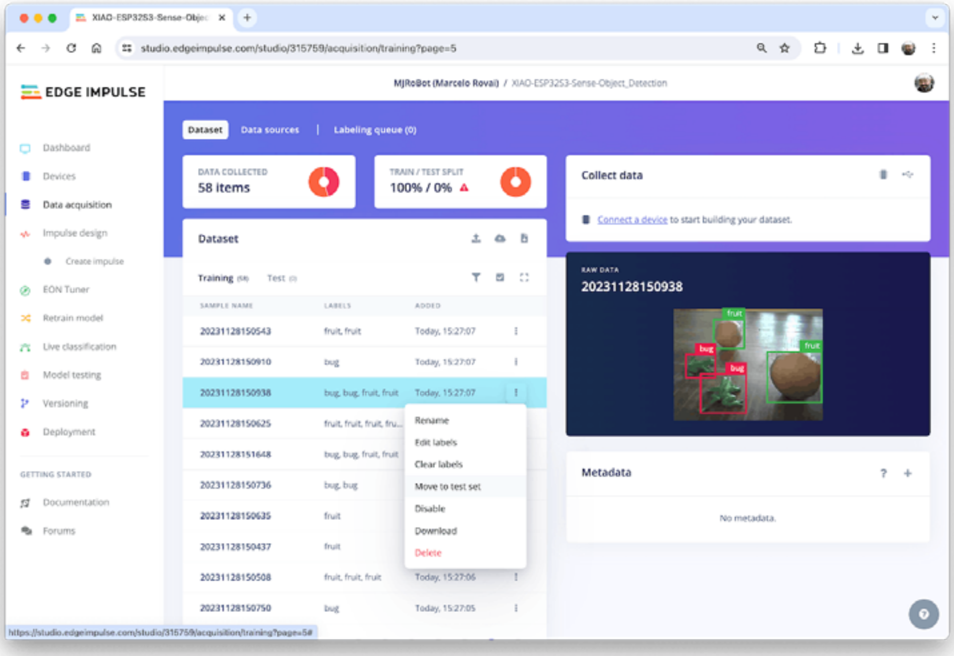
SAMPLE NAME	LABELS	ADDED
20231128150850	bug, bug, fruit, fruit	Today, 15:27:08
20231128150716	bug	Today, 15:27:08
20231128150514	fruit, fruit, fruit	Today, 15:27:08
20231128150732	bug, bug	Today, 15:27:08
20231128150733	bug	Today, 15:27:08
20231128150532	fruit, fruit, fruit, fruit	
20231128150903	bug, bug	
20231128150730	bug, bug	
20231128150718	bug	
20231128150527	fruit, fruit, fruit, fruit	
20231128150453	bug	Today, 15:27:08
20231128150720	bug	Today, 15:27:08

You will be guided to replace the wrong label and correct the dataset.


XIAO-ESP32S3-Sense-Objec
studio.edgeimpulse.com/studio/315759/acquisition/training?page=3
EDGE IMPULSE
Dataset
Training (58)
Test (0)
RAW DATA
20231128150453
Edit labels for '20231128150453'
Use your mouse to drag a box around an object to add a label.
fruit
Save labels

Balancing the dataset and split Train/Test

After labeling all data, it was realized that the class fruit had many more samples than the bug. So, 11 new and additional bug images were collected (ending with 58 images). After labeling them, it is time to select some images and move them to the test dataset. You can do it using the three-dot menu after the image name. I selected six images, representing 13% of the total dataset.



DATA COLLECTED 58 items

TRAIN / TEST SPLIT 100% / 0%

Collect data

Connect a device to start building your dataset.

Dataset

Training (58) Test (0)

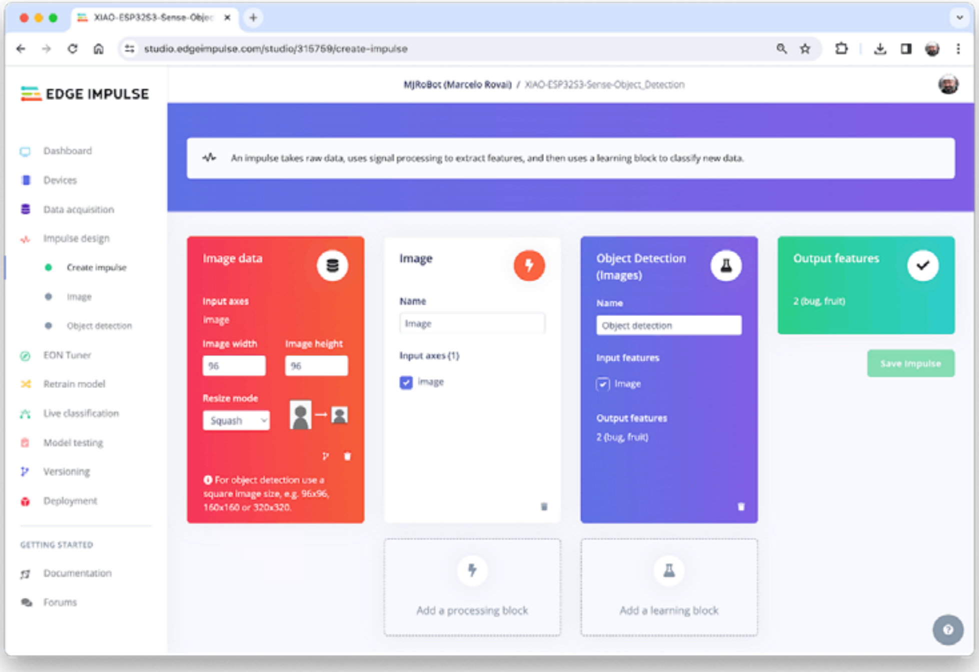
SAMPLE NAME	LABELS	ADDED
20231128150543	fruit, fruit	Today, 15:27:07
20231128150910	bug	Today, 15:27:07
20231128150938	bug, bug, fruit, fruit	Today, 15:27:07
20231128150625	fruit, fruit, fruit, fru...	
20231128151648	bug, bug, fruit, fruit	
20231128150736	bug, bug	
20231128150635	fruit	
20231128150437	fruit	
20231128150508	fruit, fruit, fruit	Today, 15:27:06
20231128150750	bug	Today, 15:27:05

RAW DATA 20231128150938

4.5.5 The Impulse Design

In this phase, you should define how to:

- **Pre-processing** consists of resizing the individual images from 320 x 240 to 96 x 96 and squashing them (squared form, without cropping). Afterward, the images are converted from RGB to Grayscale.
- **Design a Model**, in this case, "Object Detection."



An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Image data
Input axes: image
Image width: 96
Image height: 96
Resize mode: Squash
For object detection use a square image size, e.g. 96x96, 160x160 or 320x320.

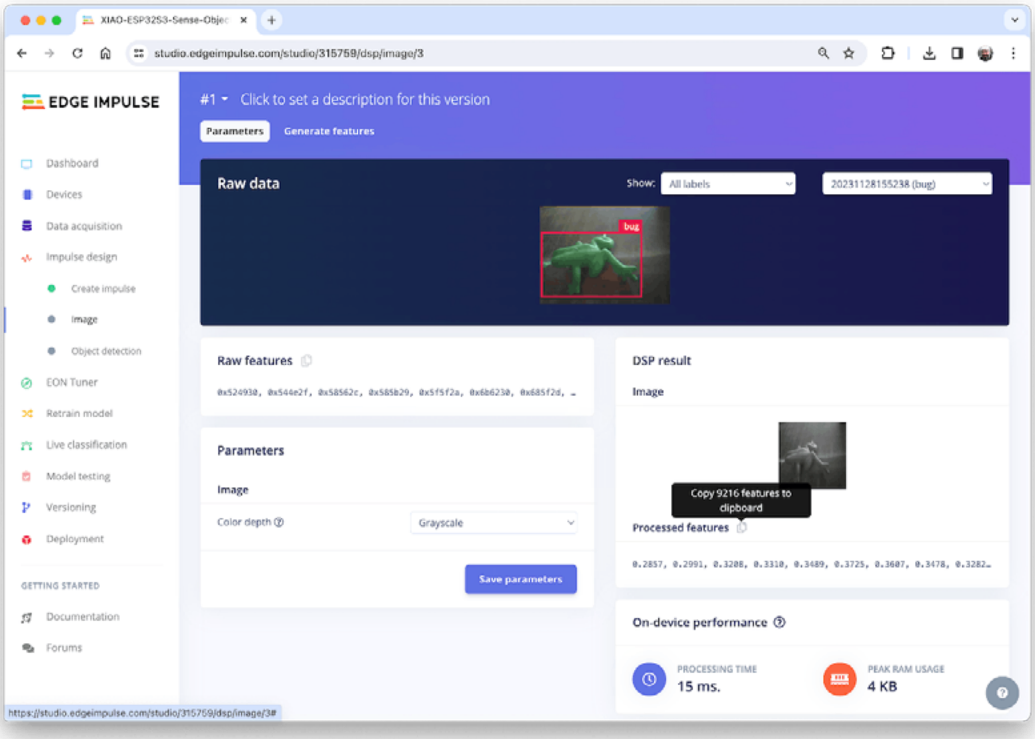
Image
Name: Image
Input axes (1): ☑ image

Object Detection (Images)
Name: Object detection
Input features: ☑ Image
Output features: 2 (bug, fruit)

Output features: 2 (bug, fruit)

Preprocessing all dataset

In this section, select **Color depth** as Grayscale, suitable for use with FOMO models and Save parameters.



Raw data

Show: All labels | 20231128155238 (bug)

Raw features

```
0x524930, 0x544e2f, 0x58562c, 0x585b29, 0x5f5f2a, 0x6b6230, 0x685f2d, …
```

Parameters

Image

Color depth: Grayscale

DSP result

Image

Processed features

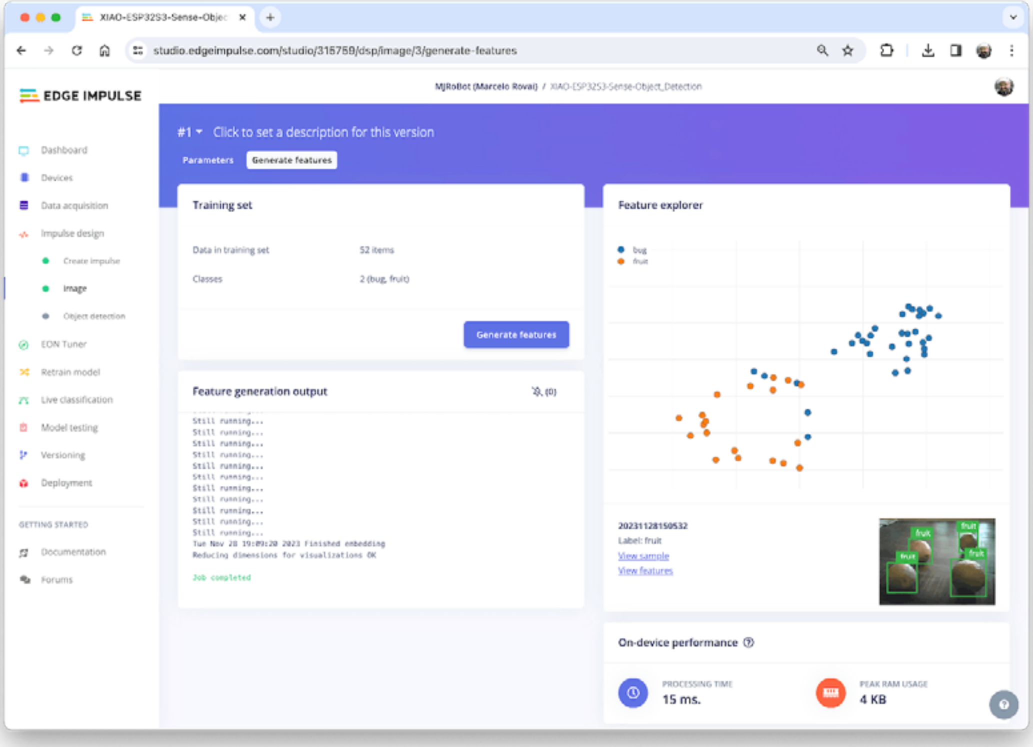
```
0.2857, 0.2991, 0.3208, 0.3310, 0.3489, 0.3725, 0.3607, 0.3478, 0.3282…
```

On-device performance

PROCESSING TIME 15 ms.

PEAK RAM USAGE 4 KB

The Studio moves automatically to the next section, Generate features, where all samples will be pre-processed, resulting in a dataset with individual 96x96x1 images or 9,216 features.



Training set

Data in training set: 52 items

Classes: 2 (bug, fruit)

Feature generation output

```
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Still running...
Tue Nov 28 19:09:20 2023 Finished embedding
Reducing dimensions for visualizations OK

Job completed
```

Feature explorer

20231128150532

Label: fruit

On-device performance

PROCESSING TIME 15 ms.

PEAK RAM USAGE 4 KB

The feature explorer shows that all samples evidence a good separation after the feature generation.

> *Some samples seem to be in the wrong space, but clicking on them confirms the correct labeling.*

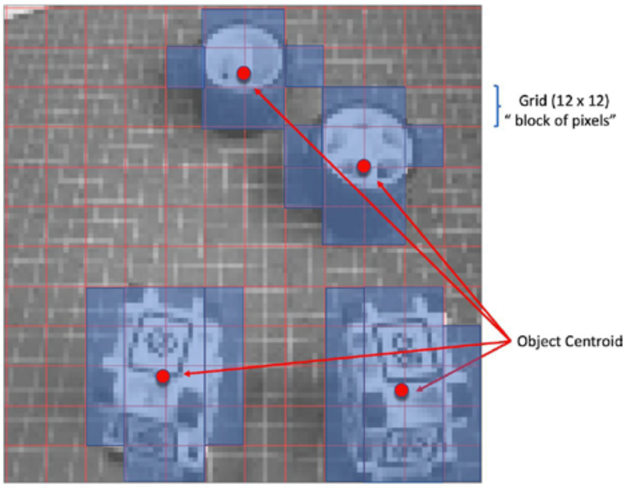
4.5.6 Model Design, Training, and Test

We will use FOMO, an object detection model based on MobileNetV2 (alpha 0.35) designed to coarsely segment an image into a grid of **background** vs **objects of interest** (here, boxes and wheels).

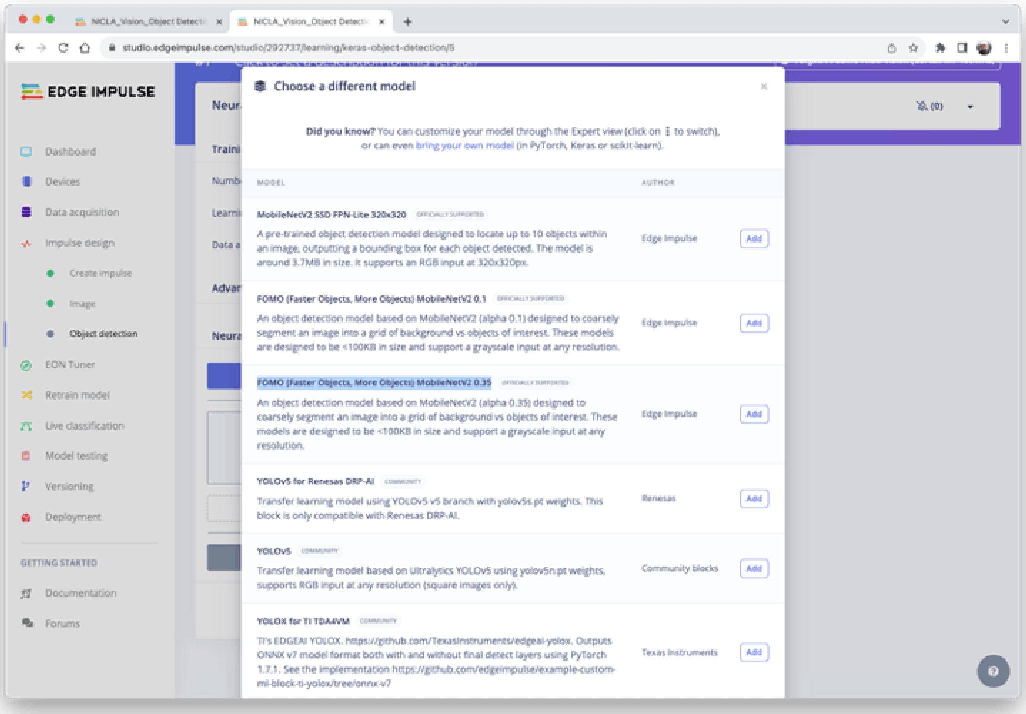
FOMO is an innovative machine learning model for object detection, which can use up to 30 times less energy and memory than traditional models like Mobilenet SSD and YOLOv5. FOMO can operate on microcontrollers with less than 200 KB of RAM. The main reason this is possible is that while other models calculate the object's size by drawing a square around it (bounding box), FOMO ignores the size of the image, providing only the information about where the object is located in the image through its centroid coordinates.

How FOMO works?

FOMO takes the image in grayscale and divides it into blocks of pixels using a factor of 8. For the input of 96x96, the grid would be 12x12 (96/8=12). Next, FOMO will run a classifier through each pixel block to calculate the probability that there is a box or a wheel in each of them and, subsequently, determine the regions that have the highest probability of containing the object (If a pixel block has no objects, it will be classified as background). From the overlap of the final region, the FOMO provides the coordinates (related to the image dimensions) of the centroid of this region.



For training, we should select a pre-trained model. **Let's use the FOMO (Faster Objects, More Objects) MobileNetV2 0.35.** This model uses around 250KB of RAM and 80KB of ROM (Flash), which suits well with our board.



Choose a different model

Did you know? You can customize your model through the Expert view (click on ⋮ to switch), or can even bring your own model (in PyTorch, Keras or scikit-learn).

