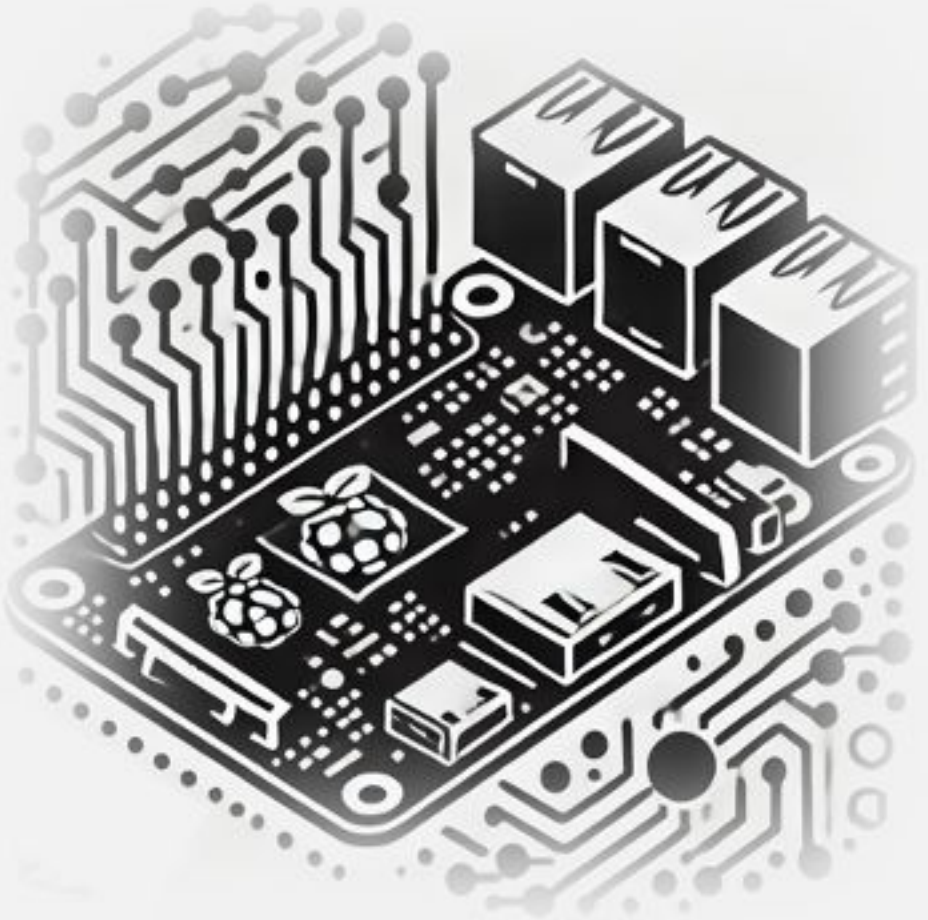
IESTI05 – Edge AI

Machine Learning System Engineering



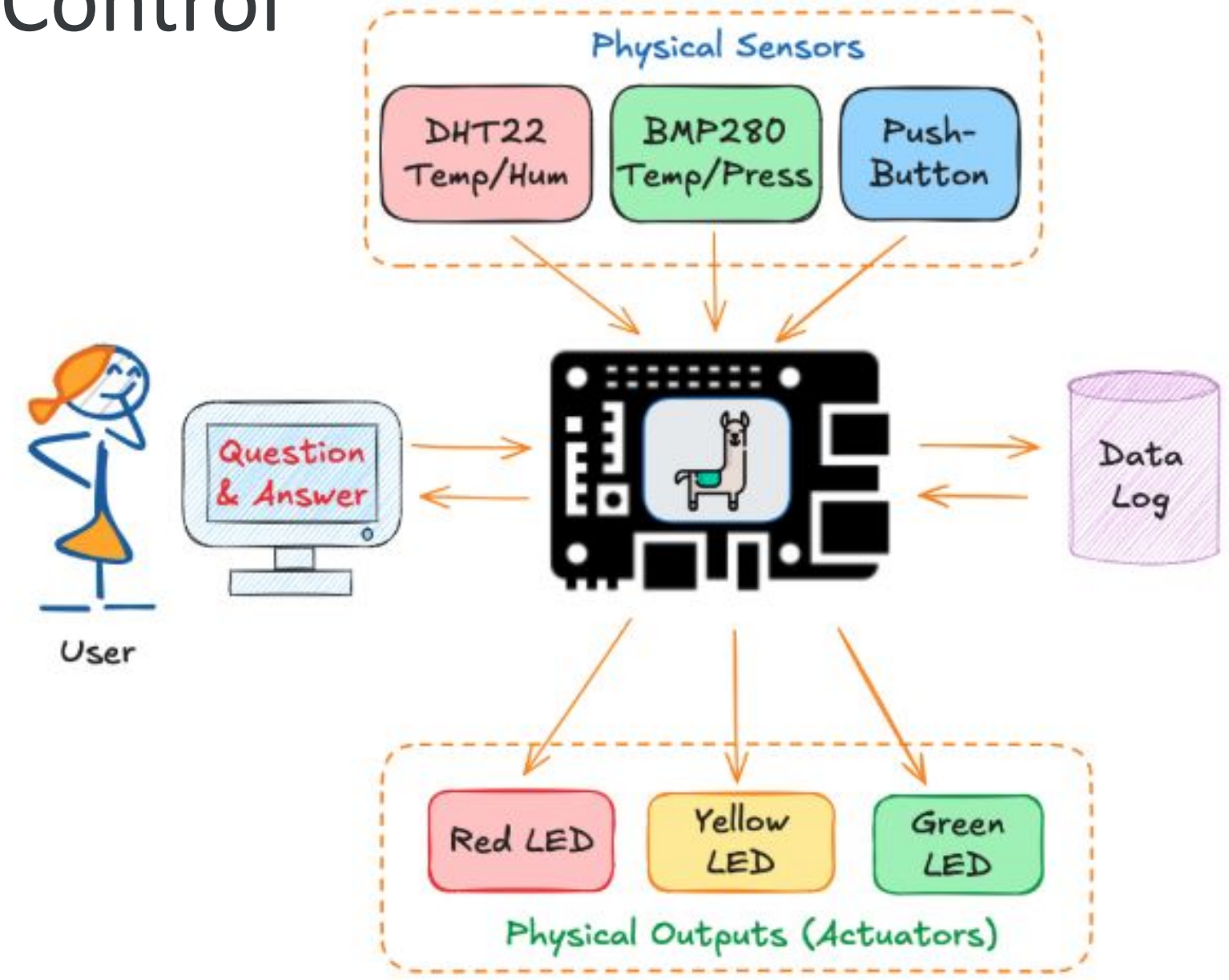
22. SLM for IoT Control

UNIFEI

Marcelo Rovai

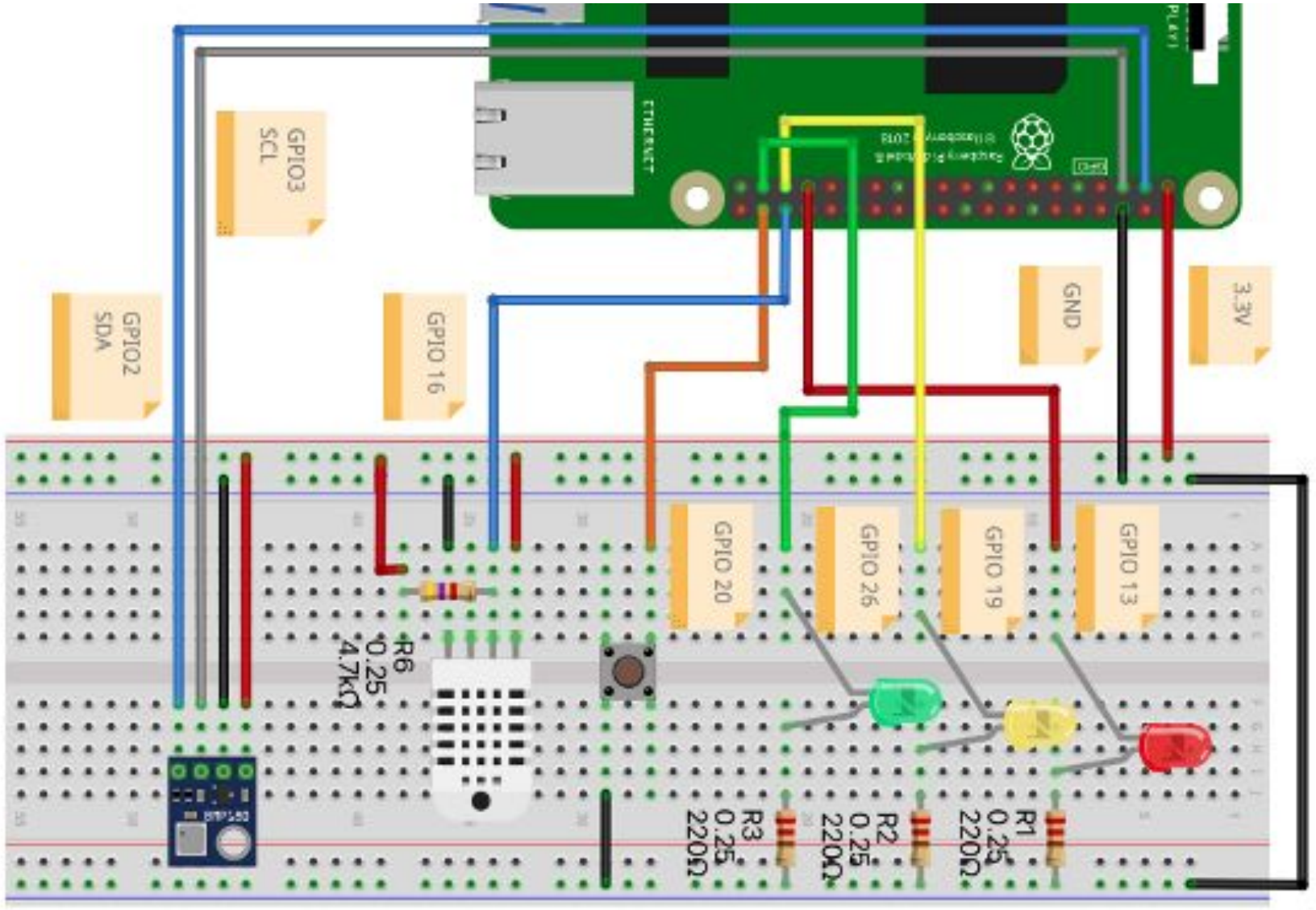
Professor Honoris Causa

SLMs for IoT Control