MODEL	AUTHOR	
MobileNetV2 SSD FPN-Lite 320x320 OFFICIALLY SUPPORTED A pre-trained object detection model designed to locate up to 10 objects within an image, outputting a bounding box for each object detected. The model is around 3.7MB in size. It supports an RGB input at 320x320px.	Edge Impulse	Add
FOMO (Faster Objects, More Objects) MobileNetV2 0.1 OFFICIALLY SUPPORTED An object detection model based on MobileNetV2 (alpha 0.1) designed to coarsely segment an image into a grid of background vs objects of interest. These models are designed to be <100KB in size and support a grayscale input at any resolution.	Edge Impulse	Add
FOMO (Faster Objects, More Objects) MobileNetV2 0.35 OFFICIALLY SUPPORTED An object detection model based on MobileNetV2 (alpha 0.35) designed to coarsely segment an image into a grid of background vs objects of interest. These models are designed to be <100KB in size and support a grayscale input at any resolution.	Edge Impulse	Add
YOLOv5 for Renesas DRP-AI COMMUNITY Transfer learning model using YOLOv5 v5 branch with yolov5s.pt weights. This block is only compatible with Renesas DRP-AI.	Renesas	Add
YOLOv5 COMMUNITY Transfer learning model based on Ultralytics YOLOv5 using yolov5n.pt weights, supports RGB input at any resolution (square images only).	Community blocks	Add
YOLOX for TI TDA4VM COMMUNITY TI's EDGEAI YOLOX. https://github.com/TexasInstruments/edgeai-yolox. Outputs ONNX v7 model format both with and without final detect layers using PyTorch 1.7.1. See the implementation https://github.com/edgeimpulse/example-custom-ml-block-ti-yolox/tree/onnx-v7	Texas Instruments	Add

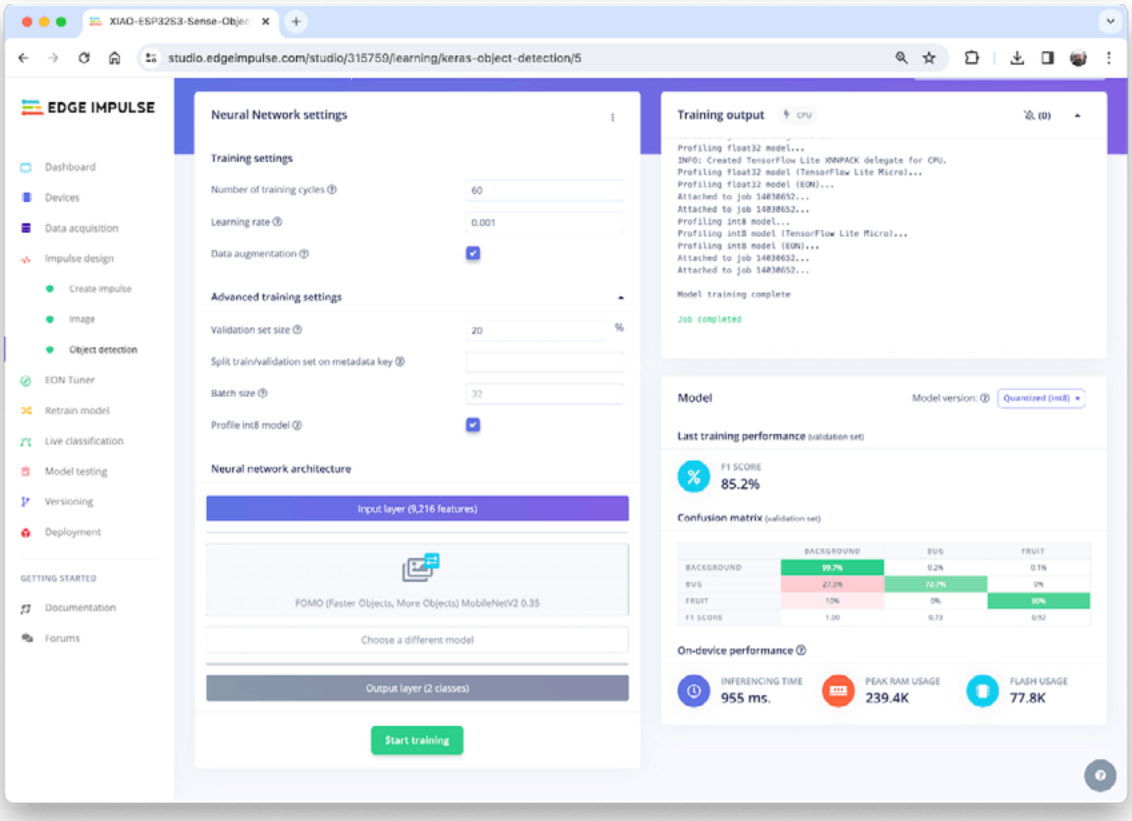
Regarding the training hyper-parameters, the model will be trained with:

- Epochs: 60
- Batch size: 32
- Learning Rate: 0.001.

For validation during training, 20% of the dataset (validation_dataset) will be spared. For the remaining 80% (train_dataset), we will apply Data Augmentation, which will randomly flip, change the size and brightness of the image, and crop them, artificially increasing the number of samples on the dataset for training.

As a result, the model ends with an overall F1 score of 85%, similar to the result when using the test data (83%).

> *Note that FOMO automatically added a 3rd label background to the two previously defined (box and wheel).*



Neural Network settings

Training settings

Number of training cycles: 60

Learning rate: 0.001

Data augmentation: ☑

Advanced training settings

Validation set size: 20 %

Split train/validation set on metadata key:

Batch size: 32

Profile int8 model: ☑

Neural network architecture

Input layer (9,216 features)

FOMO (Faster Objects, More Objects) MobileNetV2 0.35

Choose a different model

Output layer (2 classes)

Start training

Training output CPU

```
Profiling float32 model...
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
Profiling float32 model (TensorFlow Lite Micro)...
Profiling float32 model (EON)...
Attached to job 14030652...
Attached to job 14030652...
Profiling int8 model...
Profiling int8 model (TensorFlow Lite Micro)...
Profiling int8 model (EON)...
Attached to job 14030652...
Attached to job 14030652...

Model training complete

Job completed
```

Model

Model version: Quantized (int8)

Last training performance (validation set)

F1 SCORE 85.2%

Confusion matrix (validation set)

	BACKGROUND	BUG	FRUIT
BACKGROUND	99.7%	0.2%	0.1%
BUG	27.3%	72.7%	0%
FRUIT	10%	0%	90%
F1 SCORE	1.00	0.73	0.92

On-device performance

INFERENCING TIME 955 ms.

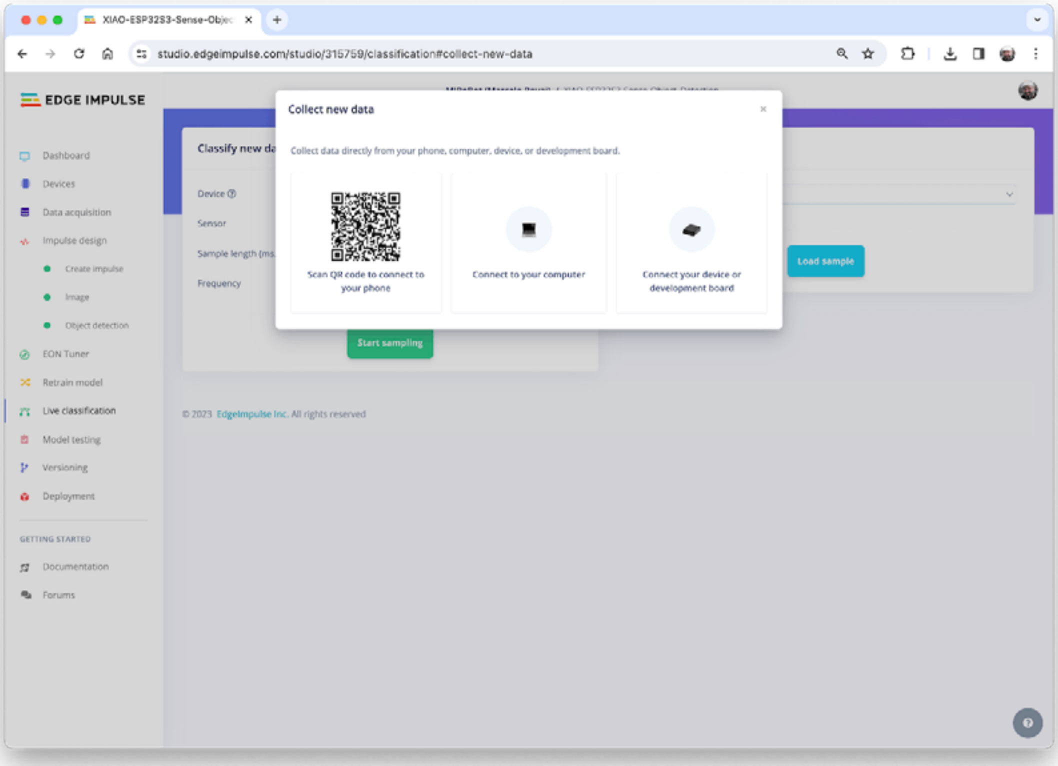
PEAK RAM USAGE 239.4K

FLASH USAGE 77.8K

> *In object detection tasks, accuracy is generally not the primary evaluation metric. Object detection involves classifying objects and providing bounding boxes around them, making it a more complex problem than simple classification. The issue is that we do not have the bounding box, only the centroids. In short, using accuracy as a metric could be misleading and may not provide a complete understanding of how well the model is performing. Because of that, we will use the F1 score.*

Test model with "Live Classification"

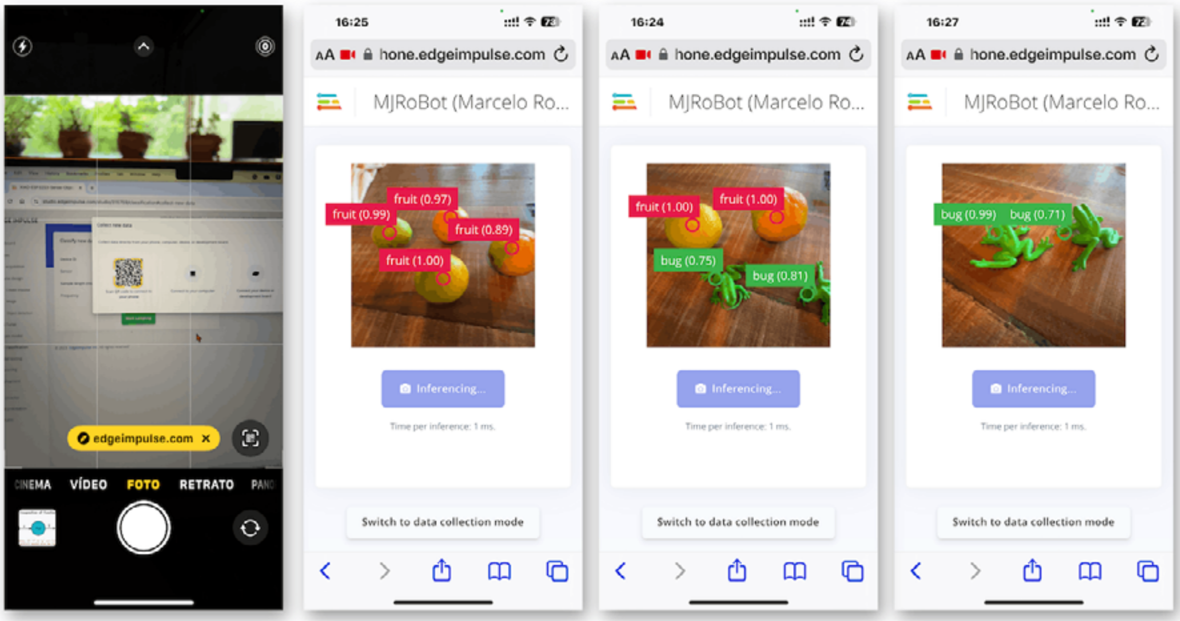
Once our model is trained, we can test it using the Live Classification tool. On the correspondent section, click on Connect a development board icon (a small MCU) and scan the QR code with your phone.



Collect new data

Collect data directly from your phone, computer, device, or development board.

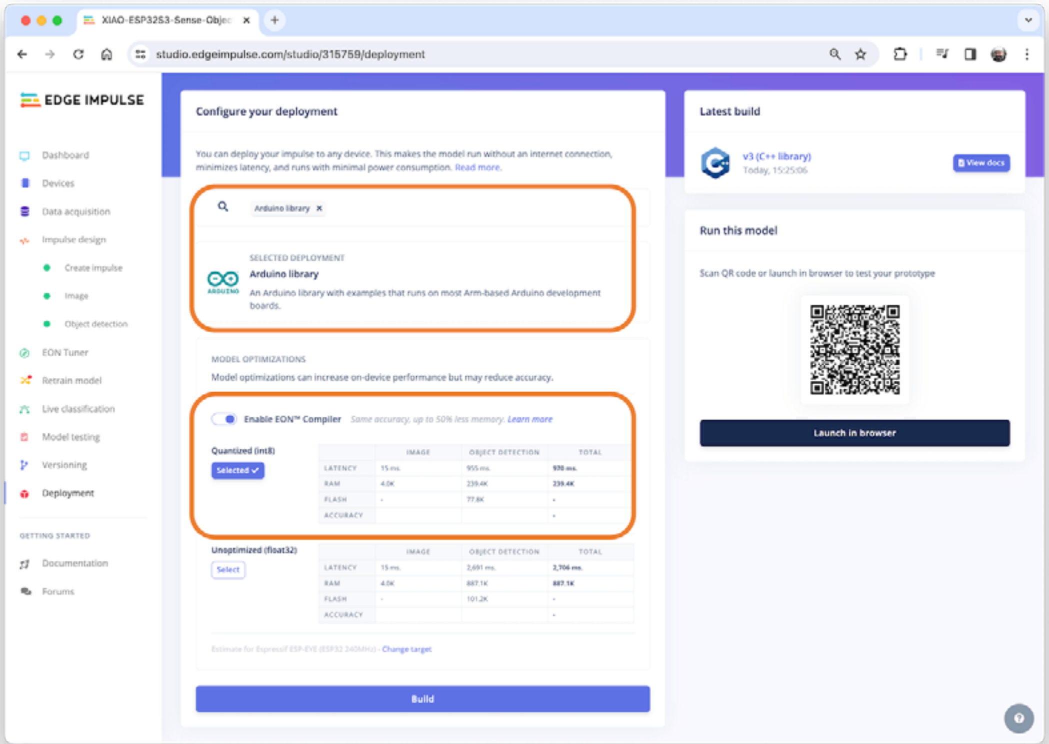
Once connected, you can use the smartphone to capture actual images to be tested by the trained model on Edge Impulse Studio.



One thing to be noted is that the model can produce false positives and negatives. This can be minimized by defining a proper Confidence Threshold (use the Three dots menu for the setup). Try with 0.8 or more.

4.5.7 Deploying the Model (Arduino IDE)

Select the Arduino Library and Quantized (int8) model, enable the EON Compiler on the Deploy Tab, and press [Build].



Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Arduino library ×

SELECTED DEPLOYMENT
Arduino library
An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.

Enable EON™ Compiler *Same accuracy, up to 50% less memory. Learn more*

Quantized (int8) — Selected ✓

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	15 ms.	955 ms.	970 ms.
RAM	4.0K	239.4K	239.4K
FLASH	-	77.8K	-
ACCURACY			-

Unoptimized (float32) — Select

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	15 ms.	2,691 ms.	2,706 ms.
RAM	4.0K	887.1K	887.1K
FLASH	-	101.2K	-
ACCURACY			-

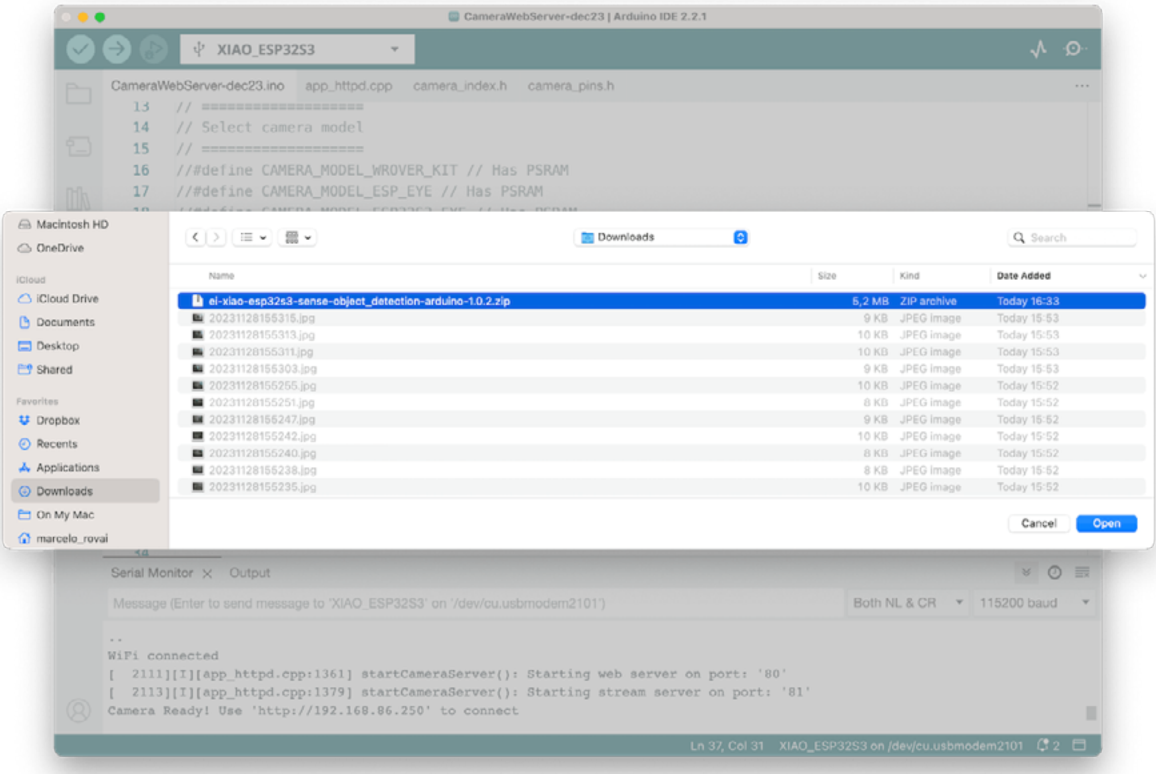
Estimate for Espressif ESP-EYE (ESP32 240MHz) - Change target

Build

Latest build
v3 (C++ library)
Today, 15:25:06
View docs

Run this model
Scan QR code or launch in browser to test your prototype
Launch in browser

Open your Arduino IDE, and under Sketch, go to Include Library and add.ZIP Library. Select the file you download from Edge Impulse Studio, and that's it!



```
// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
```

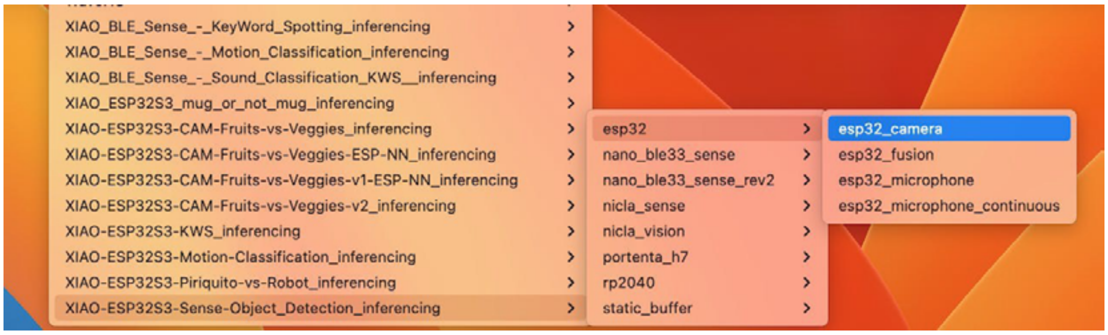
Name	Size	Kind	Date Added
ei-xiao-esp32s3-sense-object_detection-arduino-1.0.2.zip	5,2 MB	ZIP archive	Today 16:33
20231128155315.jpg	9 KB	JPEG image	Today 15:53
20231128155313.jpg	10 KB	JPEG image	Today 15:53
20231128155311.jpg	10 KB	JPEG image	Today 15:53
20231128155303.jpg	9 KB	JPEG image	Today 15:53
20231128155255.jpg	10 KB	JPEG image	Today 15:52
20231128155251.jpg	8 KB	JPEG image	Today 15:52
20231128155247.jpg	9 KB	JPEG image	Today 15:52
20231128155242.jpg	10 KB	JPEG image	Today 15:52
20231128155240.jpg	8 KB	JPEG image	Today 15:52
20231128155238.jpg	8 KB	JPEG image	Today 15:52
20231128155235.jpg	10 KB	JPEG image	Today 15:52

Message (Enter to send message to 'XIAO_ESP32S3' on '/dev/cu.usbmodem2101')
Both NL & CR
115200 baud

```
..
WiFi connected
[  2111][I][app_httpd.cpp:1361] startCameraServer(): Starting web server on port: '80'
[  2113][I][app_httpd.cpp:1379] startCameraServer(): Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.86.250' to connect
```

Ln 37, Col 31   XIAO_ESP32S3 on /dev/cu.usbmodem2101

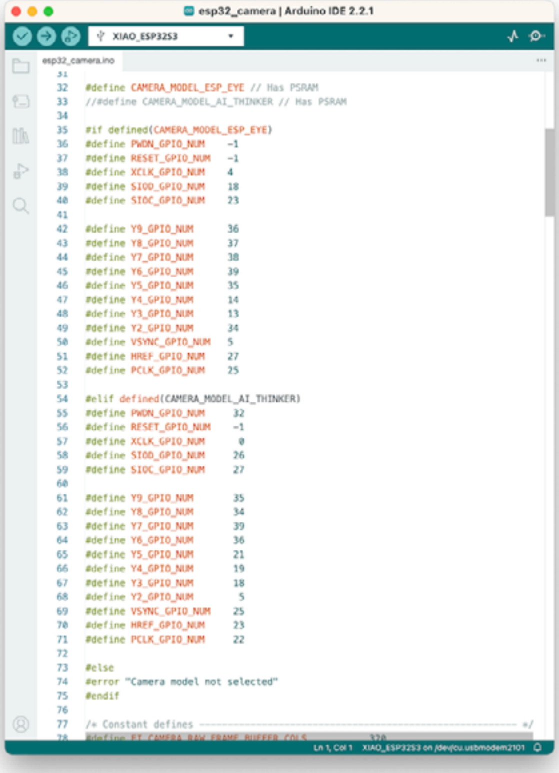
Under the Examples tab on Arduino IDE, you should find a sketch code (`esp32 > esp32_camera`) under your project name.


XIAO_BLE_Sense_-_KeyWord_Spotting_inferencing
XIAO_BLE_Sense_-_Motion_Classification_inferencing
XIAO_BLE_Sense_-_Sound_Classification_KWS__inferencing
XIAO_ESP32S3_mug_or_not_mug_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies-ESP-NN_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v1-ESP-NN_inferencing
XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v2_inferencing
XIAO-ESP32S3-KWS_inferencing
XIAO-ESP32S3-Motion-Classification_inferencing
XIAO-ESP32S3-Piriquito-vs-Robot_inferencing
XIAO-ESP32S3-Sense-Object_Detection_inferencing
esp32
nano_ble33_sense
nano_ble33_sense_rev2
nicla_sense
nicla_vision
portenta_h7
rp2040
static_buffer
esp32_camera
esp32_fusion
esp32_microphone
esp32_microphone_continuous

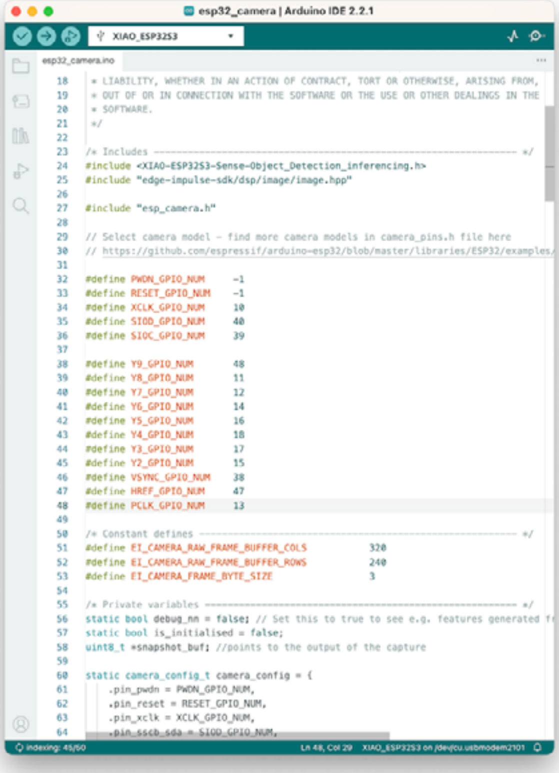
You should change lines 32 to 75, which define the camera model and pins, using the data related to our model. Copy and paste the below lines, replacing the lines 32-75:

```
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
#define Y9_GPIO_NUM       48
#define Y8_GPIO_NUM       11
#define Y7_GPIO_NUM       12
#define Y6_GPIO_NUM       14
#define Y5_GPIO_NUM       16
#define Y4_GPIO_NUM       18
#define Y3_GPIO_NUM       17
#define Y2_GPIO_NUM       15
#define VSYNC_GPIO_NUM    38
#define HREF_GPIO_NUM     47
#define PCLK_GPIO_NUM     13
```

Here you can see the resulting code:


esp32_camera | Arduino IDE 2.2.1
XIAO_ESP32S3
esp32_camera.ino
Ln 1, Col 1   XIAO_ESP32S3 on /dev/cu.usbmodem2101

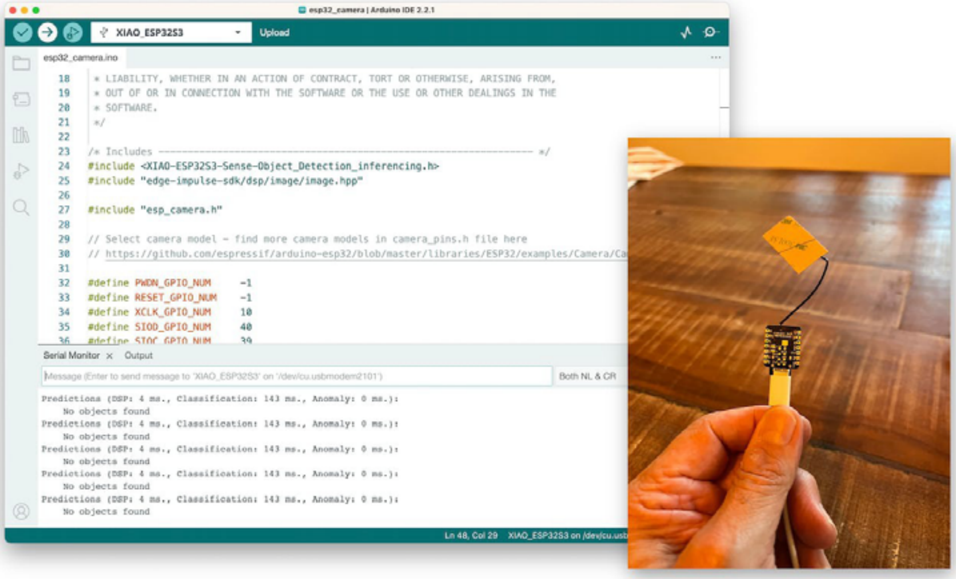
```
31
32  #define CAMERA_MODEL_ESP_EYE // Has PSRAM
33  //#define CAMERA_MODEL_AI_THINKER // Has PSRAM
34
35  #if defined(CAMERA_MODEL_ESP_EYE)
36  #define PWDN_GPIO_NUM    -1
37  #define RESET_GPIO_NUM   -1
38  #define XCLK_GPIO_NUM    4
39  #define SIOD_GPIO_NUM    18
40  #define SIOC_GPIO_NUM    23
41
42  #define Y9_GPIO_NUM      36
43  #define Y8_GPIO_NUM      37
44  #define Y7_GPIO_NUM      38
45  #define Y6_GPIO_NUM      39
46  #define Y5_GPIO_NUM      35
47  #define Y4_GPIO_NUM      14
48  #define Y3_GPIO_NUM      13
49  #define Y2_GPIO_NUM      34
50  #define VSYNC_GPIO_NUM   5
51  #define HREF_GPIO_NUM    27
52  #define PCLK_GPIO_NUM    25
53
54  #elif defined(CAMERA_MODEL_AI_THINKER)
55  #define PWDN_GPIO_NUM     32
56  #define RESET_GPIO_NUM    -1
57  #define XCLK_GPIO_NUM      0
58  #define SIOD_GPIO_NUM     26
59  #define SIOC_GPIO_NUM     27
60
61  #define Y9_GPIO_NUM       35
62  #define Y8_GPIO_NUM       34
63  #define Y7_GPIO_NUM       39
64  #define Y6_GPIO_NUM       36
65  #define Y5_GPIO_NUM       21
66  #define Y4_GPIO_NUM       19
67  #define Y3_GPIO_NUM       18
68  #define Y2_GPIO_NUM        5
69  #define VSYNC_GPIO_NUM    25
70  #define HREF_GPIO_NUM     23
71  #define PCLK_GPIO_NUM     22
72
73  #else
74  #error "Camera model not selected"
75  #endif
76
77  /* Constant defines -------------------------------------------------------- */
78  #define EI_CAMERA_RAW_FRAME_BUFFER_COLS           320
```


esp32_camera | Arduino IDE 2.2.1
XIAO_ESP32S3
esp32_camera.ino
indexing: 45/60
Ln 48, Col 29   XIAO_ESP32S3 on /dev/cu.usbmodem2101

```
18   * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
19   * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
20   * SOFTWARE.
21   */
22
23  /* Includes ---------------------------------------------------------------- */
24  #include <XIAO-ESP32S3-Sense-Object_Detection_inferencing.h>
25  #include "edge-impulse-sdk/dsp/image/image.hpp"
26
27  #include "esp_camera.h"
28
29  // Select camera model - find more camera models in camera_pins.h file here
30  // https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/
31
32  #define PWDN_GPIO_NUM     -1
33  #define RESET_GPIO_NUM    -1
34  #define XCLK_GPIO_NUM     10
35  #define SIOD_GPIO_NUM     40
36  #define SIOC_GPIO_NUM     39
37
38  #define Y9_GPIO_NUM       48
39  #define Y8_GPIO_NUM       11
40  #define Y7_GPIO_NUM       12
41  #define Y6_GPIO_NUM       14
42  #define Y5_GPIO_NUM       16
43  #define Y4_GPIO_NUM       18
44  #define Y3_GPIO_NUM       17
45  #define Y2_GPIO_NUM       15
46  #define VSYNC_GPIO_NUM    38
47  #define HREF_GPIO_NUM     47
48  #define PCLK_GPIO_NUM     13
49
50  /* Constant defines -------------------------------------------------------- */
51  #define EI_CAMERA_RAW_FRAME_BUFFER_COLS           320
52  #define EI_CAMERA_RAW_FRAME_BUFFER_ROWS           240
53  #define EI_CAMERA_FRAME_BYTE_SIZE                 3
54
55  /* Private variables ------------------------------------------------------- */
56  static bool debug_nn = false; // Set this to true to see e.g. features generated f
57  static bool is_initialised = false;
58  uint8_t *snapshot_buf; //points to the output of the capture
59
60  static camera_config_t camera_config = {
61      .pin_pwdn = PWDN_GPIO_NUM,
62      .pin_reset = RESET_GPIO_NUM,
63      .pin_xclk = XCLK_GPIO_NUM,
64      .pin_sscb_sda = SIOD_GPIO_NUM,
```

Upload the code to your XIAO ESP32S3 Sense, and you should be OK to start detecting fruits and bugs. You can check the result on Serial Monitor.

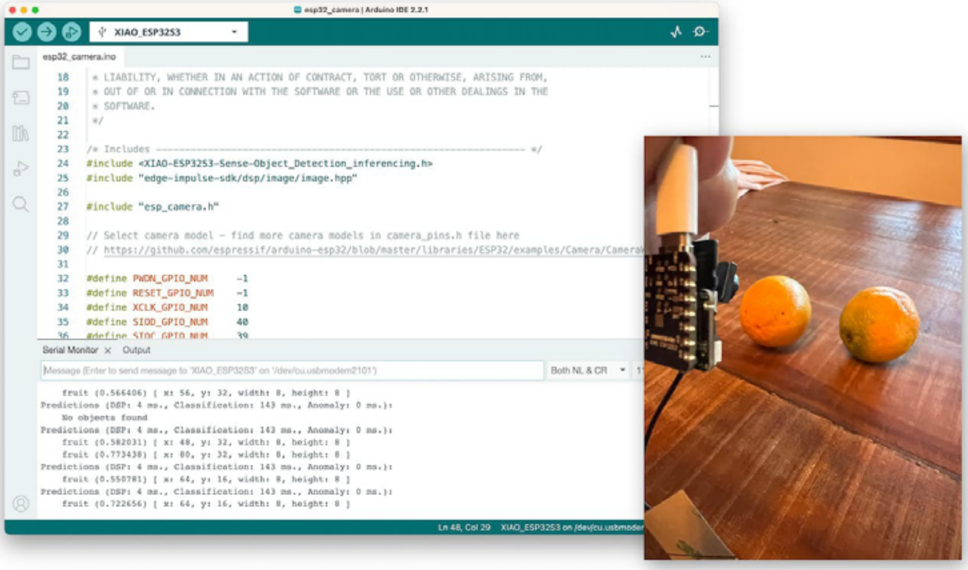
Background


esp32_camera | Arduino IDE 2.2.1
XIAO_ESP32S3
Upload
esp32_camera.ino
Serial Monitor
Output
Message (Enter to send message to 'XIAO_ESP32S3' on '/dev/cu.usbmodem2101')
Both NL & CR
Ln 48, Col 29
XIAO_ESP32S3 on /dev/cu.us

```
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

/* Includes ---------------------------------------------------------------- */
#include <XIAO-ESP32S3-Sense-Object_Detection_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here
// https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera/Ca

#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
```

```
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
```

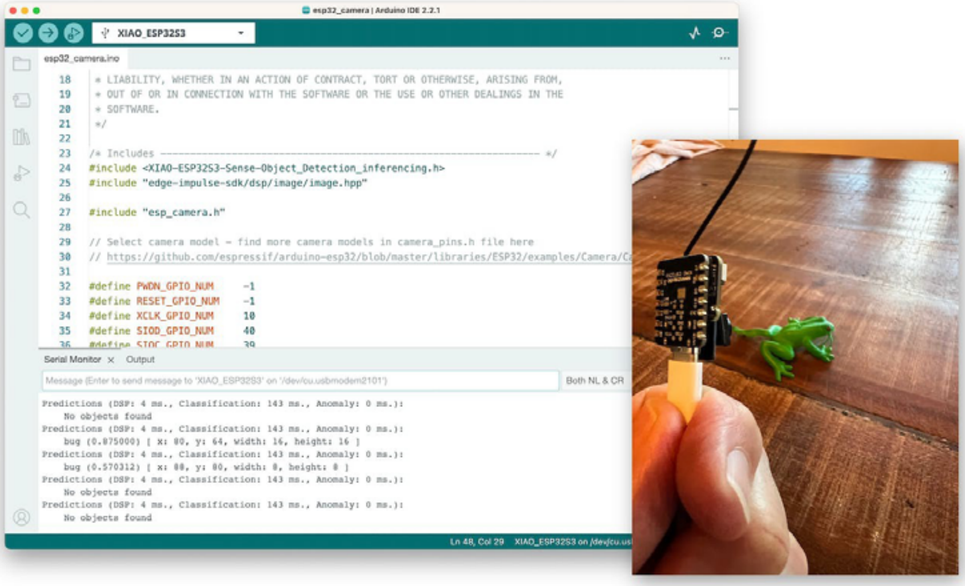
Fruits


esp32_camera | Arduino IDE 2.2.1
XIAO_ESP32S3
esp32_camera.ino
Serial Monitor
Output
Message (Enter to send message to 'XIAO_ESP32S3' on '/dev/cu.usbmodem2101')
Both NL & CR
Ln 48, Col 29
XIAO_ESP32S3 on /dev/cu.usbmoder

```
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

/* Includes ---------------------------------------------------------------- */
#include <XIAO-ESP32S3-Sense-Object_Detection_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here
// https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera/CameraW

#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
```

```
    fruit (0.566406) [ x: 56, y: 32, width: 8, height: 8 ]
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    fruit (0.582031) [ x: 48, y: 32, width: 8, height: 8 ]
    fruit (0.773438) [ x: 80, y: 32, width: 8, height: 8 ]
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    fruit (0.550781) [ x: 64, y: 16, width: 8, height: 8 ]
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    fruit (0.722656) [ x: 64, y: 16, width: 8, height: 8 ]
```

Bugs


esp32_camera | Arduino IDE 2.2.1
XIAO_ESP32S3
esp32_camera.ino
Serial Monitor
Output
Message (Enter to send message to 'XIAO_ESP32S3' on '/dev/cu.usbmodem2101')
Both NL & CR
Ln 48, Col 29
XIAO_ESP32S3 on /dev/cu.us

```
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 */

/* Includes ---------------------------------------------------------------- */
#include <XIAO-ESP32S3-Sense-Object_Detection_inferencing.h>
#include "edge-impulse-sdk/dsp/image/image.hpp"

#include "esp_camera.h"

// Select camera model - find more camera models in camera_pins.h file here
// https://github.com/espressif/arduino-esp32/blob/master/libraries/ESP32/examples/Camera/Ca

#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
```

```
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    bug (0.875000) [ x: 80, y: 64, width: 16, height: 16 ]
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    bug (0.570312) [ x: 88, y: 80, width: 8, height: 8 ]
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
Predictions (DSP: 4 ms., Classification: 143 ms., Anomaly: 0 ms.):
    No objects found
```

Note that the model latency is 143ms, and the frame rate per second is around 7 fps (similar to what we got with the Image Classification project). This happens because FOMO is cleverly built over a CNN model, not with an object detection model like the SSD MobileNet. For example, when running a MobileNetV2 SSD FPN-Lite 320x320 model on a Raspberry Pi 4, the latency is around five times higher (around 1.5 fps).