Hardware

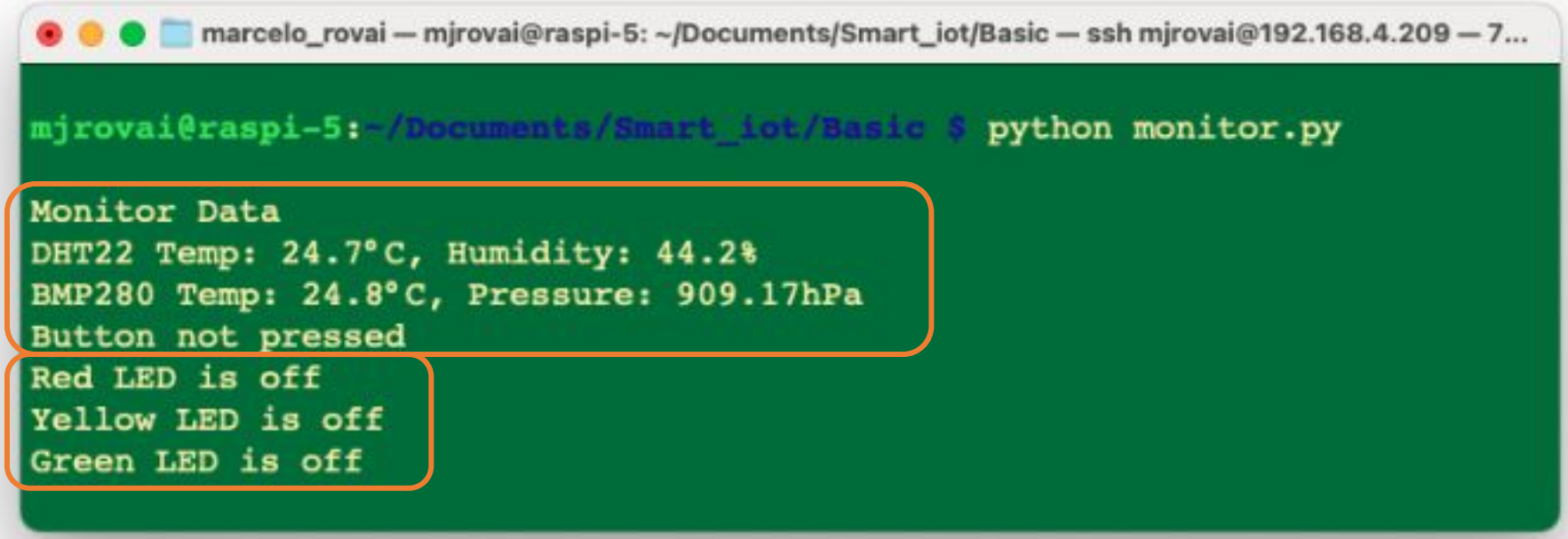
Component	GPIO Pin
DHT22	GPIO16
BMP280 - SCL	GPIO03
BMP280 - SDA	GPIO02
Red LED	GPIO13
Yellow LED	GPIO19
Green LED	GPIO26
Button	GPIO20



Monitor.py

- GPIO Initialization

- collect_data()
- led_status()