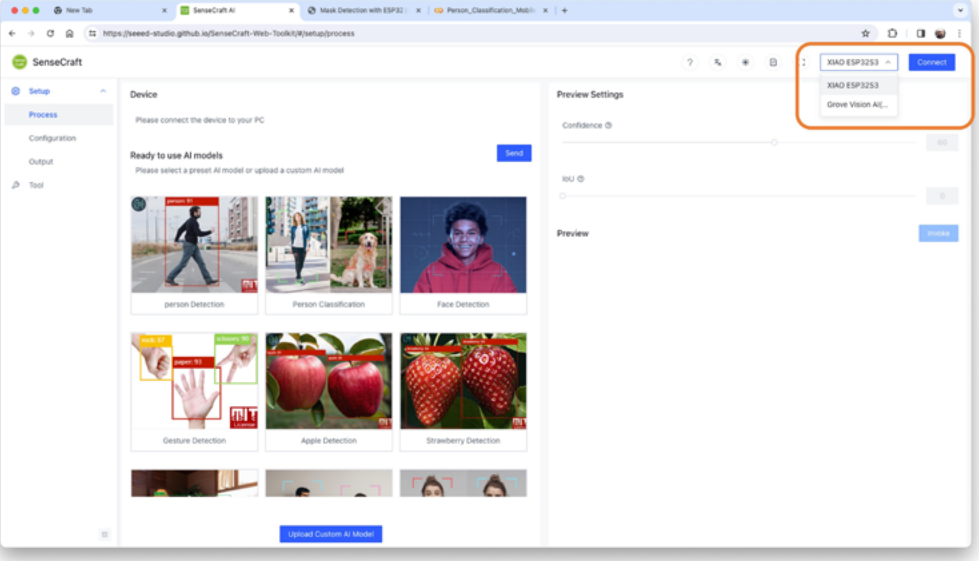
4.5.8 Deploying the Model (SenseCraft-Web-Toolkit)

As discussed in the Image Classification chapter, verifying inference with Image models on Arduino IDE is very challenging because we can not see what the camera focuses on. Again, let's use the **SenseCraft-Web Toolkit**.

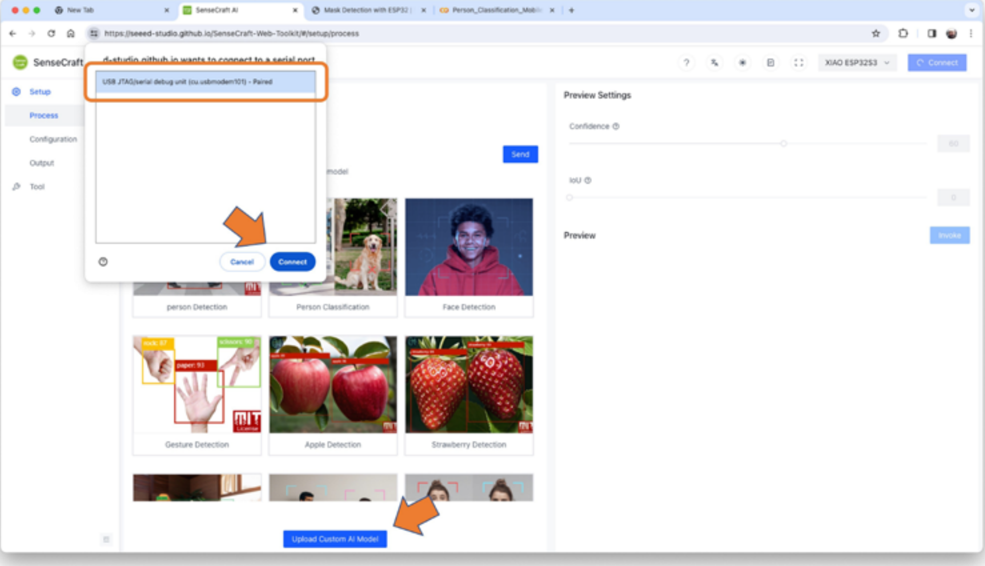
Follow the following steps to start the SenseCraft-Web-Toolkit:

1. Open the SenseCraft-Web-Toolkit website.
2. Connect the XIAO to your computer:

- Having the XIAO connected, select it as below:



- Select the device/Port and press `[Connect]`:



> *You can try several Computer Vision models previously uploaded by Seeed Studio. Try them and have fun!*

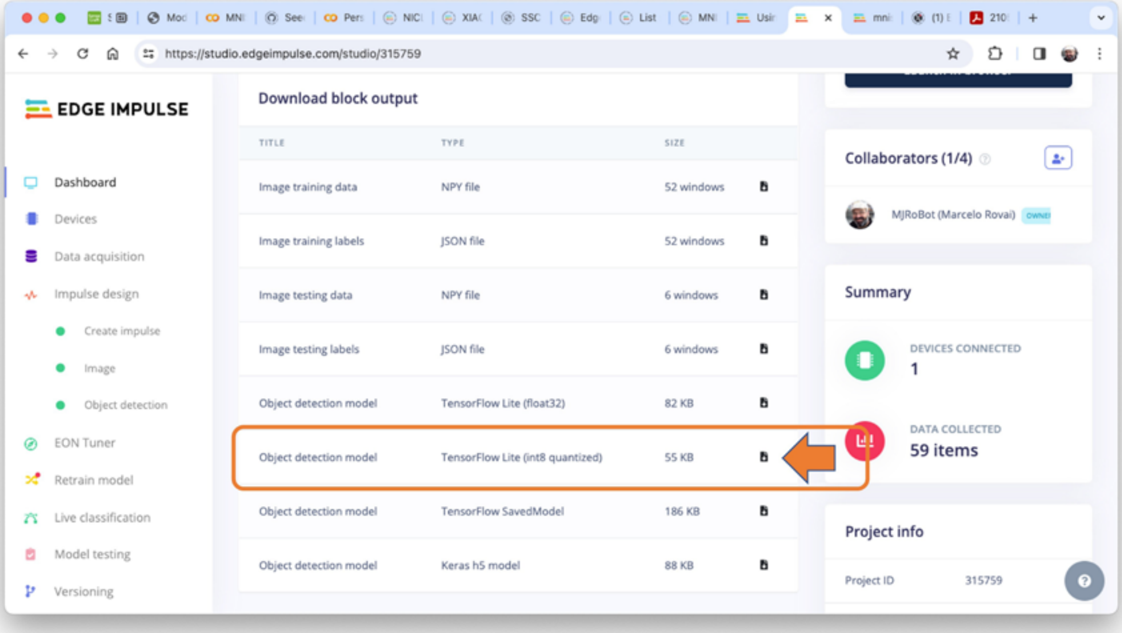
In our case, we will use the blue button at the bottom of the page: `[Upload Custom AI Model]`.

But first, we must download from Edge Impulse Studio our quantized .tflite model.

3. Go to your project at Edge Impulse Studio, or clone this one:

- [XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v1-ESP-NN]()

4. On `Dashboard`, download the model ("block output"): `Object Detection model - TensorFlow Lite (int8 quantized)`



https://studio.edgeimpulse.com/studio/315759

Download block output

TITLE	TYPE	SIZE
Image training data	NPY file	52 windows
Image training labels	JSON file	52 windows
Image testing data	NPY file	6 windows
Image testing labels	JSON file	6 windows
Object detection model	TensorFlow Lite (float32)	82 KB
Object detection model	TensorFlow Lite (int8 quantized)	55 KB
Object detection model	TensorFlow SavedModel	186 KB
Object detection model	Keras h5 model	88 KB

Collaborators (1/4)

MJRoBot (Marcelo Rovai)

Summary

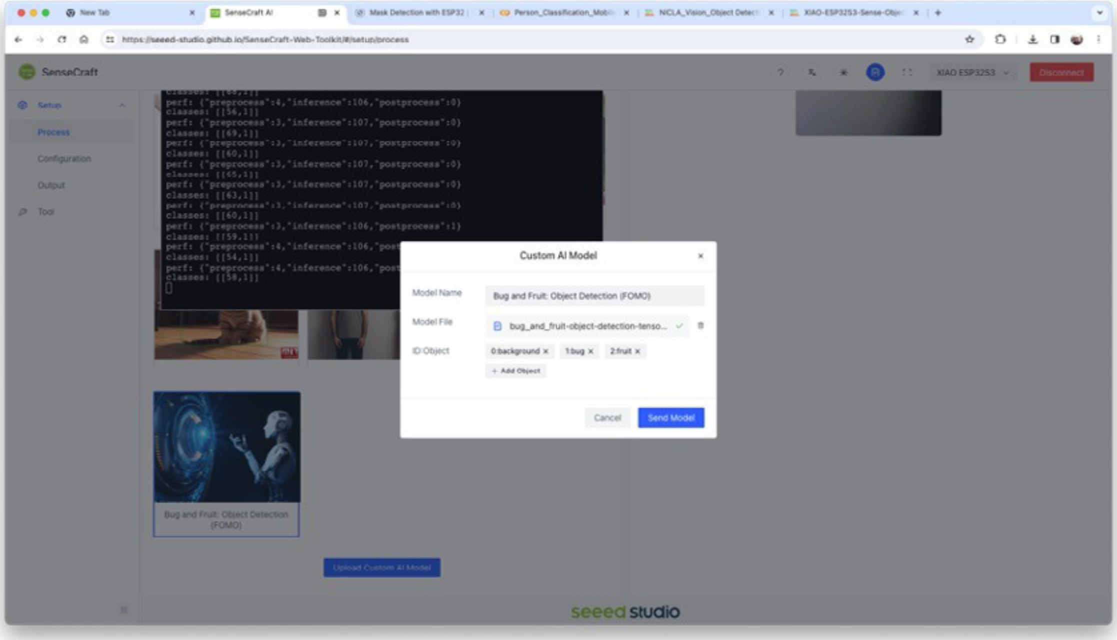
DEVICES CONNECTED 1

DATA COLLECTED 59 items

Project info

Project ID 315759

5. On SenseCraft-Web-Toolkit, use the blue button at the bottom of the page: `[Upload Custom AI Model]`. A window will pop up. Enter the Model file that you downloaded to your computer from Edge Impulse Studio, choose a Model Name, and enter with labels (ID: Object):



https://seeed-studio.github.io/SenseCraft-Web-Toolkit/#/setup/process

```
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[56,1]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[69,1]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[60,1]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[65,1]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[63,1]]
perf: {"preprocess":3,"inference":107,"postprocess":0}
classes: [[60,1]]
perf: {"preprocess":3,"inference":106,"postprocess":1}
classes: [[59,1]]
perf: {"preprocess":4,"inference":106,"post
classes: [[54,1]]
perf: {"preprocess":4,"inference":106,"post
classes: [[58,1]]
```

Custom AI Model

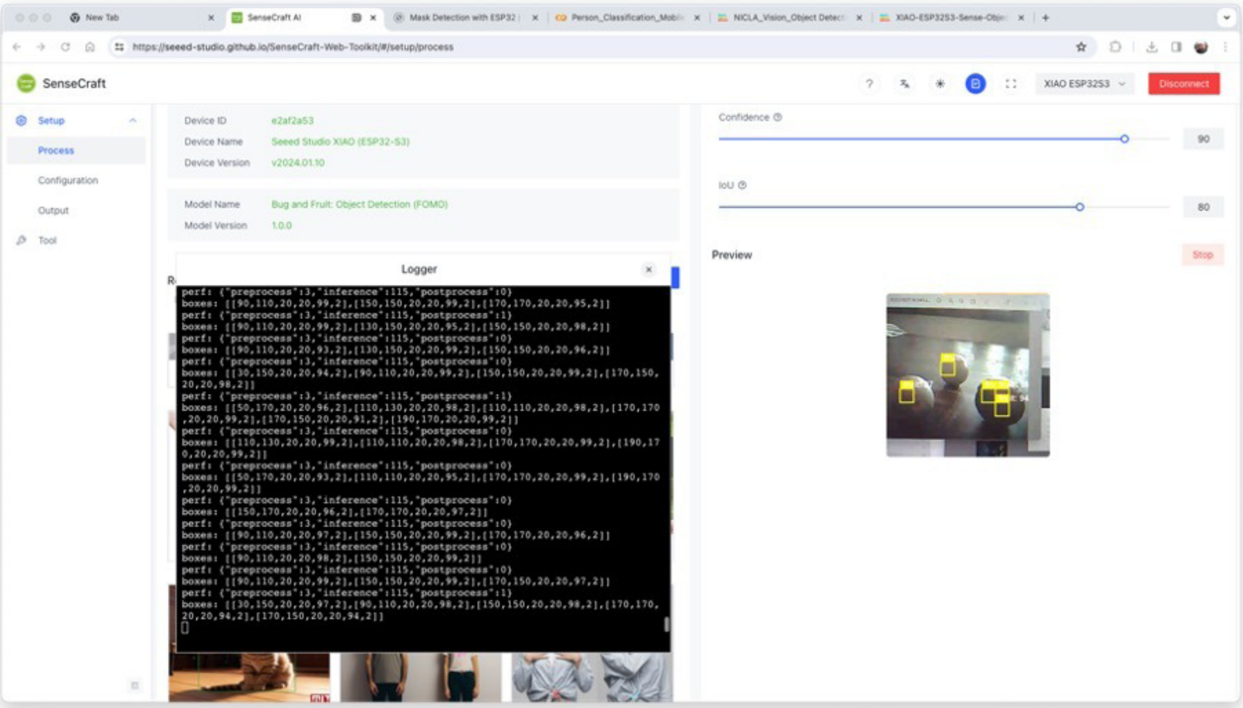
Model Name: Bug and Fruit: Object Detection (FOMO)

Model File: bug_and_fruit-object-detection-tenso...

ID:Object: 0:background 1:bug 2:fruit

Note that you should use the labels trained on EI Studio, entering them in alphabetic order (in our case: background, bug, fruit).

After a few seconds (or minutes), the model will be uploaded to your device, and the camera image will appear in real-time on the Preview Sector:



https://seeed-studio.github.io/SenseCraft-Web-Toolkit/#/setup/process

Device ID e2af2a53
Device Name Seeed Studio XIAO (ESP32-S3)
Device Version v2024.01.10

Model Name Bug and Fruit: Object Detection (FOMO)
Model Version 1.0.0

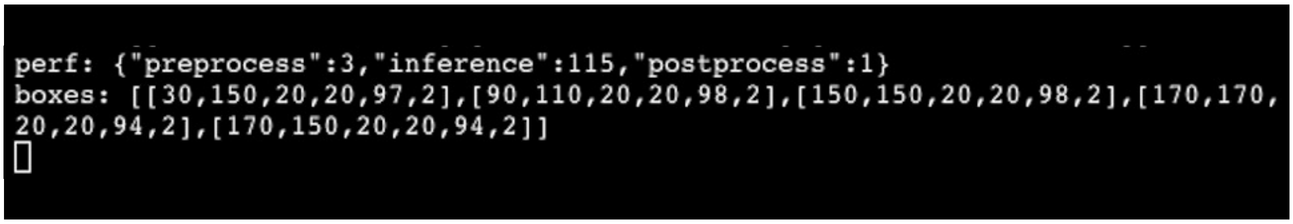
Confidence 90

IoU 80

```
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[90,110,20,20,99,2],[150,150,20,20,99,2],[170,170,20,20,95,2]]
perf: {"preprocess":3,"inference":115,"postprocess":1}
boxes: [[90,110,20,20,99,2],[130,150,20,20,95,2],[150,150,20,20,98,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[90,110,20,20,93,2],[130,150,20,20,99,2],[150,150,20,20,96,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[30,150,20,20,94,2],[90,110,20,20,99,2],[150,150,20,20,99,2],[170,150,
20,20,98,2]]
perf: {"preprocess":3,"inference":115,"postprocess":1}
boxes: [[50,170,20,20,96,2],[110,130,20,20,98,2],[110,110,20,20,98,2],[170,170
,20,20,99,2],[170,150,20,20,91,2],[190,170,20,20,99,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[110,130,20,20,99,2],[110,110,20,20,98,2],[170,170,20,20,99,2],[190,17
0,20,20,99,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[50,170,20,20,93,2],[110,110,20,20,95,2],[170,170,20,20,99,2],[190,170
,20,20,99,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[150,170,20,20,96,2],[170,170,20,20,97,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[90,110,20,20,97,2],[150,150,20,20,99,2],[170,170,20,20,96,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[90,110,20,20,98,2],[150,150,20,20,99,2]]
perf: {"preprocess":3,"inference":115,"postprocess":0}
boxes: [[90,110,20,20,99,2],[150,150,20,20,99,2],[170,150,20,20,97,2]]
perf: {"preprocess":3,"inference":115,"postprocess":1}
boxes: [[30,150,20,20,97,2],[90,110,20,20,98,2],[150,150,20,20,98,2],[170,170,
20,20,94,2],[170,150,20,20,94,2]]
```

The detected objects will be marked (the centroid). You can select the Confidence of your inference cursor Confidence. and IoU, which is used to assess the accuracy of predicted bounding boxes compared to truth bounding boxes

Clicking on the top button (Device Log), you can open a Serial Monitor to follow the inference, as we did with the Arduino IDE.



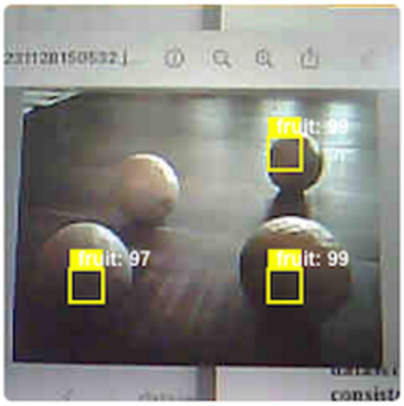
```
perf: {"preprocess":3,"inference":115,"postprocess":1}
boxes: [[30,150,20,20,97,2],[90,110,20,20,98,2],[150,150,20,20,98,2],[170,170,
20,20,94,2],[170,150,20,20,94,2]]
```

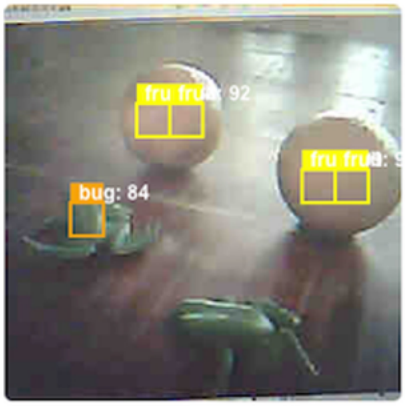
On Device Log, you will get information as:

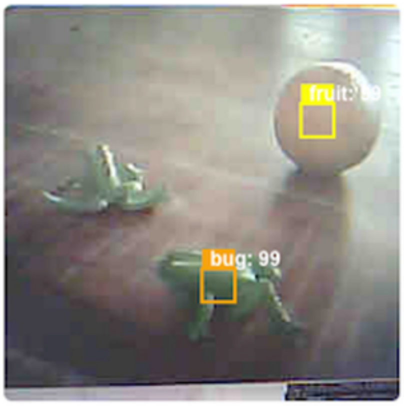
- Preprocess time (image capture and Crop): 3 ms;
- Inference time (model latency): 115 ms,
- Postprocess time (display of the image and marking objects): 1 ms.
- Output tensor (boxes), for example, one of the boxes: [[30,150, 20, 20,97, 2]]; where 30,150, 20, 20 are the coordinates of the box (around the centroid); 97 is the inference result, and 2 is the class (in this case 2: fruit)

Note that in the above example, we got 5 boxes because none of the fruits got 3 centroids. One solution will be post-processing, where we can aggregate close centroids in one.