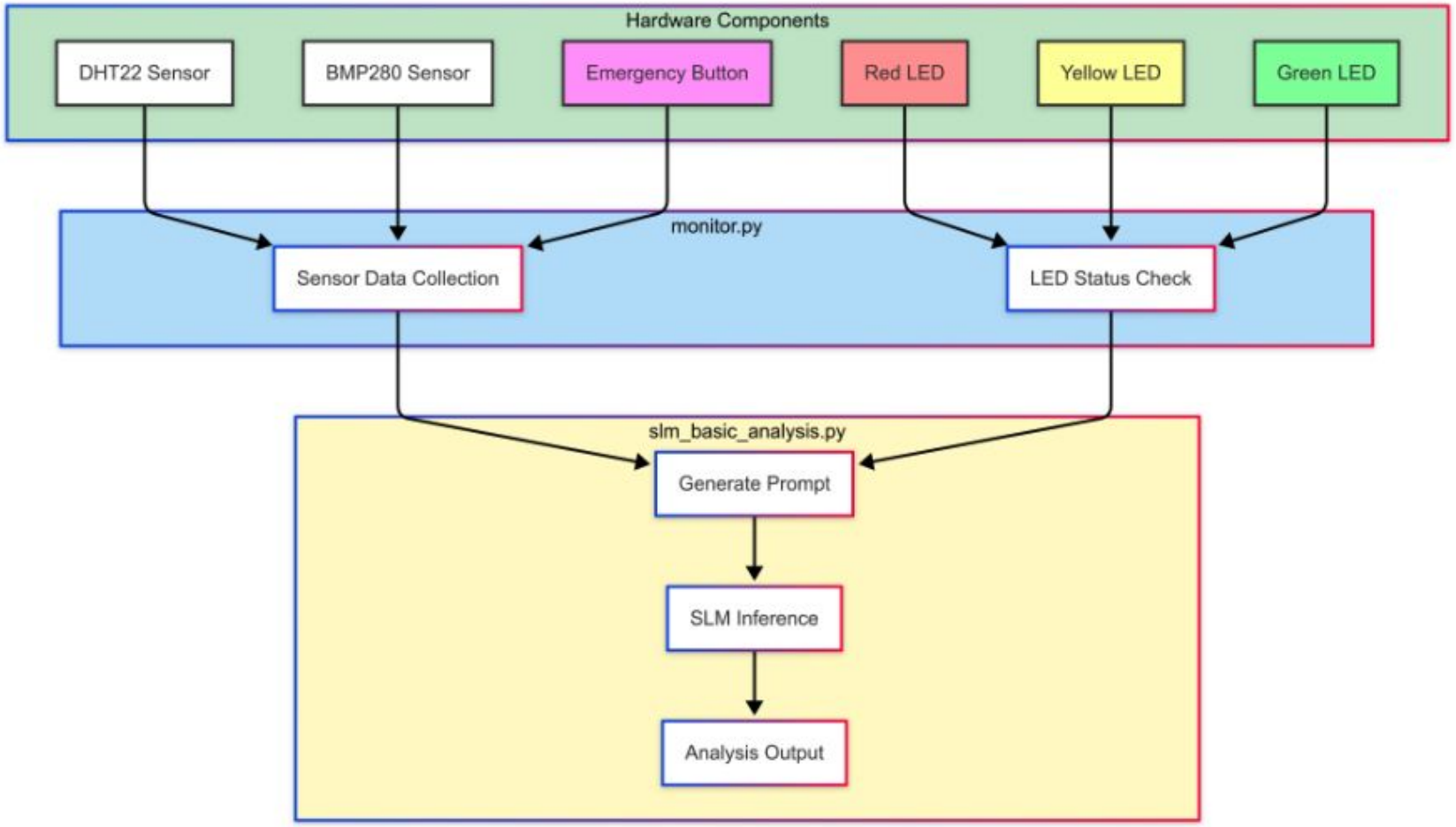


- control_leds(red, yellow, green)

monitor.py

SLM Basic Analysis

- The button, not pressed, shows a **normal operation**
- The button, when pressed, shows an **emergency**
- If the temperature is over 20°C, it means a **warning situation**
- The red LED, when on, indicates a problem/emergency.
- The yellow LED indicates a warning situation when on.
- The green LED is on, indicating the system is OK.

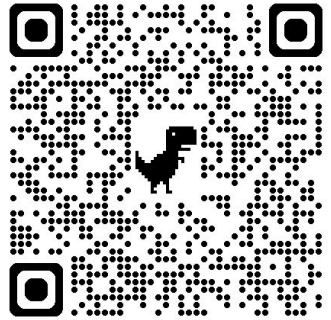


slm_basic_analysis.py

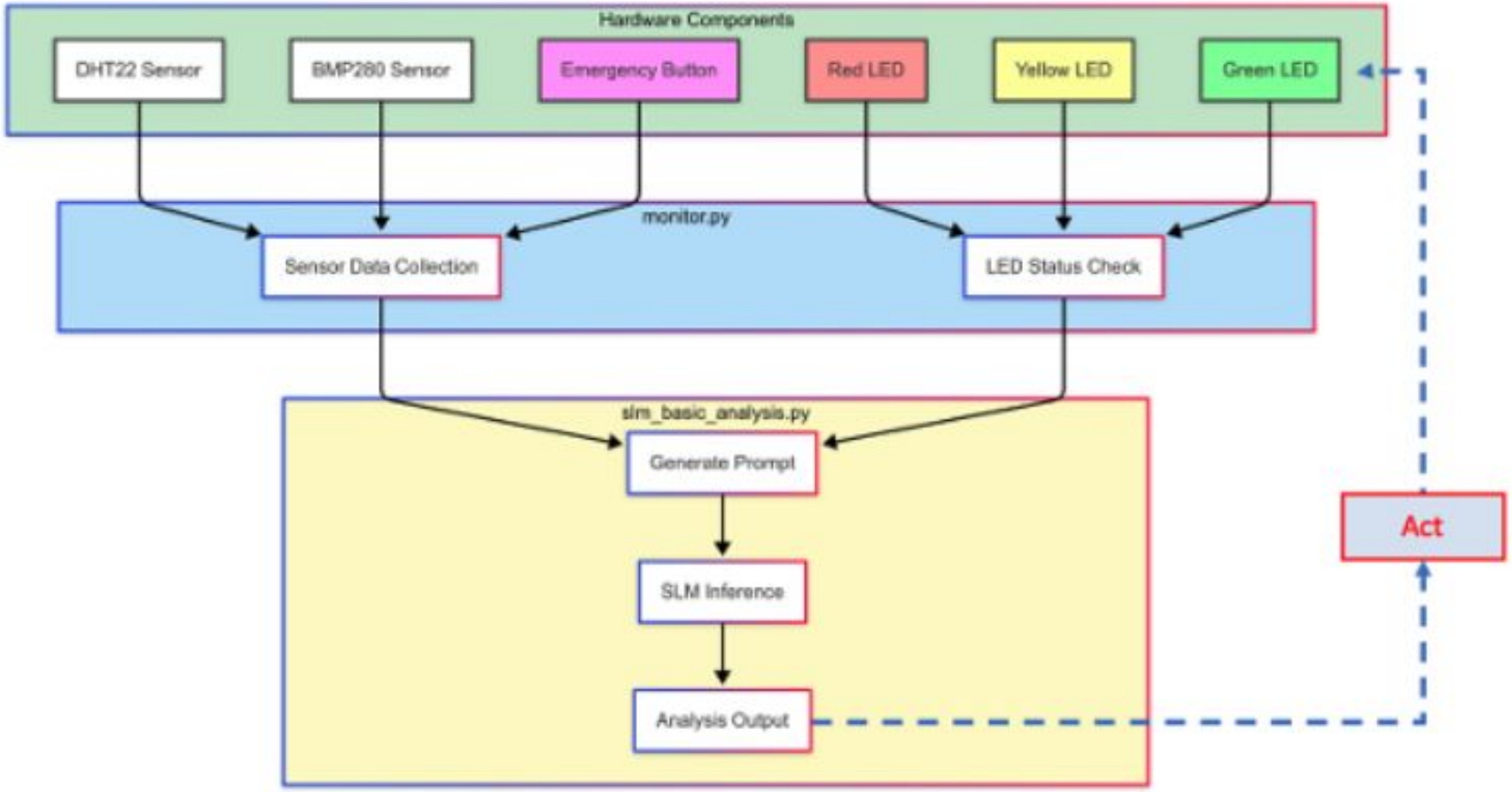
SLMs and IoT Control

Projects

SLM_IoT.ipynb



Acting on Output (Actuators)