Here are other screen shots:







4.5.9 Conclusion

FOMO is a significant leap in the image processing space, as Louis Moreau and Mat Kelcey put it during its launch in 2022:

> *FOMO is a ground-breaking algorithm that brings real-time object detection, tracking, and counting to microcontrollers for the first time.*

Multiple possibilities exist for exploring object detection (and, more precisely, counting them) on embedded devices.

4.6
To learn more

This section contains links to courses, books, and projects to learn more about Machine Learning and TinyML applications.

Online Courses

Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning

Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard

Introduction to Embedded Machine Learning - Coursera/Edge Impulse

Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse

UNIFEI-IESTI01 TinyML: “Machine Learning for Embedding Devices”

Books

“Python for Data Analysis by Wes McKinney”

“Deep Learning with Python” by François Chollet - GitHub Notebooks

“TinyML” by Pete Warden, Daniel Situnayake

“TinyML Cookbook” by Gian Marco Iodice

“Technical Strategy for AI Engineers, In the Era of Deep Learning” by Andrew Ng

“AI at the Edge” book by Daniel Situnayake, Jenny Plunkett

“MACHINE LEARNING SYSTEMS for TinyML” Collaborative effort

Projects Repositories

Edge Impulse Expert Network

MRovai XIAO ESP32S3 Movement/Sound/Image

Chapter 5:
Creative Experiments

Since its launch, the Seeed Studio XIAO series has been widely acclaimed for its compact size, powerful performance, and versatile product range. The maker community has produced a large number of projects created with XIAO. Due to space constraints, we have selected some outstanding projects made with XIAO by our makers. These projects fully demonstrate the powerful functions and wide applications of XIAO. Let us follow the makers' steps, stimulate creativity, and explore the endless possibilities of XIAO. We hope you can draw inspiration from these projects, use your imagination, and explore new territories with XIAO.

5.1
Creative and useful XIAO projects

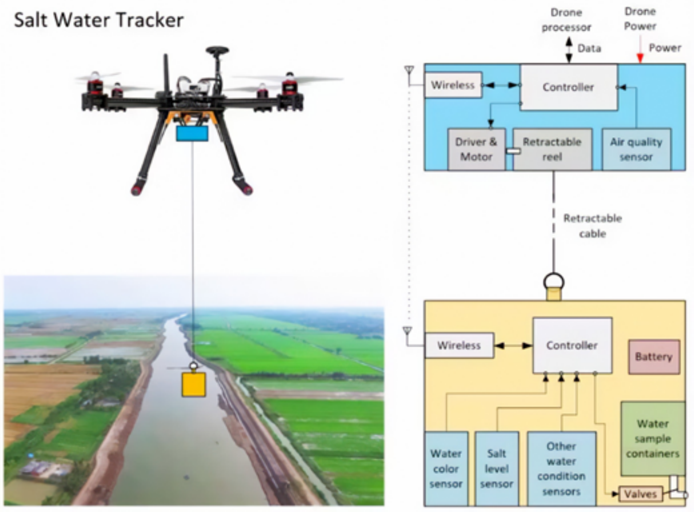
After going through the sections in this book, you might have many novel ideas that you can't wait to implement. But before you rush into it, let's take a look at what interesting stuff others have done with XIAO. For this, we have collected some user-made project cases using XIAO primarily from the globally renowned innovators' communities like hackster.io and instructables, to help you see more possibilities of XIAO.

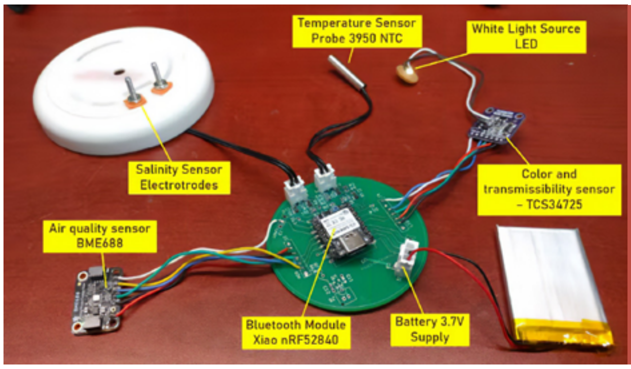
5.1.1 Drone-borne Salt Water Tracker (SWT)

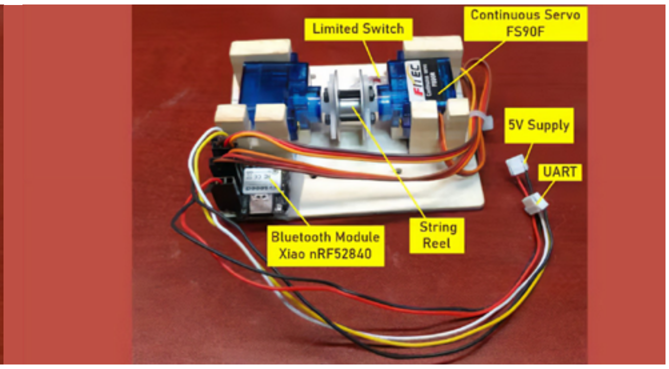
https://www.hackster.io/txnghia/salt-water-tracker-swt-cb68be

Author: Nghia Tran

The 'Salt Water Tracker' project, using the XIAO BLE nRF52840 Sense, addresses the problem of seawater erosion in rice fields in areas like the Mekong Delta in Vietnam. The project integrates a saltwater sensor system onto a Hovergames drone, turning the drone into an efficient saltwater tracking tool. This project helps farmers monitor the salinity of rivers and large water networks in real-time to ensure water safety and guide the allocation of reservoir water. The system also features temperature, water quality, air quality sensors, and a camera function for taking pictures or videos of the water area and assisting in determining water type and conditions.




Temperature Sensor Probe 3950 NTC
White Light Source LED
Salinity Sensor Electrotrodes
Color and transmissibility sensor – TCS34725
Air quality sensor BME688
Bluetooth Module Xiao nRF52840
Battery 3.7V Supply



5.1.2 SAJAC Project: Intelligent Jacket for Caving Adventure

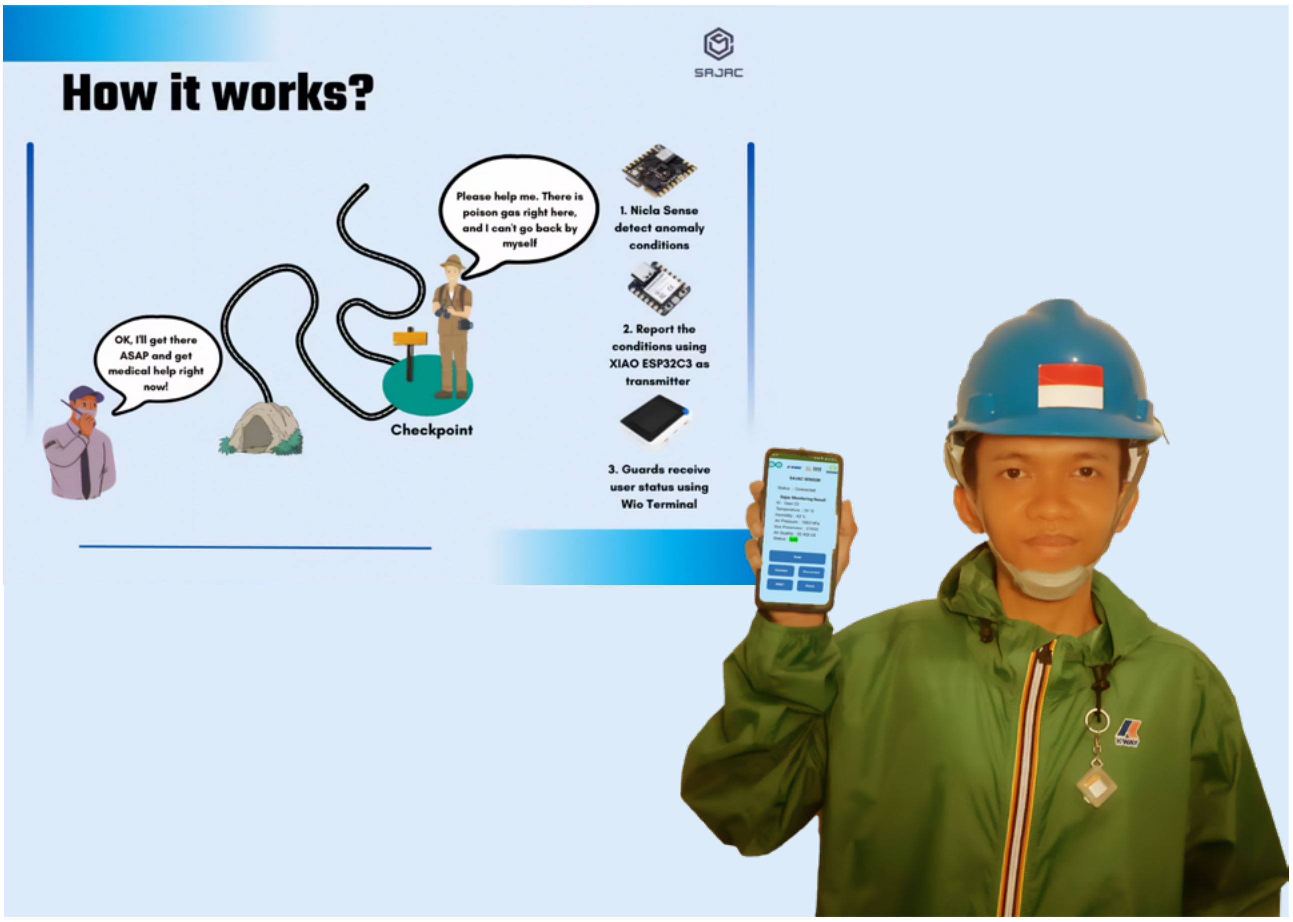
https://www.hackster.io/rifqiabdillah/sajac-project-smart-jacket-for-caving-0e6a19

Author: Rifqi Abdillah

Caving has become increasingly popular in recent years. However, cavers may face a variety of safety hazards, including extreme temperatures, damp air, low air pressure, poor air quality, and toxic gases. To address this, we developed the SAJAC project, an intelligent monitoring system designed to observe environmental conditions within a cave. The system uses Nicla Sense ME to measure the environmental quality around the user and sends the results to the SAJAC app on the user's smartphone. If the cave conditions are not suitable for exploration, Nicla Sense ME or the user's smartphone will receive notification reminders. Meanwhile, at each checkpoint within the cave, there will be a transmitter directly connected to the guard outside the cave. The user can quickly seek help via the transmitter when in danger.

Considering that there is no internet connection in the cave, we use a LoRa communication system based on XIAO ESP32C3 to transmit checkpoint data. When the user reaches the checkpoint, they just need to connect to the transmitter and press the "send" button. If the user encounters a situation where they cannot continue the exploration, they can decide whether to return on their own or wait for the guard's response.

The main post guard will use LoRa to receive data transmitted from within the cave. There is a Wio Terminal equipped with Grove Wio E5 at the outpost to receive data from transmitters inside the cave. The Wio Terminal only needs a 5-volt power supply, suitable for places with limited power.


How it works?
SAJAC
Please help me. There is poison gas right here, and I can't go back by myself
OK, I'll get there ASAP and get medical help right now!
Checkpoint
1. Nicla Sense detect anomaly conditions
2. Report the conditions using XIAO ESP32C3 as transmitter
3. Guards receive user status using Wio Terminal

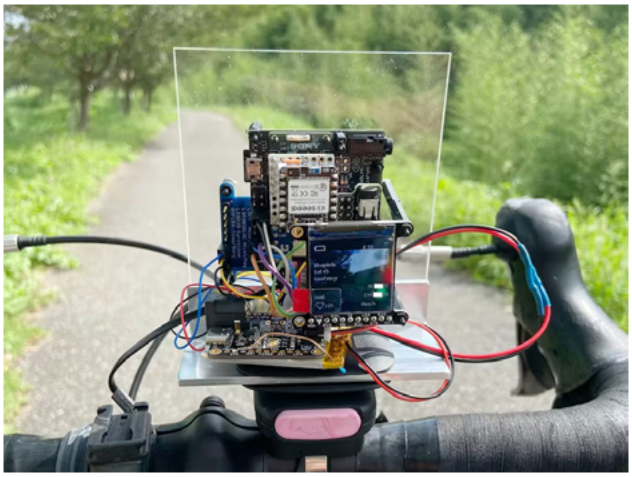
5.1.3 Bicycle Computer on Spresense

https://www.hackster.io/jens6151/bicycle-computer-on-spresense-b0e332

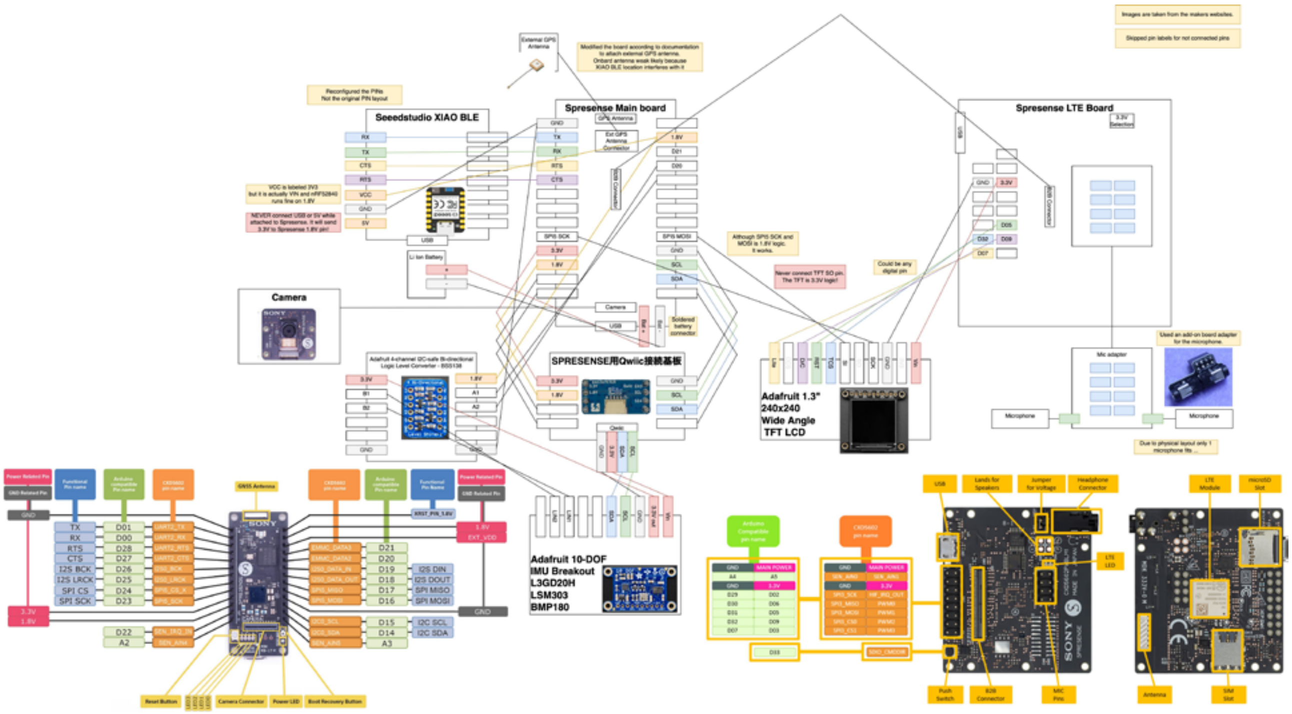
Author: Jens

The goal of this project is to build a bike computer using the Sony Spresense main board, LTE expansion board, XIAO, and other peripherals. The main features include:

1. Capture a low-resolution video stream and display it on a monitor. Option to take high-resolution photos and store them on an SD card.
2. Capture mono audio, using the OPUS codec and OGG container format for high compression, to be sent or recorded to SD card via an LTE-M connection.
3. Track location via GNSS, combining the location with weather data and points of interest (POI) data received from cloud services via an LTE connection.
4. Connect bike sensors (currently heart rate) via Bluetooth Low Energy, display data on the monitor, and record.
5. Remote access to the camera, real-time audio stream, and various data (including location) via MQTT.
6. Theft detection and notification via GNSS geofencing, accelerometer, and monitoring for nearby smartphones.



This project by Jens demonstrates the astonishing complexity of a hardcore prototype project, as can be seen from the schematic on the right.



Seeedstudio XIAO BLE
Spresense Main board
Spresense LTE Board
Camera
SPRESENSE用Qwiic接続基板
Adafruit 1.3" 240x240 Wide Angle TFT LCD
Adafruit 10-DOF IMU Breakout L3GD20H LSM303 BMP180
Images are taken from the makers websites.
Skipped pin labels for not connected pins

5.1.4 IoT AI-driven Yogurt Processing & Texture Prediction W/ Blynk

https://www.instructables.com/IoT-AI-driven-Yogurt-Processing-Texture-Prediction/

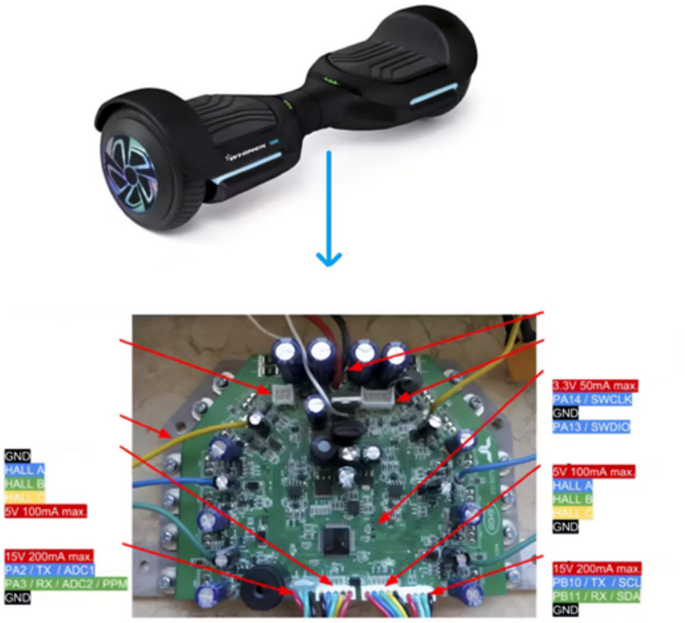
Author: Kutluhan Aktar

The aim of this project is to provide texture prediction for yogurt processing using IoT technology and AI. By using the XIAO ESP32C3 development board, along with a temperature and humidity sensor, integrated pressure sensor kit, I2C weight sensor kit, and DS18B20 waterproof temperature sensor, the project creator built an artificial neural network model and trained it with Edge Impulse to predict yogurt texture without the addition of chemical additives. Users can remotely view sensor readings and control devices through the Blynk app. Finally, the author designed a durable enclosure suitable for a dairy environment. This project has the potential to help dairy product manufacturers reduce costs and improve product quality.