slm_basic_act_leds.py

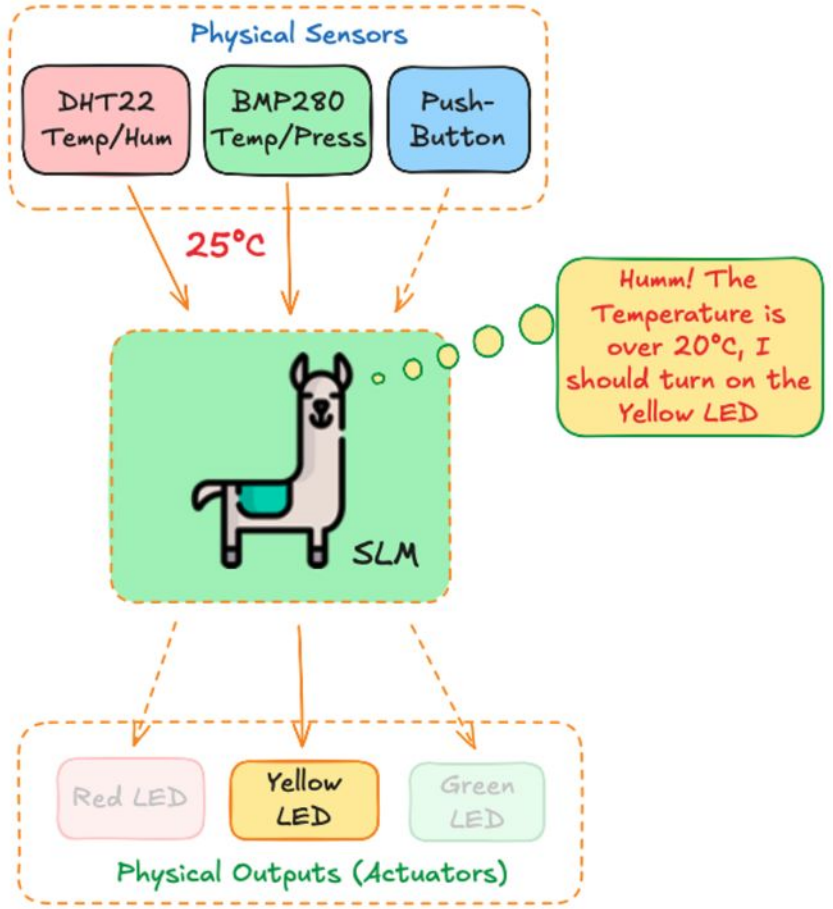
Prompting Engineering

Key Changes in the code:

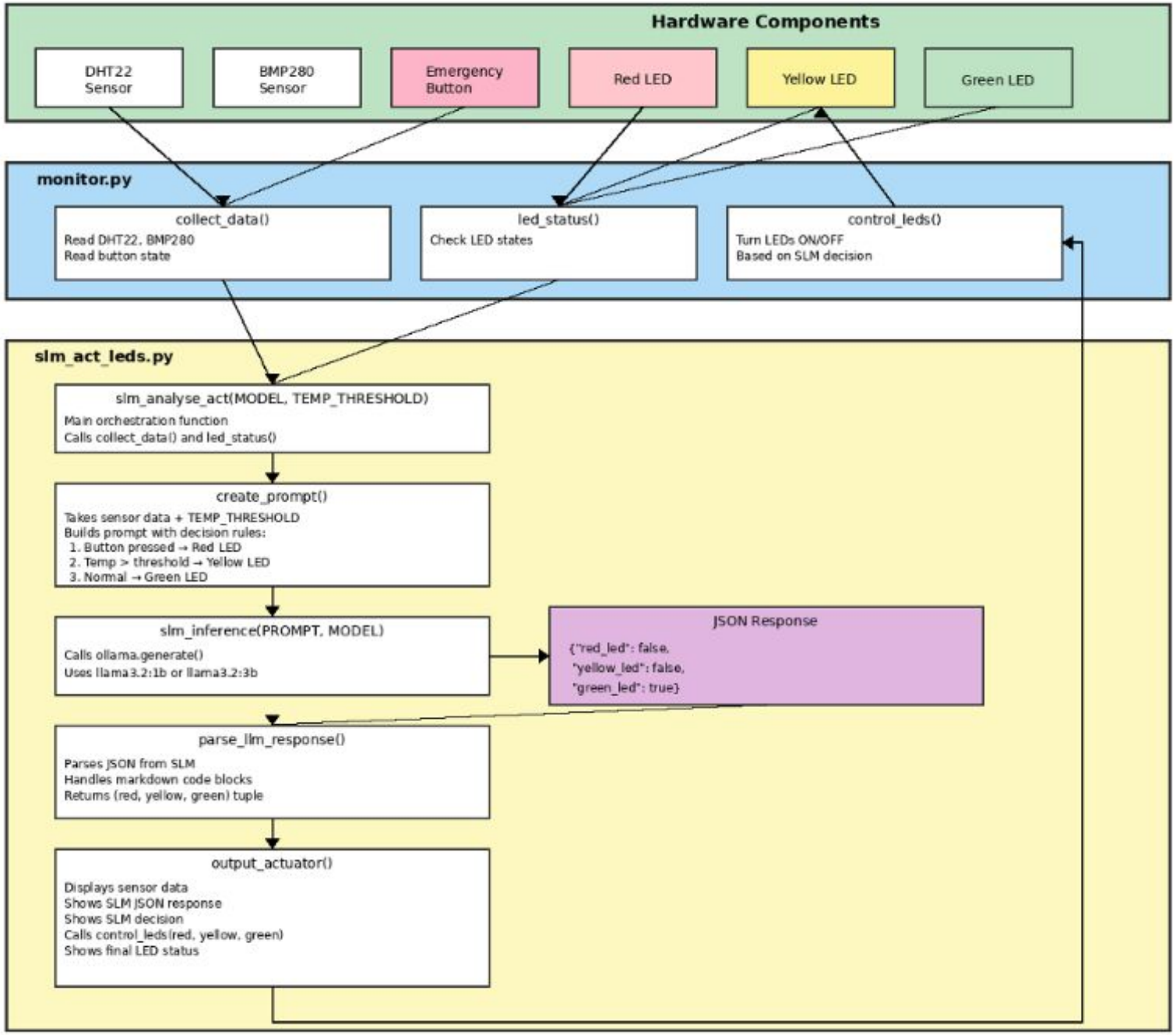
- Added JSON import
- Changed to JSON format - The prompt now asks for a structured JSON response:
- {"red_led": true, "yellow_led": false, "green_led": false}
- Updated parser - Now parses JSON instead of searching for text strings. Includes error handling and fallback to safe state (all LEDs off) if parsing fails.