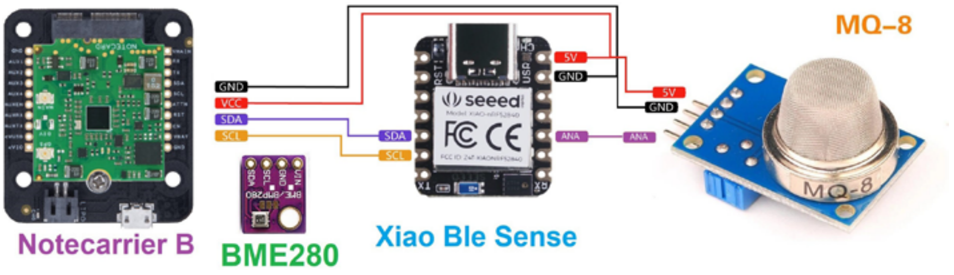
5.1.5 Web browser operated robot for gas leak detection

https://www.hackster.io/ivan-arakistain/web-browser-operated-robot-for-gas-leak-detection-4cbe1b

Author: Ivan Arakistain



This project repurposes an old hoverboard into a remote-controlled robot equipped with a hydrogen sensor for early detection of hydrogen leaks. It uses Bluetooth to connect the Seedstudio Xiao Ble Sense, MQ-8 gas sensor, and other devices, and uses Edge Impulse Studio to train a machine learning model. The robot also uses the Blues Wireless Notecard NBGL cellular connection technology to upload data to the cloud. With Remo.TV, it can be remotely operated to drive the robot and view real-time camera feeds through a browser.

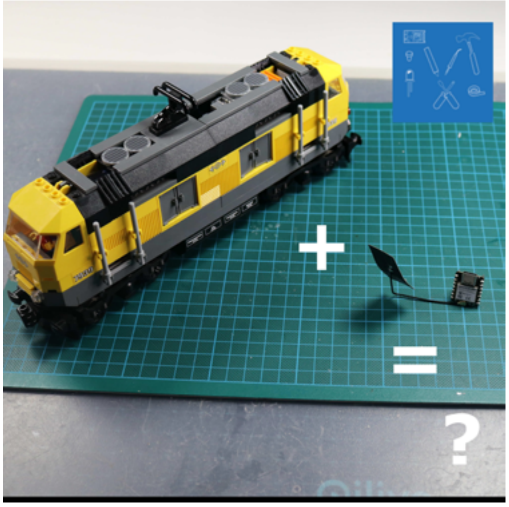


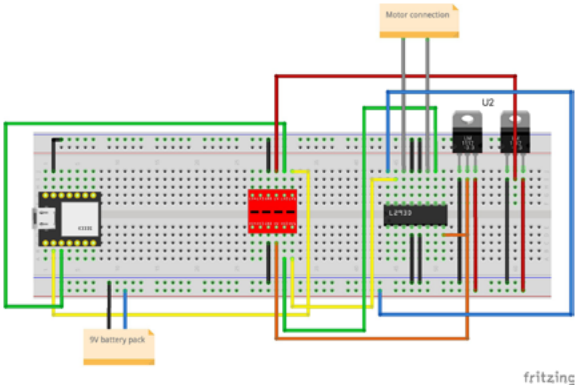
5.1.6 Train Controller With Seeed Studio XIAO ESP32C3

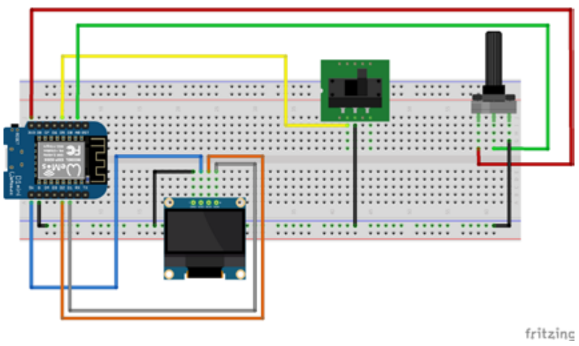
https://www.instructables.com/Train-Controller-With-Seeed-Studio-XIAO-ESP32C3/

Author: Tiago Santos

This project designs a train controller using the XIAO ESP32C3 module from Seeed Studio. The project is divided into a train part and a controller part. The train part uses the XIAO ESP32C3 module to connect to the train and controls the train motor through the L293D motor driver. The controller part uses the Wemos D1 Mini to receive speed and direction information and displays the actual speed on a 0.96-inch ssd1306 screen. The controller communicates with the train part through Wi-Fi and an MQTT server. The project simplifies the complexity of traditional Lego train remote control systems and improves control efficiency.





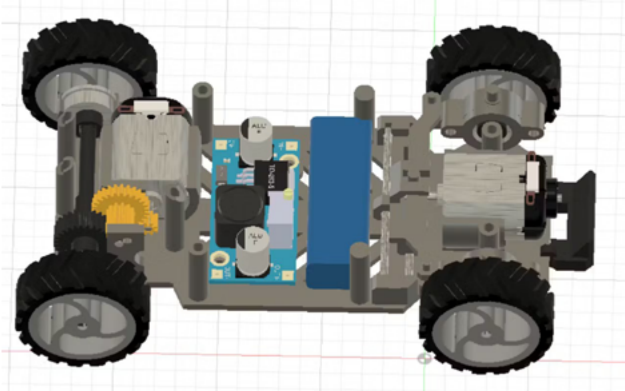


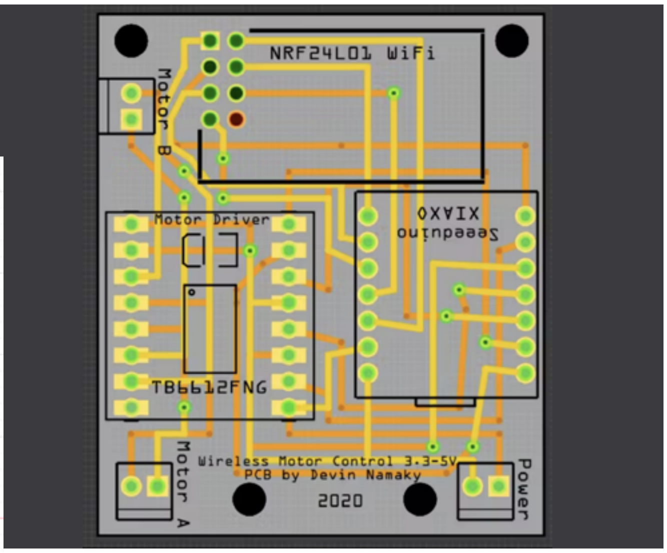
5.1.7 RC Car (Arduino-Based 3D Resin Printed) RC_Car_RP

https://www.hackster.io/devinnamaky/rc-car-arduino-based-3d-resin-printed-rc-car-rp-9b4dce

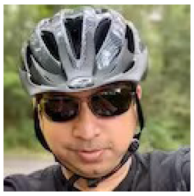
Author: Devin Namaky

This project is a 3D printed remote-controlled car based on Arduino Nano and Seeeduino XIAO, named RC_Car_RP. The project uses two standard 130 type DC motors as drive and steering, and the steering system uses gear transmission. The Seeeduino XIAO module is used to control the motor driver TB6612FNG, realizing the control of the car speed and direction. Communication between the remote control and the car is achieved through the nRF24L01 wireless module. The project is small in size, simple in design, easy to build, and can meet the remote-controlled car needs in different scenarios.



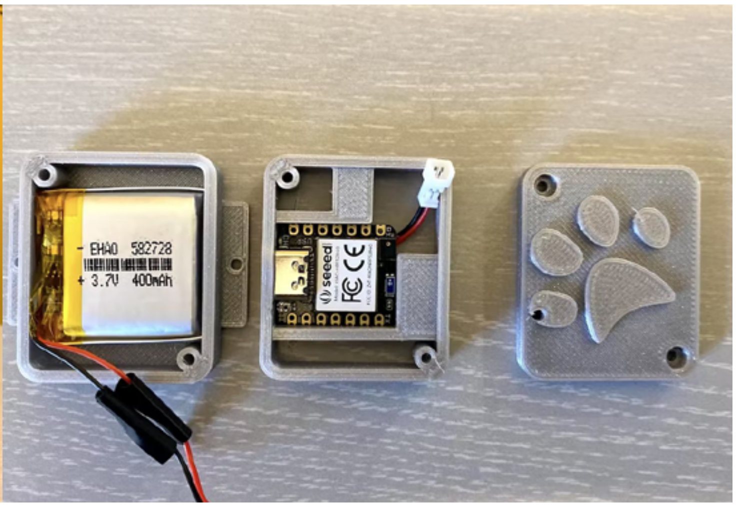


5.1.8 Pet Activity Tracker using XIAO BLE Sense & Edge Impulse



https://www.hackster.io/mithun-das/pet-activity-tracker-using-xiao-ble-sense-edge-impulse-858d73

Author: Mithun Das



This project is a wearable device that tracks pet activities using XIAO BLE Sense and Edge Impulse, aimed at helping our pets stay active. The XIAO BLE Sense is a mini controller equipped with a powerful Nordic nRF52840 MCU, built-in Bluetooth 5.0 module, and designed around a 32-bit ARM® Cortex™-M4 CPU. It features a 6-axis IMU that can be used to predict activities such as rest, walking, and running.

With the accompanying smartphone app, users can connect to the device via Bluetooth and obtain minute-by-minute prediction data. The data is stored in the smartphone's local storage and presented graphically to provide meaningful insights.

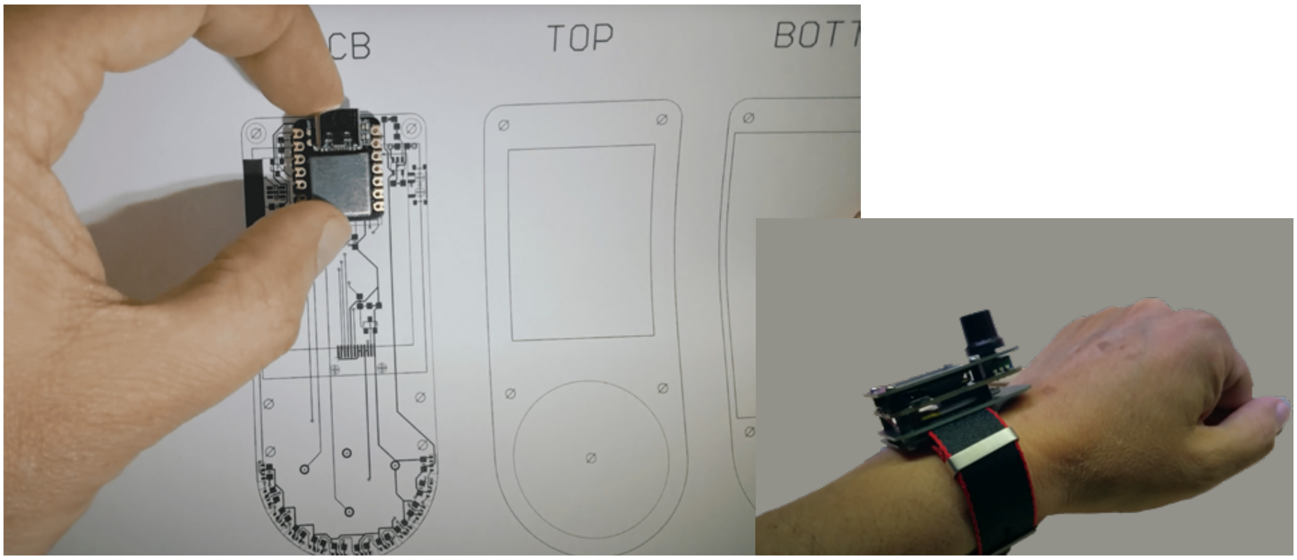
The project collects data via the EI Blue mobile app, creates machine learning models using Edge Impulse Studio, and builds an iOS app using Google Flutter. The whole system can monitor the pet's activity status in real-time and view the data through the mobile app.

5.1.9 H.E.D.S. On your wrist, New Seeeduino XIAO Board

https://www.hackster.io/ihayri1/h-e-d-s-on-your-wrist-new-seeeduino-xiao-board-7d8f74 https://youtu.be/ql2wnFtSQqQ

Author: Hayri Uygur

Hayri has made a Maker-style multifunctional wristwatch, H.E.D.S., using XIAO. It provides a set of small, handy tools with many functions and variations, and is equipped with a beautiful, sharp 240x240 pixel IPS display.



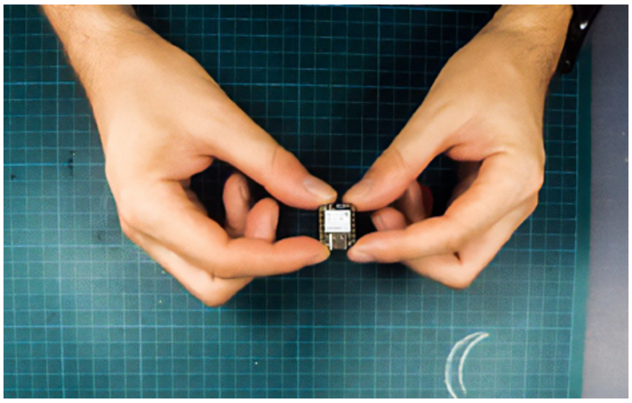
5.1.10 Hearbeat Monitor With XIAO NRF52840

https://www.instructables.com/Hearbeat-Monitor-With-XIAO-NRF52840/

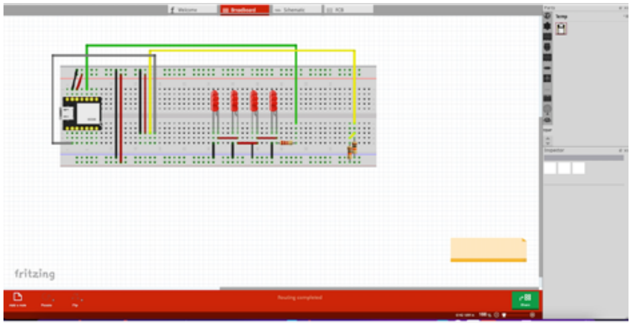
Author: TiagoSantos

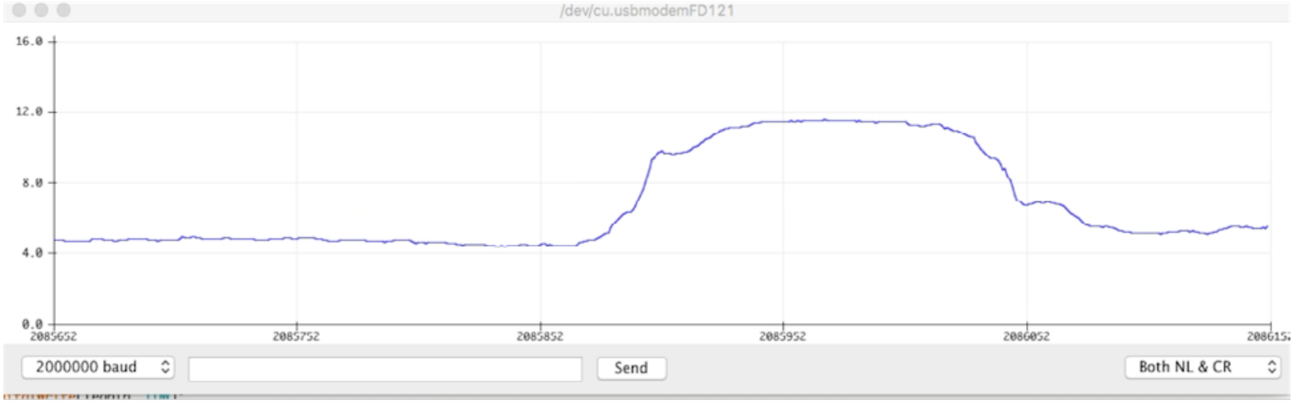
This project uses a XIAO NRF52840 microcontroller, based on the Nordic nRF52840 CPU, to make a heartbeat monitor. This microcontroller supports Bluetooth 5.0 and NFC and has a super small size, making it ideal for wearable devices and other projects with limited space. The project uses another biomedical microcontroller called Bitalino to monitor the heartbeat. The XIAO NRF52840 receives information from the ECG (Electrocardiogram) sensor and then transmits it to a set of LEDs. Through this project, we can view the heart rate in real-time and observe the data of heart activity.

1. Prepare the Bluetooth version of XIAO nRF52840. Its small size is very suitable for wearable devices.
2. Bitalino is a biomedical kit similar to Arduino developed by Hugo Silva in Portugal. This project will use some modules from it.

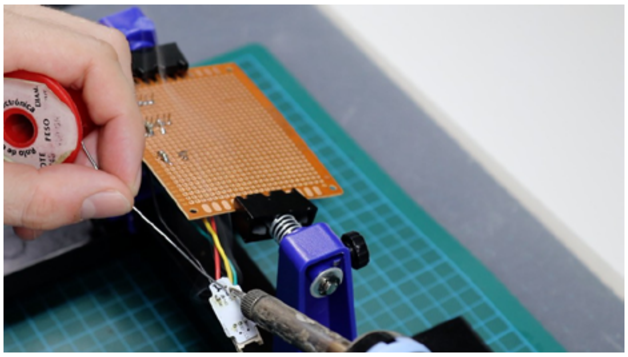


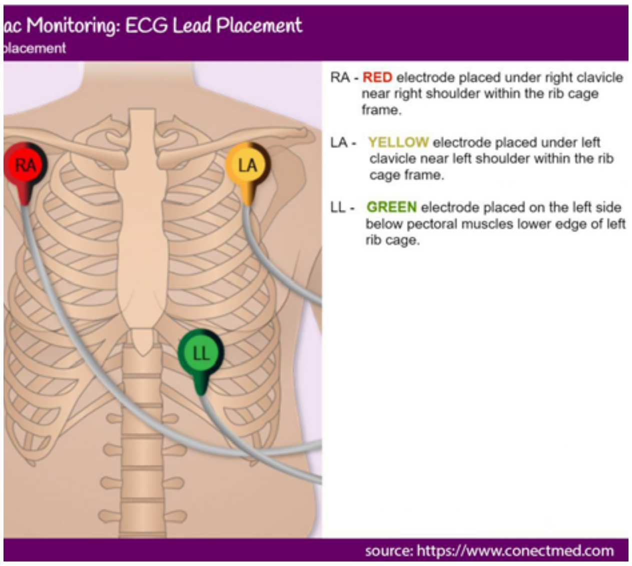
3. Circuit diagram: XIAO receives heart rate information from the ECG sensor, converts it, and sends it. The LED flashes with the heart rate, and the Arduino serial port plotter displays the graphical information of the heart rate.