Why JSON is Better:

- More reliable: No ambiguity about which LEDs to activate
- Structured: Clear true/false values instead of parsing text
- Error-resistant: The parser handles markdown code blocks (some models wrap JSON in ```) and provides safe fallback
- Flexible: Easy to add more fields later if needed

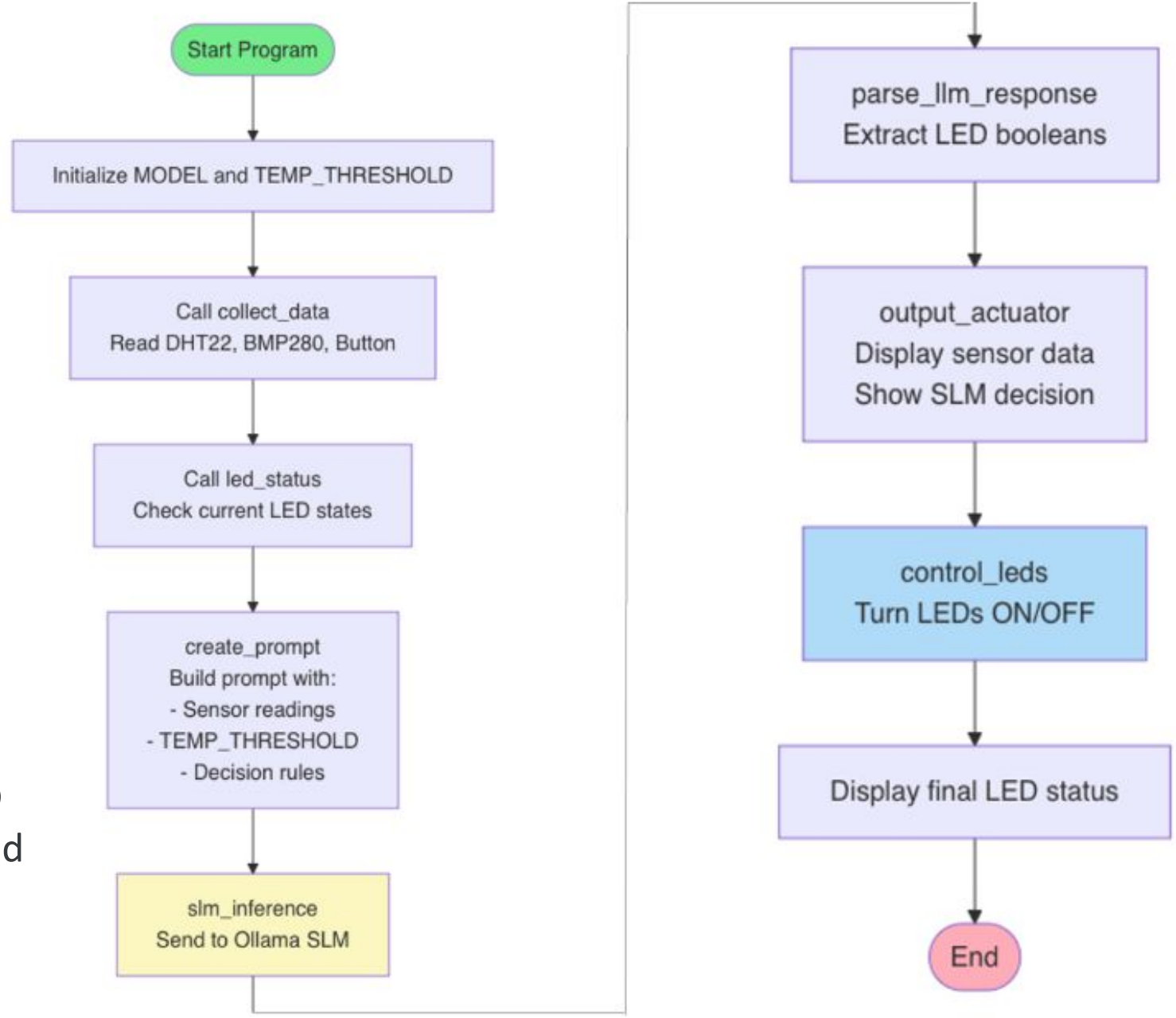


slm_act_leds.py


Hardware Components
DHT22 Sensor
BMP280 Sensor
Emergency Button
Red LED
Yellow LED
Green LED
monitor.py
collect_data()
Read DHT22, BMP280
Read button state
led_status()
Check LED states
control_leds()
Turn LEDs ON/OFF
Based on SLM decision
slm_act_leds.py
slm_analyse_act(MODEL, TEMP_THRESHOLD)
Main orchestration function
Calls collect_data() and led_status()
create_prompt()
Takes sensor data + TEMP_THRESHOLD
Builds prompt with decision rules:
1. Button pressed → Red LED
2. Temp > threshold → Yellow LED
3. Normal → Green LED
slm_inference(PROMPT, MODEL)
Calls ollama.generate()
Uses llama3.2:1b or llama3.2:3b
JSON Response
{"red_led": false,
"yellow_led": false,
"green_led": true}
parse_llm_response()
Parses JSON from SLM
Handles markdown code blocks
Returns (red, yellow, green) tuple
output_actuator()
Displays sensor data
Shows SLM JSON response
Shows SLM decision
Calls control_leds(red, yellow, green)
Shows final LED status

Autonomous system:

- Continuously monitored sensors
- Automatically decided LED states based on pre-defined rules
- No user interaction

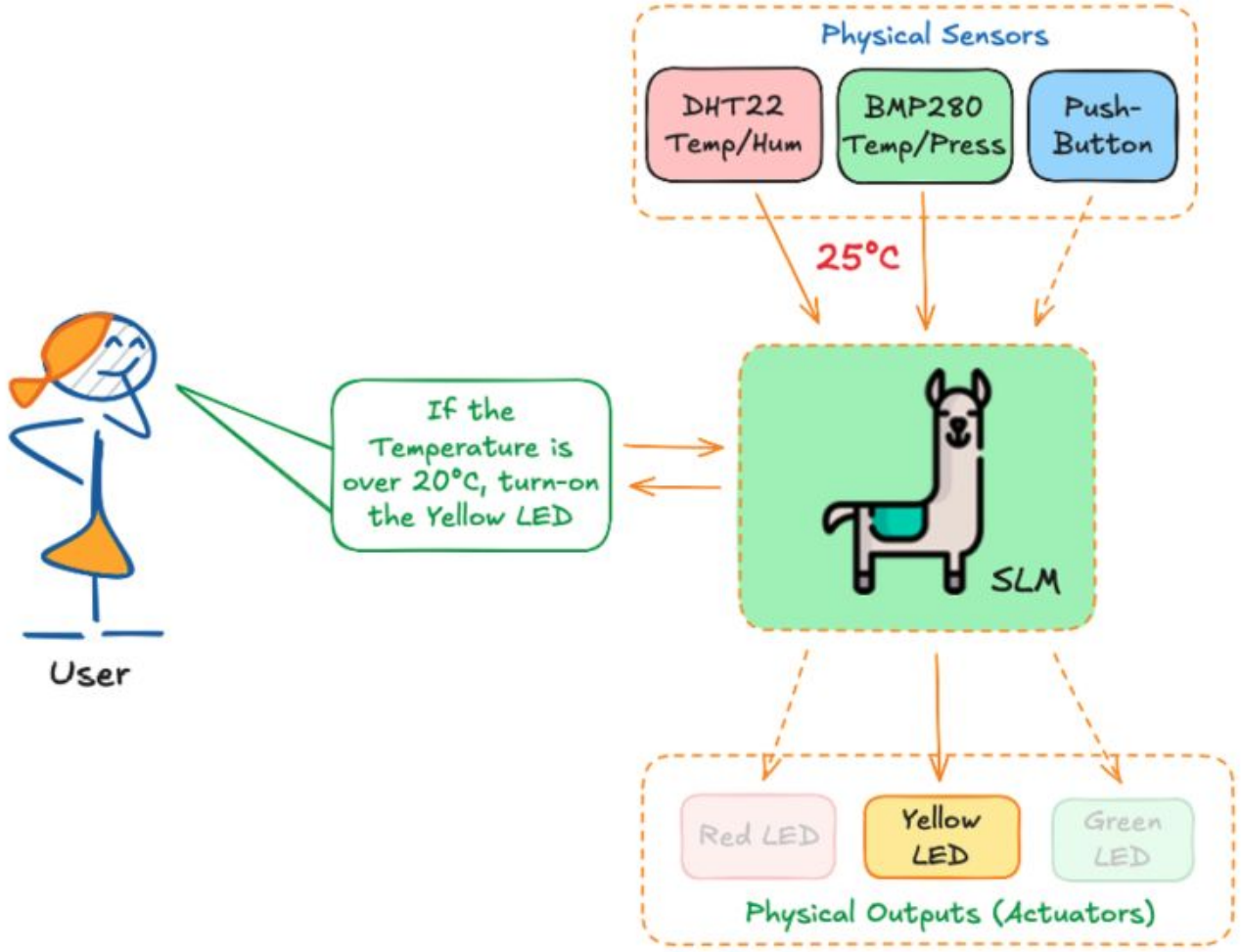


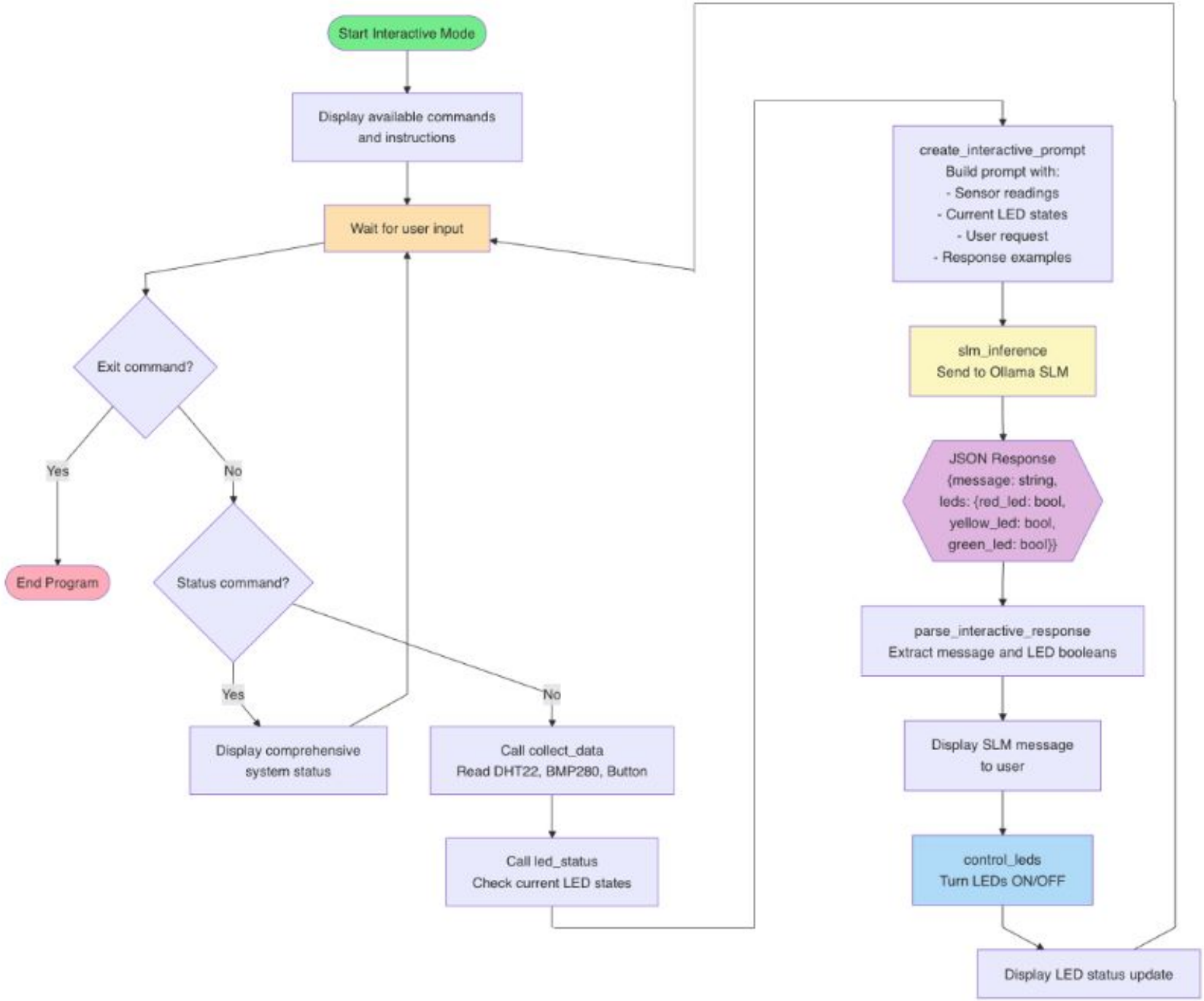
Interactions with Natural Language Commands

Interactive System:

- Waits for user commands
- Accepts natural language queries and commands
- Provides conversational responses
- Executes actions based on user requests
- Displays comprehensive system status

slm_act_leds_interactive.py




Start Interactive Mode
Display available commands and instructions
Wait for user input
Exit command?
Yes
No
End Program
Status command?
Yes
No
Display comprehensive system status
Call collect_data
Read DHT22, BMP280, Button
Call led_status
Check current LED states
create_interactive_prompt
Build prompt with:
- Sensor readings
- Current LED states
- User request
- Response examples
slm_inference
Send to Ollama SLM
JSON Response
{message: string,
leds: {red_led: bool,
yellow_led: bool,
green_led: bool}}
parse_interactive_response
Extract message and LED booleans
Display SLM message to user
control_leds
Turn LEDs ON/OFF
Display LED status update

Prompt Optimization and Efficiency

1. **System message**: It defines the assistant's behavior and should be sent once at initialization, not at PROMPT
2. **Prompt:** Should be drastically shortened
3. **API**: Ollama.Chat Instead of Ollama.Generate
4. **Model:** Pre-loading

```
Assistant: [Thinking...]
Assistant: All LEDs turned off.

LED Update: Red=OFF, Yellow=OFF, Green=OFF


Total Duration: 88.81 seconds
prompt_eval_duration: 80.41 s
load_duration: 1.98 s
eval_count: 32
eval_duration: 6.41 s
eval_rate: 4.99 tokens/s
```

```
Assistant: [Thinking...]
Assistant: All LEDs are now OFF

LED Update: Red=OFF, Yellow=OFF, Green=OFF


Total Duration: 13.01 seconds
prompt_eval_duration: 6.54 s
load_duration: 0.16 s
eval_count: 33
eval_duration: 6.31 s
eval_rate: 5.23 tokens/s
```