4. Use a perforated board to place components and solder. First, place resistors and the female pins of XIAO, then solder the ECG sensor. Finally, cut the perforated board to the required size.





ac Monitoring: ECG Lead Placement

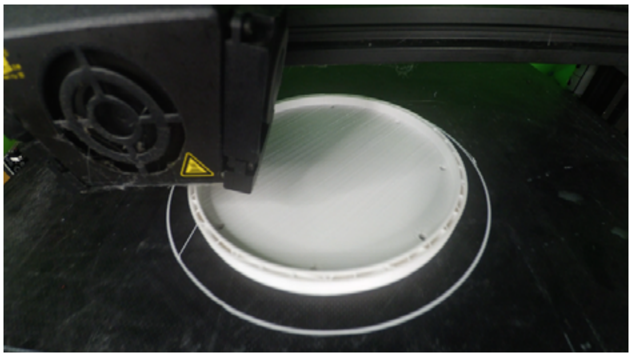
placement

RA - RED electrode placed under right clavicle near right shoulder within the rib cage frame.

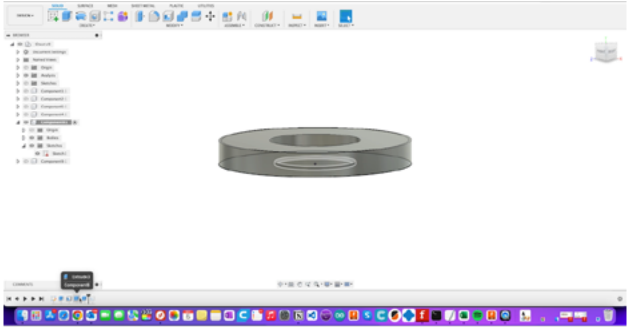
LA - YELLOW electrode placed under left clavicle near left shoulder within the rib cage frame.

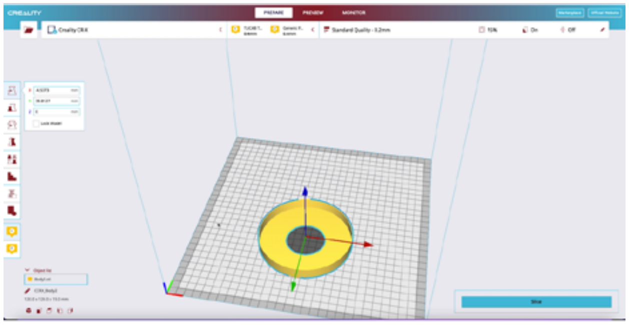
LL - GREEN electrode placed on the left side below pectoral muscles lower edge of left rib cage.

source: https://www.conectmed.com

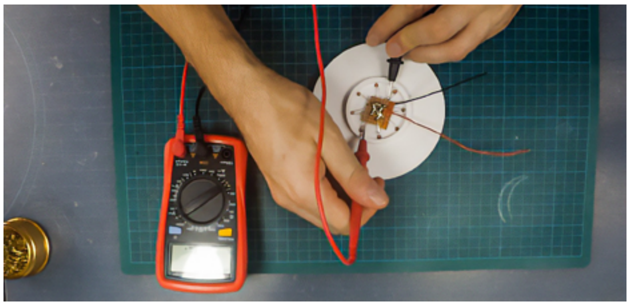


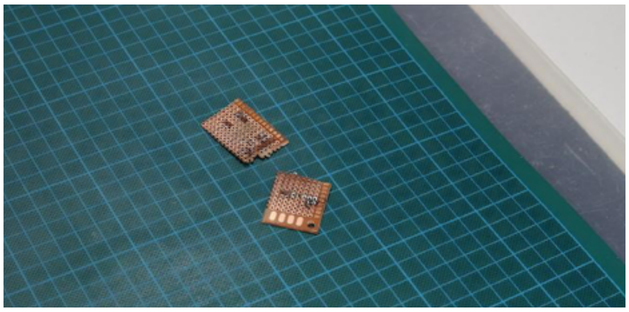
5. Use Fusion 360 to design the LED shell, the main shell, and the structure of the chest part. Use Creality Slicer to transcode and send it to the 3D printer to get structural parts.





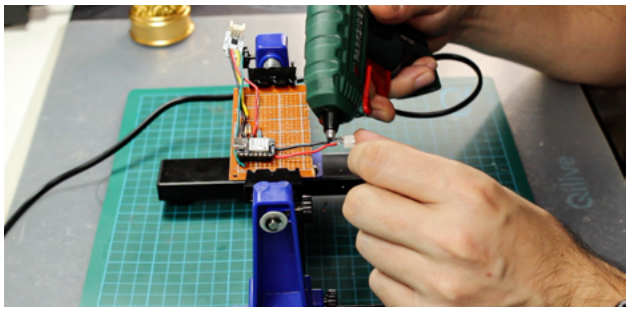
6. When connecting the LED, use a perforated board to connect all cathodes and place a ground connector. After all connections are completed, it is necessary to check whether VCC is isolated from the ground and perform a test.
7. Not everything can go as expected. During the connection check, the fixture exerted too much force, causing the perforated board to break. It had to be redone.

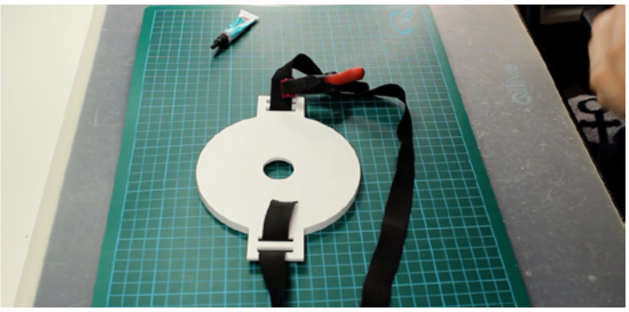




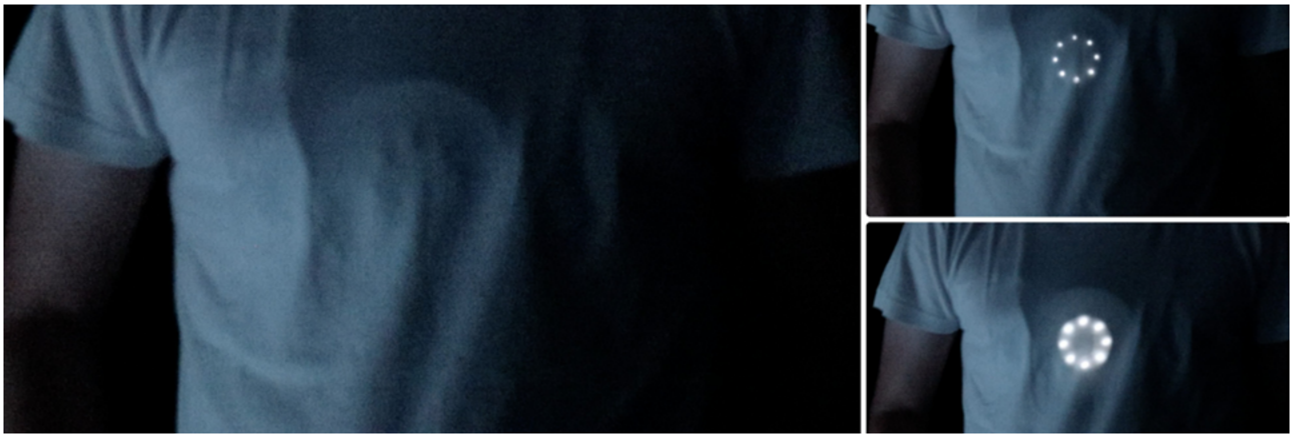
8. Finally, it's time to connect the battery and isolate all circuits to avoid short circuits. Usually, heat-shrink tubing would be used here, but if there is no suitable size, hot glue can also work.

9. Place all components on the 3D printed parts and perform a test, then use super glue to connect the parts. The part fixed on the chest was pasted with an elastic band. Finally, replace the LED and remove the resistor to get more noticeable light effects.





10. The final effect.

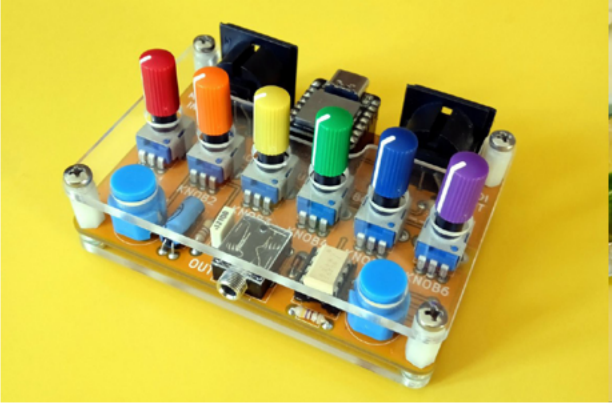


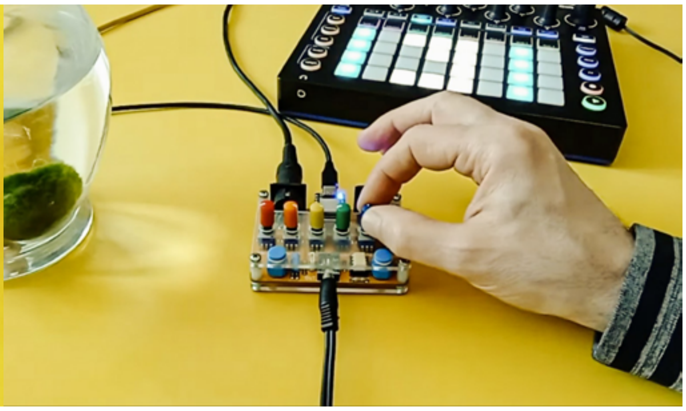
5.1.11 Multi MIDI Controller, Filter, Router & Sound Generator

https://www.synthtopia.com/content/2022/03/29/multi-midi-controller-filter-router-sound-generator/ https://github.com/pangrus/multi

Author: Pangrus

Multi is a multifunctional MIDI controller, primarily used for audio synthesis, with a very small size. Compared with the latest generation of commercial controllers, it has a USB port and two DIN interfaces. The Multi controller is fully programmable, allowing for some functionalities in a computer-free setup. In addition, it can also be used as a sound generator as it is equipped with a 10-bit DAC converter, making it ideal for exploring digital synthesis technology. The Multi controller is powered by the robust Seeeduino XIAO, featuring 6 knobs, 2 buttons, 2 Midi DIN interfaces, and a 1/8 inch audio interface. Its MIDI input has opto-isolation to avoid ground loops, complying with the official specification.



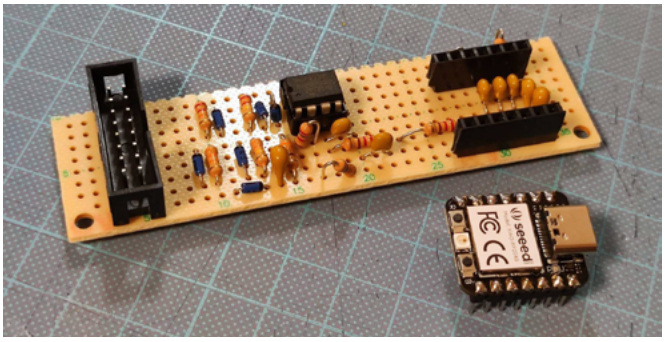


5.1.12 DIY eurorack modular synth Raspberry Pi VCO with Seeed XIAO

https://www.hackster.io/hagiwo/diy-eurorack-modular-synth-rasberry-pi-vco-with-seeed-xiao-133ac0

Author: HAGIWO/ハギヲ

A maker from Japan, HAGIWO / ハギヲ, used the Seeed XIAO RP2040 development board to create a Voltage-Controlled Oscillator (VCO) module for a Eurorack modular synthesizer. This board has a Raspberry Pi RP2040 microcontroller, 4 AD converters, and is easier to use than the Raspberry Pi Pico. The VCO module has three modes: Wavefold, FM, and AM, with eight built-in waveforms, costing only about 1100 yen.

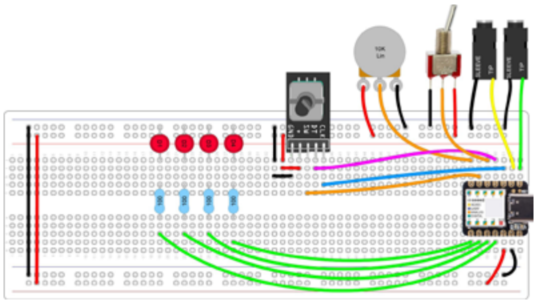


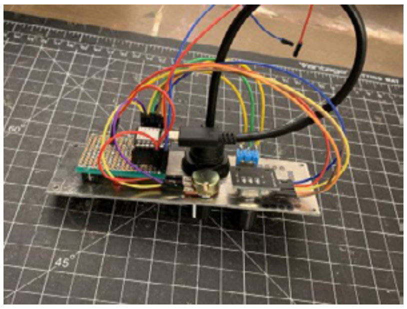
5.1.13 Xiao CV Sequencer

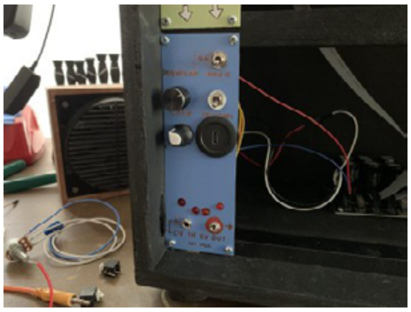
https://www.instructables.com/Xiao-CV-Sequencer/

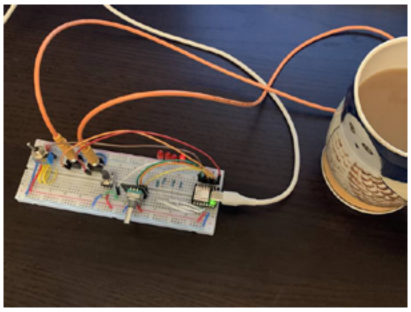
Author: analogsketchbook

Using the Seeduino Xiao microcontroller and a few parts, a decent CV synthesizer was created, mainly for modular synthesizer systems. Xiao's role in this project is to output Control Voltage (CV) signals through its analog output pins for passing note information between modules. It also controls other features such as adjusting speed, mode switching, and sequence selection.







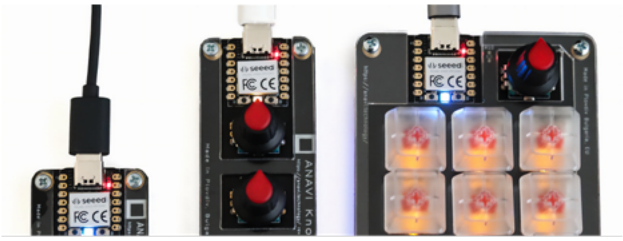


5.1.14 ANAVI Macro Pad 10 & Knobs

https://www.crowdsupply.com/anavi-technology/anavi-macro-pad-10

Author: Crowd Supply

A company has designed and manufactured three small, programmable, open-source mechanical input devices through crowdfunding: ANAVI Macro Pad 10 keyboard, ANAVI Knob 3, and ANAVI Knob 1. All are driven by the powerful Raspberry Pi RP2040 microcontroller inside Seeed XIAO RP2040, support USB Type-C, and run the KMK firmware based on CircuitPython. These customizable devices are suitable for video or audio editing, entertainment broadcasting, gaming, programming, etc., providing precise control and practical lighting effects. They are simple to use, and their plans and schematics can be found on GitHub.

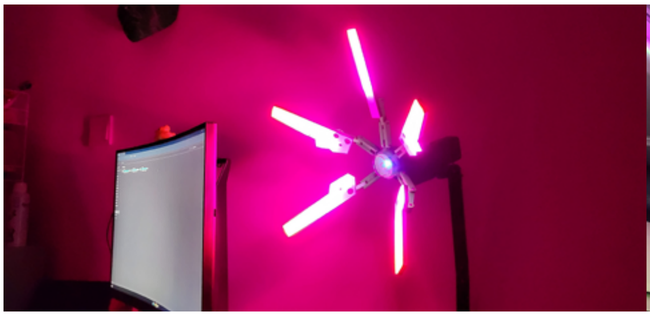


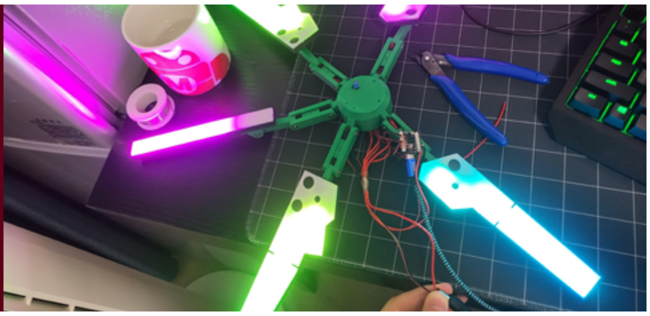
5.1.15 Death Stranding Desk Lamp

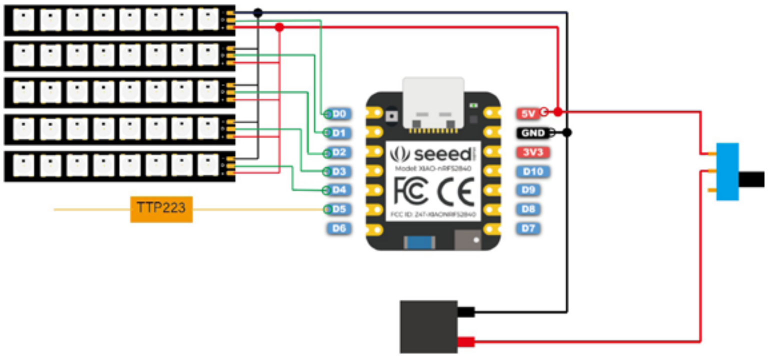
https://www.hackster.io/wyx269263336/death-stranding-desk-lamp-ae5f71

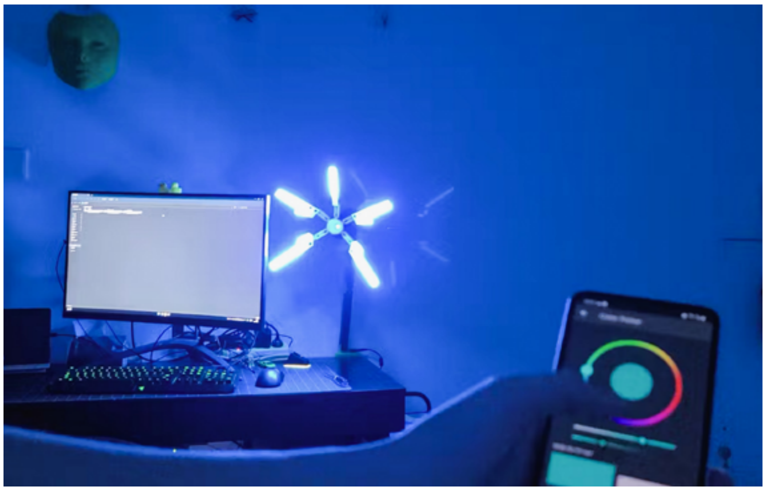
Author: Pinkman

This smart lamp, based on the multifunctional scanning device Odradek in the game Death Stranding, is made up of five separate light blades, each with three degrees of freedom, so you can adjust the desired angle at any time. It integrates the XIAO nRF52840 Sense Bluetooth main control board and WS2812 magic color light strip, and you can control its color and brightness through a mobile app.