slm_act_leds_interactive_optimized.py

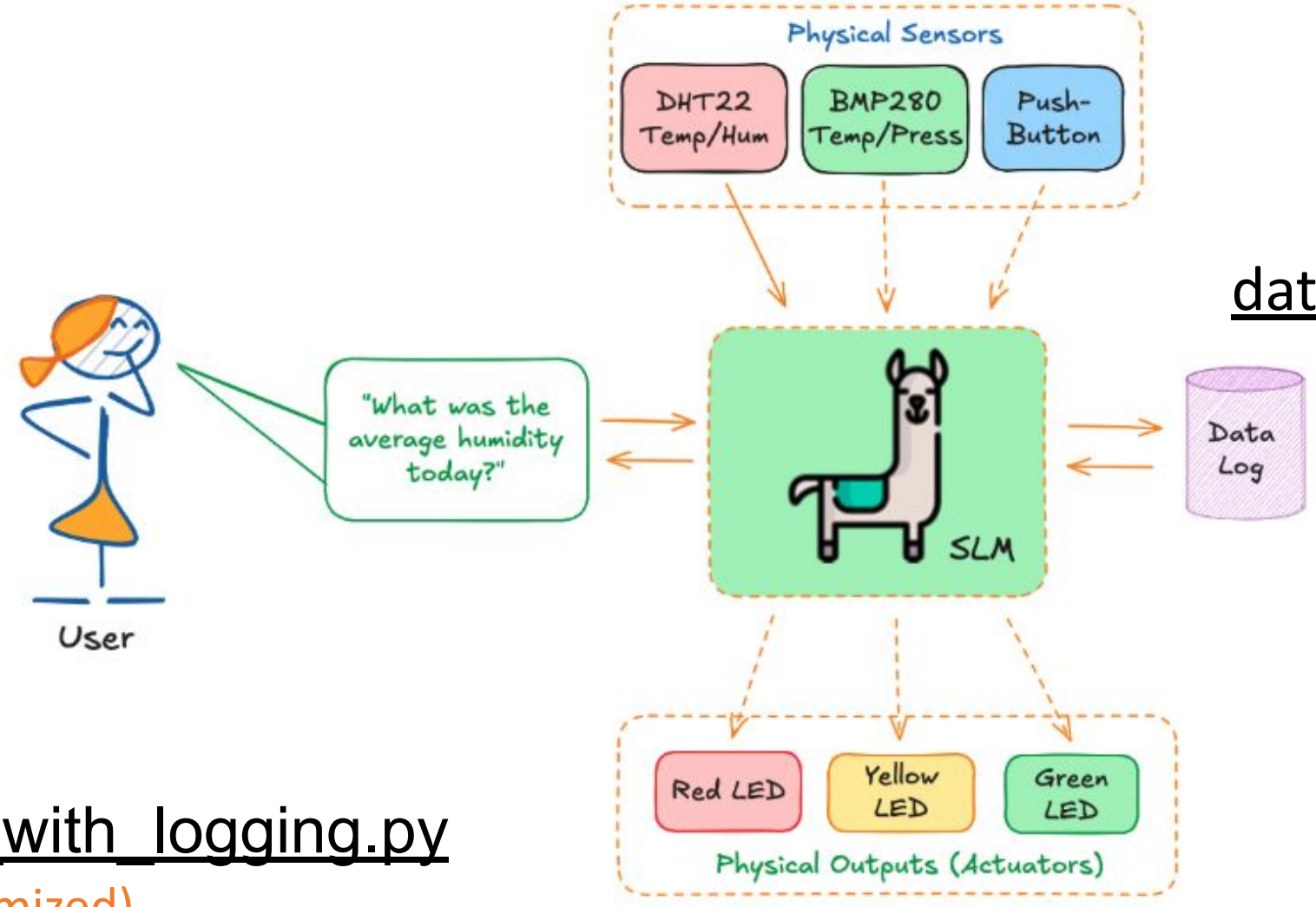
Using Pydantic

Using **Pydantic** is a robust way to improve the reliability, efficiency, and maintainability of our system, mainly since we rely on the LLM to output precise JSON

- **No more parsing errors** - Guaranteed valid JSON
- **Cleaner output** - No markdown code blocks
- **Type safety** - IDE autocomplete and type checking
- **Faster generation** - Constrained decoding is more efficient
- **Better validation** - Pydantic catches invalid values

slm_act_leds_interactive_pydantic.py

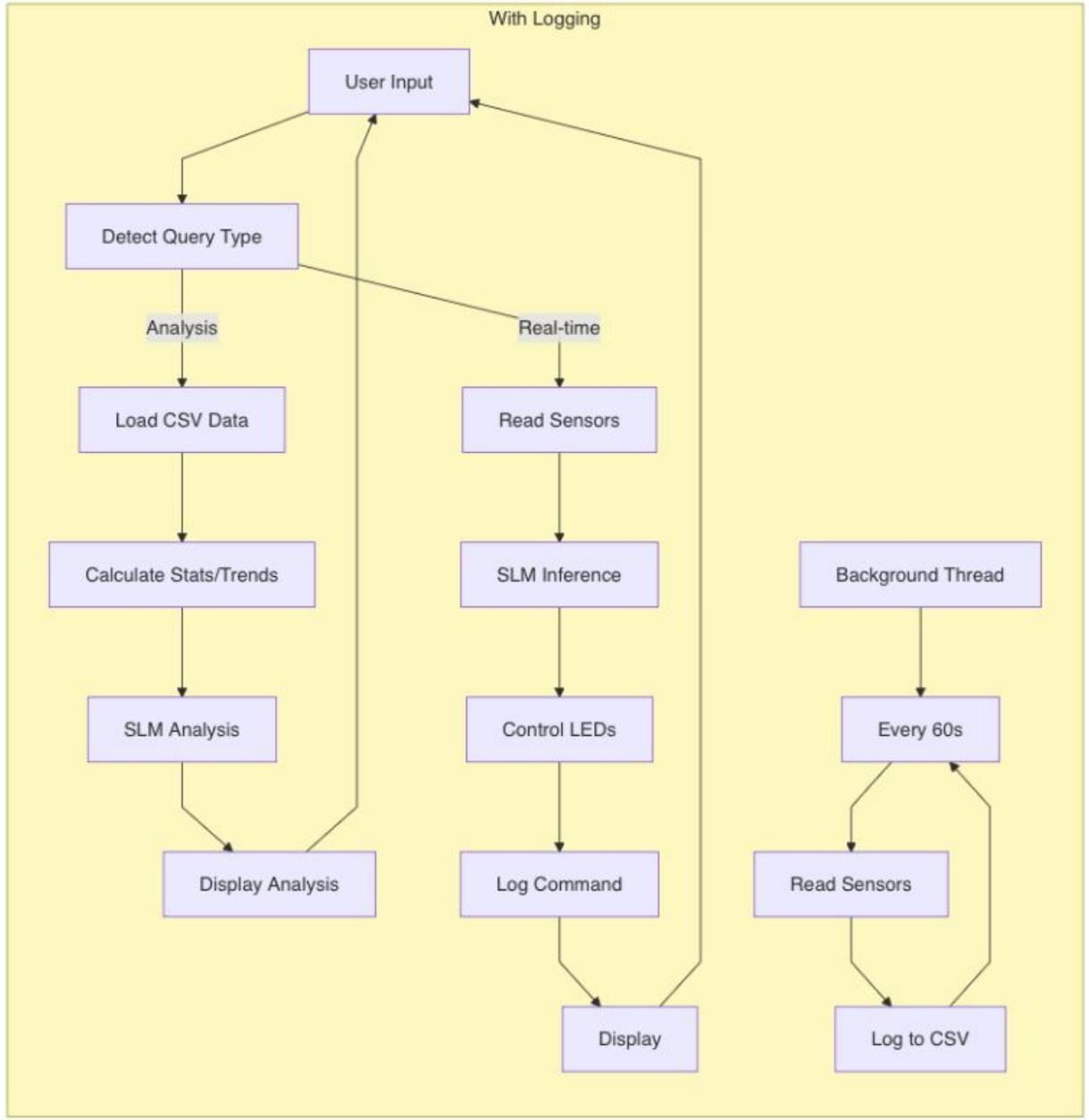
Adding Data Logging



data_logger.py

slm_act_leds_with_logging.py

(Not optimized)


With Logging
User Input
Detect Query Type
Analysis
Real-time
Load CSV Data
Read Sensors
Calculate Stats/Trends
SLM Inference
Background Thread
SLM Analysis
Control LEDs
Every 60s
Display Analysis
Log Command
Read Sensors
Display
Log to CSV

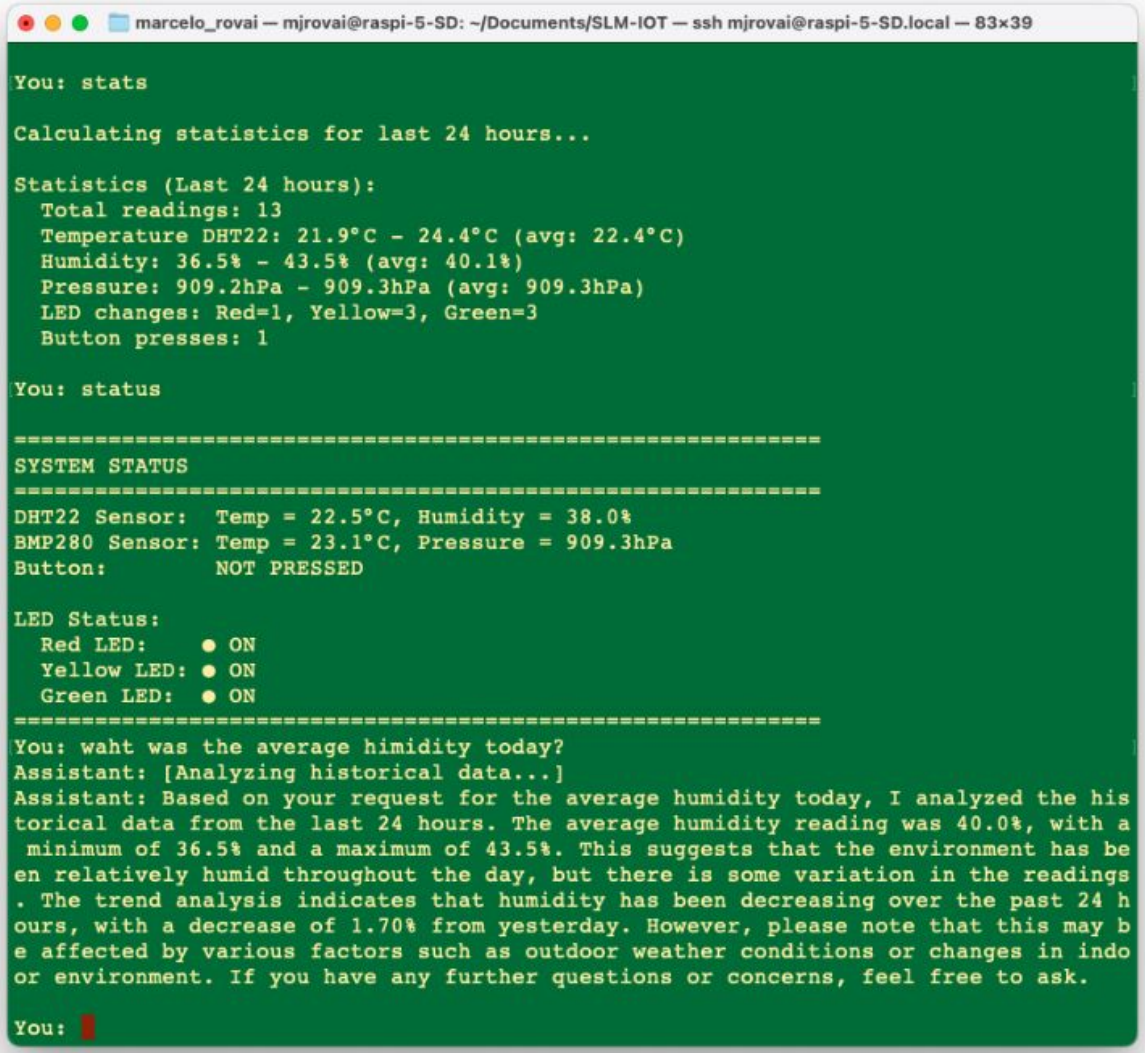
Querying the Datalog

command_history

timestamp	user_command	slm_response	red_led	yellow_led	green_led
2025-10-29T10:14:20.718046	what is the current temperature?	The current temperature is 21.9°C from DHT22 and 22.8°C from BMP280.	FALSE	FALSE	FALSE
2025-10-29T10:15:38.480255	turn on all leds	All LEDs turned on.	TRUE	TRUE	TRUE
2025-10-29T10:16:45.710702	if the button is pressed, turn off the red led	Red LED turned off.	FALSE	TRUE	TRUE
2025-10-29T10:17:54.806233	if the button is pressed, turn off all leds	Red LED turned off.	FALSE	TRUE	TRUE
2025-10-29T10:19:00.775716	turn off all leds	All LEDs turned off.	FALSE	FALSE	FALSE
2025-10-29T10:20:18.943857	if the button is pressed, turn on all leds	All LEDs turned on.	TRUE	TRUE	TRUE
2025-10-29T10:26:40.514342	what is the temperature actual?	The current temperature is 22.1°C from DHT22 and 27.2°C from BMP280.	TRUE	TRUE	TRUE

sensor_readings

timestamp	temp_dht	humidity	temp_bmp	pressure	button_pressed	red_led	yellow_led	green_led
2025-10-29T10:13:12.504743	21.9	40.8	22.7673828125	909.2205601683819	FALSE	FALSE	FALSE	FALSE
2025-10-29T10:14:12.836790	22.0	40.9	22.7966796875	909.2401774060825	FALSE	FALSE	FALSE	FALSE
2025-10-29T10:15:13.169628	22.0	40.9	22.79921875	909.2460173312777	FALSE	FALSE	FALSE	FALSE
2025-10-29T10:17:13.501777	22.0	40.2	22.8359375	909.24629860536	TRUE	FALSE	TRUE	TRUE
2025-10-29T10:18:13.832472	22.0	40.4	22.808984375	909.2400992352984	FALSE	FALSE	TRUE	TRUE
2025-10-29T10:19:14.163144	22.0	40.8	22.81875	909.2775134584971	FALSE	FALSE	FALSE	FALSE
2025-10-29T10:21:14.750170	22.0	40.1	22.8578125	909.3348075973149	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:22:15.083681	22.0	39.9	22.8724609375	909.3255284665197	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:23:15.417258	22.0	39.5	22.848046875	909.3310873280028	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:25:15.999074	22.1	40.0	22.8724609375	909.3455856635653	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:26:16.332596	23.6	43.5	25.2212890625	909.2463453002865	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:27:16.667466	24.4	36.5	23.6203125	909.3112568670979	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:28:16.999152	23.4	37.6	23.266015625	909.3002891478848	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:30:17.581253	22.5	39.1	23.097265625	909.2874610215446	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:31:17.916684	22.4	39.5	23.055859375	909.2812657542946	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:32:18.254584	22.4	38.4	23.0705078125	909.3266505549615	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:33:18.585493	22.3	39.0	23.04609375	909.3204667356669	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:34:18.915975	22.2	38.7	23.0533203125	909.3024426505486	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:35:19.246659	22.3	38.0	23.06796875	909.2795854094724	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:36:19.577312	22.3	38.1	23.0802734375	909.3467049671432	FALSE	TRUE	TRUE	TRUE
2025-10-29T10:37:19.907911	22.3	37.8	23.0900390625	909.342026800072	FALSE	TRUE	TRUE	TRUE



⚠️ IMPORTANT: The Notebook Kernel should end after using to liberate GPIOs

> The problem is that **Jupyter Notebook is still holding onto the GPIO pins** even after our code finishes running. When we try to run it from the terminal, those pins are already claimed by the Jupyter process. So, before running a code that deals with GPIO in the terminal, In Jupyter:
>
> - Click **Kernel → Restart Kernel**
> - Or: Click **Kernel → Shutdown Kernel**

Questions?

Prof. Marcelo J. Rovai

rovai@unifei.edu.br

UNIFEI