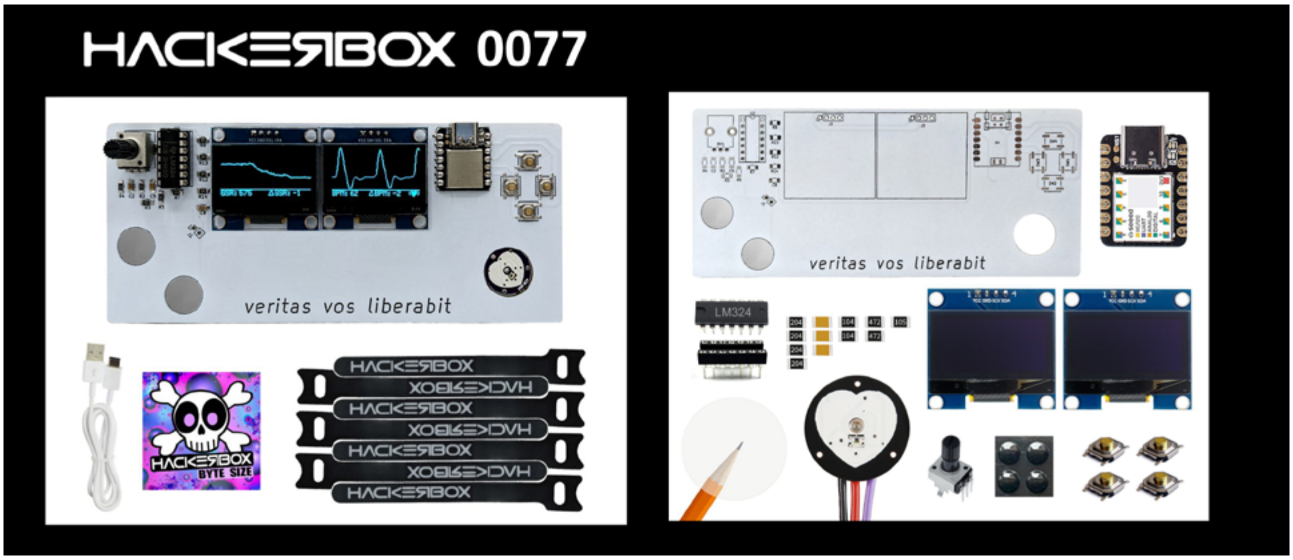


5.1.16 HackerBox 0077: Veritas

https://www.instructables.com/HackerBox-0077-Veritas/

Author: HackerBoxes

This project teaches you how to make a simple lie detector. It involves configuring the Seeeduino XIAO microcontroller module, modifying the OLED module to achieve dual display operation with a single microcontroller, assembling a Galvanic Skin Response (GSR) sensor based on an operational amplifier, and integrating a heart rate sensor. XIAO acts as the core controller in the project, realizing data collection, processing, and display.



5.1.17 DISCIPLINE - A workout timer

https://www.hackster.io/rw2493/discipline-a-workout-timer-6b5614

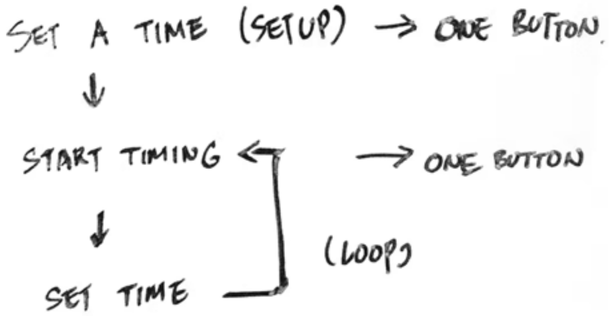
Author: Rui Wang

DISCIPLINE: This is a homemade timer that helps you strictly control rest intervals during muscle training. The project uses the Seeeduino XIAO microcontroller, along with two buttons, a display screen, a battery, and other components to achieve a simple user interface and a portable design. XIAO is responsible for the core control function of the timer in the project, providing accurate timing services to users.

The design goals include:

- Small, portable, and compact
- Complete timer functions
- Simple user interface design
- Clear interaction flow
- Cool appearance

Interaction is designed to be as simple as possible to minimize operation steps.



Yellow and blue button light interaction description: After some playtests, I then use the yellow button to control the time setup, and I use the blue button to start the counting. To provide a good indication, I did several things for the LEDs. (Y for Yellow, B for Blue) When powering it on: Y -> Fade; B -> ON, indicate to pick up a time period.

- When powering it on: Y -> Fade; B -> ON, indicate to pick up a time period.

Press Y to switch timing options: 30s, 60s, 90s, 120s.

- Press Y to switch timing options: 30s, 60s, 90s, 120s.

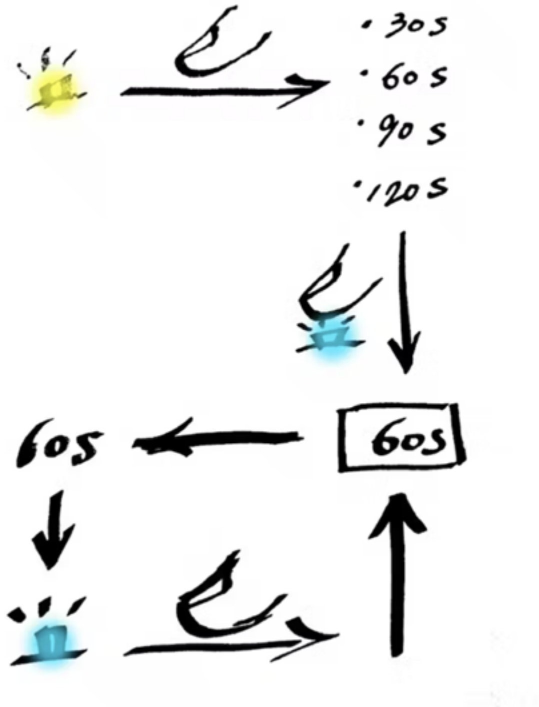
Press B to confirm your choice, the timer starts counting down. Y -> OFF; B -> OFF.

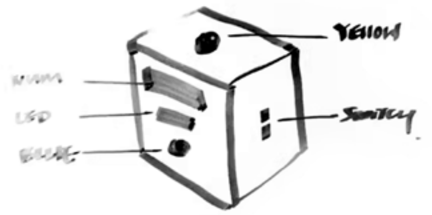
- Press B to confirm your choice, the timer starts counting down. Y -> OFF; B -> OFF.

Timer ends counting, B -> ON; Y -> OFF forever.

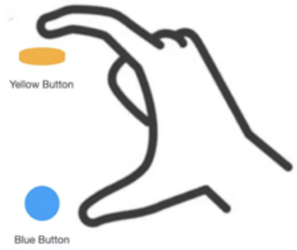
- Timer ends counting, B -> ON; Y -> OFF forever.

Two finger operation: The final design choice was to allow users to hold it easily with one hand and operate it with two fingers.

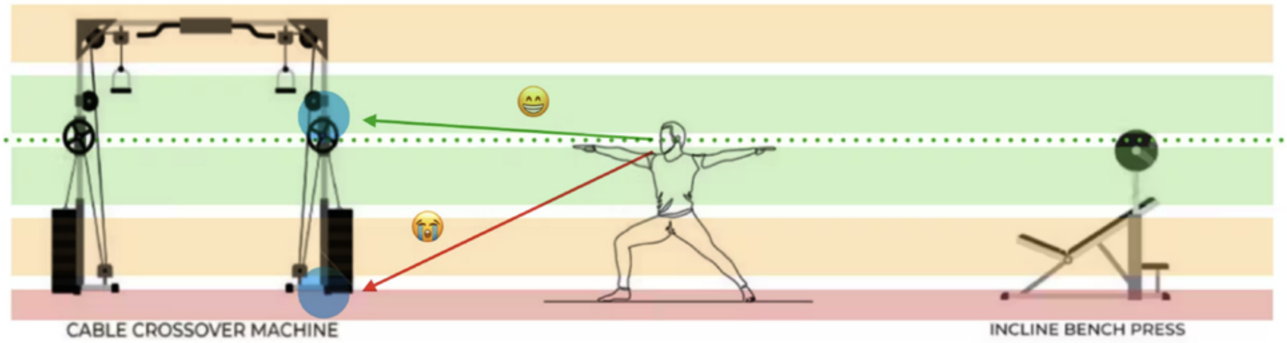


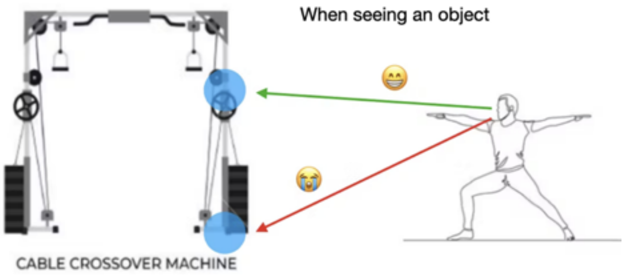


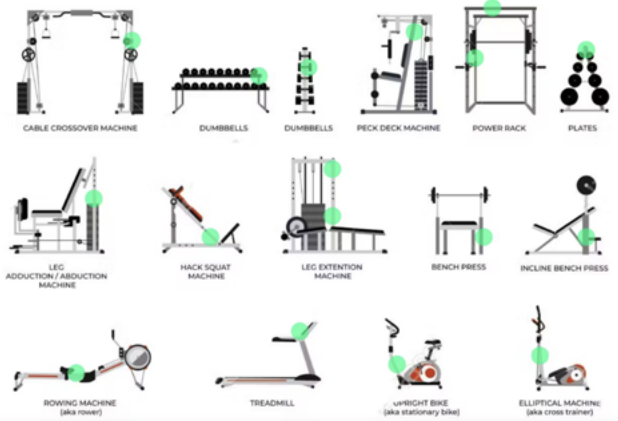
NUM
LED BAR
YELLOW BUTTON

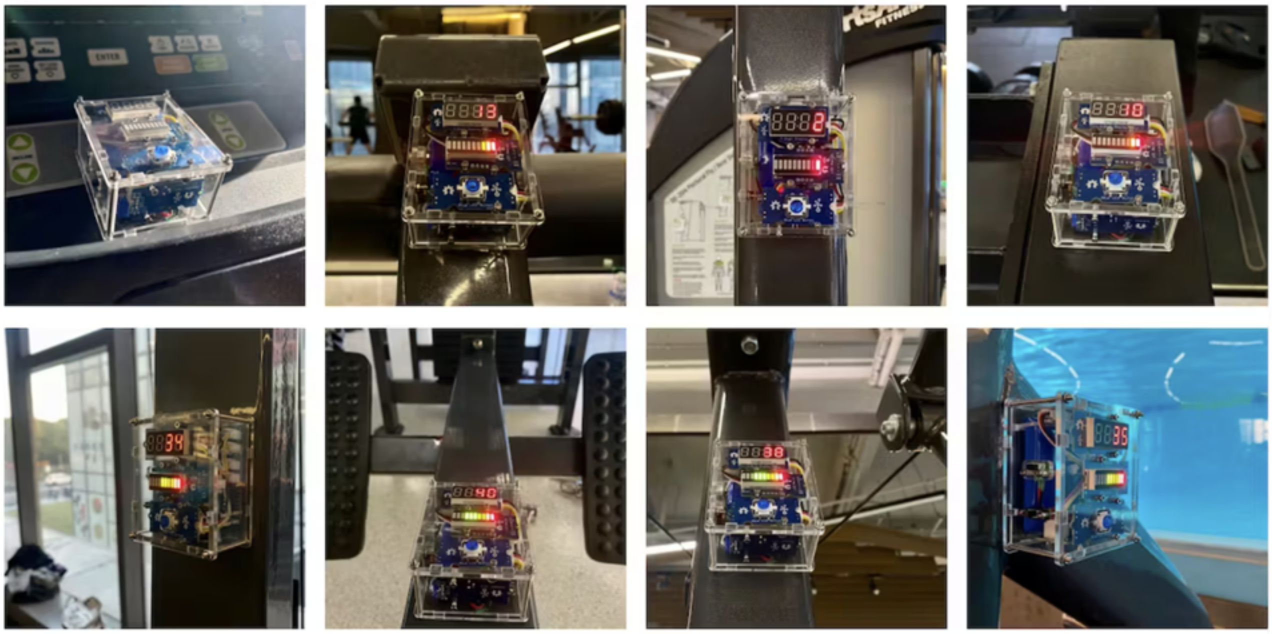


Magnetic Attachment: After analyzing pain points, it was decided to use magnets to attach the product to places where interaction and operation are more easily realized.









5.1.18 Seeed Fusion DIY XIAO Mecha

https://www.seeedstudio.com/seeed-fusion-diy-xiao-mechanical-keyboard-contest.html

XIAO 的小巧尺寸与其强悍的性能，没想到在 DIY 键盘与控制器玩家中得到认可，为此 Seeed 在 2022 年 7 月至 10 月，组织了一次 Fusion XIAO 机器键盘大赛，下面我们展示了此次比赛的一些获奖项目，以帮助对 DIY 键盘有兴趣的读者。

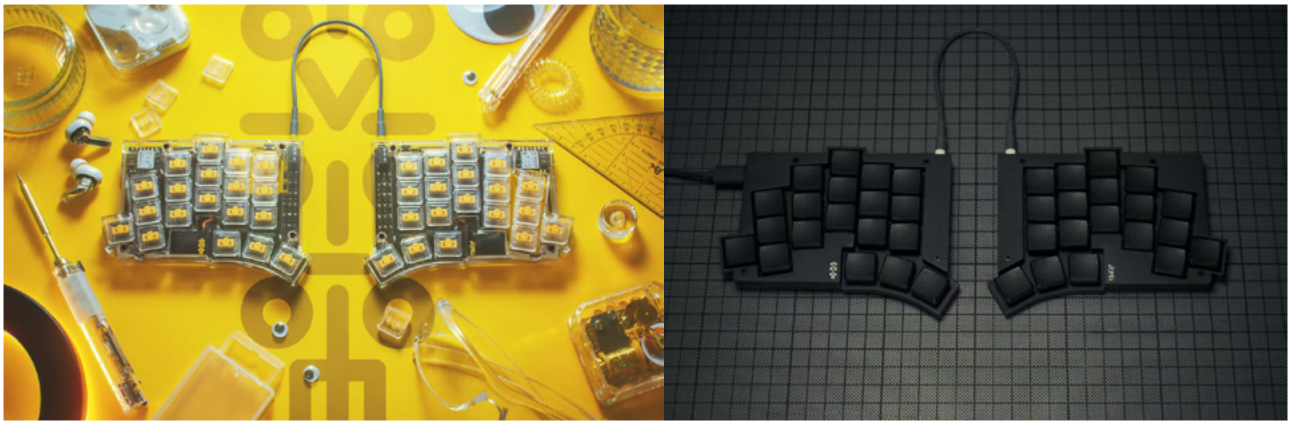
1st Prize:

TOTEM | a tiny splitkeyboard with splay

(2x)19 key ergo split: 3-key thumb cluster, pinky splay, low profile. Useful repo and classy, unique case. Nicely documented and open source. And it's a usable keyboard, which could be used as a daily driver. Other than that, Marc took a great effort to present his design aesthetically

https://www.hackster.io/geist/totem-a-tiny-splitkeyboard-with-splay-cb2e43

Author: Marc Rühl



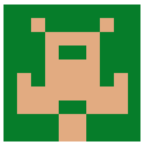
2nd Prize:

Beyblock20 | a magnetic, modular MacroPad

https://github.com/ChrisChrisLoLo/beyblock20

Author: Christian Lo

Purple Owl | a 60% keyboard powered by Seeed XIAO RP2040



https://www.hackster.io/sonalpinto/purple-owl-a-60-keyboard-powered-by-seeed-xiao-rp2040-f73604

Author: Sonal Pinto

3rd Prize:

KLEIN | a wireless ergonomical keyboard

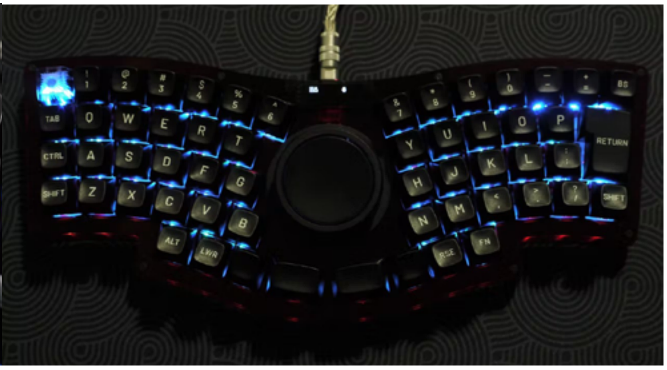
https://www.hackster.io/nosnk/klein-a-wireless-ergonomical-keyboard-b4cd9a

Author: Shashank

GRIN Quern | an ergonomic keyboard on center trackpad

https://www.hackster.io/policium/grin-quern-ergonomic-keyboard-on-center-trackpad-8b58c3

Author: policium



Kidoairaku Swallowtail | a cute butterfly-shaped keyboard

Author: yswallow

About the authors

Lei Feng is the leader of the technical support group and product curriculum at Seeed Studio. An experienced author in the fields of open-source hardware and edge computing, he has published several books in China, including “GameGo Beginner Programming Course for Arcade 《做游戏，玩编程——零基础开发微软 Arcade 掌机游戏》,” “Grove Beginner Kit For Arduino - Codecraft Graphical Programming Course 《Arduino 图形化编程轻松学》”, and the Chinese translation of “IoT for Beginners 《深入浅出 IoT：完整项目通关实战》” with support from Microsoft China.

Lei Feng has created numerous tutorials and open-source documentation in Chinese and English with his team. His hands-on experience developing IoT and edge computing projects gives him unique insights into simplifying complex concepts for beginners. As an engaging writer and patient teacher, Lei Feng is the ideal guide to make Arduino and TinyML approachable for newcomers worldwide.

LinkedIn profile: https://www.linkedin.com/in/leon-feng-a029bb1/

Marcelo Rovai is a recognized figure in engineering and technology education, holding the title of Professor Honoris Causa from the Federal University of Itajubá, Brazil. His educational background includes an Engineering degree from UNIFEI and an advanced specialization from the Polytechnic School of São Paulo University. Further enhancing his expertise, he earned an MBA from IBMEC (INSPER) and a Master’s in Data Science from the Universidad del Desarrollo in Chile.

With a career spanning several high-profile technology companies such as AVIBRAS Airspace, ATT, NCR, and IGT, where he served as Vice President for Latin America, he brings a wealth of industry experience to his academic endeavors. He is a prolific writer on electronics-related topics and shares his knowledge through open platforms like Hackster.io.

In addition to his professional pursuits, he is dedicated to educational outreach, serving as a volunteer professor at UNIFEI and engaging with the TinyML4D group as a Co-Chair, promoting TinyML education in developing countries. His work underscores a commitment to leveraging technology for societal advancement.

LinkedIn profile: https://www.linkedin.com/in/marcelo-jose-rovai-brazil-chile/