OVERVIEW - SCHEMATIC AND COMMUNICATION PROTOCOL

Wiring Schematic

Items	Model	Quantity	Power Supply
NodeMCU ESP8266	ESP8266-12E	3	4.5 to 9 VDC
Raspberry Pi (RPi)	RPi 5	1	5 to 15 VDC (27W)
Photoresistor		3	With resistors
Light Emitting Diode	RGB LEDs	4	With resistors
Resistor	220 Ω	4	
Resistor	10k Ω	3	
Push Button	Bread-board Mount	1	

Table 1. Equipment list

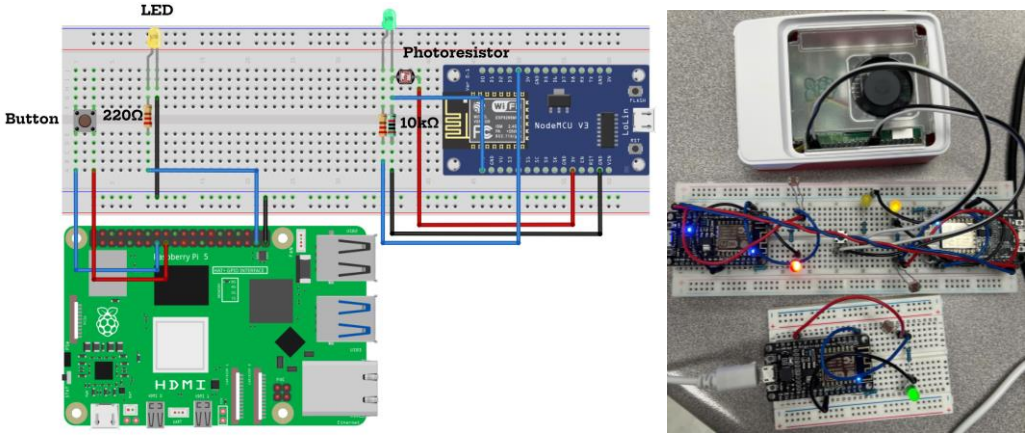


Figure 1. Wiring schematic for RPi5 and each ESP8266 module

- ESP8266:
    - 3.3V → Photoresistor (+)
    - A0 → Photoresistor (+)
    - D4 → RGB Led (+)
- Raspberry Pi 5
    - 3.3V → Button (+)
    - GPIO 22 → Button (-)
    - GPIO 26 → Yellow LED (+)

Communication Protocol/Message Description

Protocol Overview

- The system uses broadcast UDP messages over a mobile hotspot network to facilitate communication among multiple ESP8266 devices and a Raspberry Pi. Each message follows a specific format, allowing the devices to identify the type of message, sender, and content.

Message Format

- Each message contains:
    - Start Bit: Identifies the beginning of a message.
    - Payload: Contains specific information (e.g., sensor readings, commands).
    - End Bit: Marks the end of a message.
- General Message Structure:
    - Start Bit: "+++" | "~~~" → indicates the start of the message.
    - End Bit: "***" | "---" → indicates the end of the message.
    - Payload: Content varies depending on the message type (sensor data or control command).
- For example: +++"<PAYLOAD>"***

Communication Flow

- Communication Between ESP8266 Modules:
    - Each ESP8266 device broadcasts its Swarm ID and analog sensor reading periodically over UDP.
    - All ESP8266 devices listen to these broadcasts to compare their sensor readings and determine if they should be the Master based on the highest reading.
- Sensor Data Broadcast from ESP8266 to Raspberry Pi:
    - Each ESP8266 sends its Swarm ID (self-assigned from the last digit of its IP address) and analog sensor reading (from a light sensor) to the Raspberry Pi.
    - The RPi5 listens to incoming messages and processes each one by logging the data and triggering LED feedback based on the analog reading.
- System Reset Broadcast from RPi5 to ESP8266:
    - When the RPi5's button is pressed, it broadcasts a reset command "RESET_REQUESTED" to all ESP8266 devices.

- ESP8266 devices, upon receiving this command, reset their internal state as required (not explicitly coded here, but implied for the reset functionality).

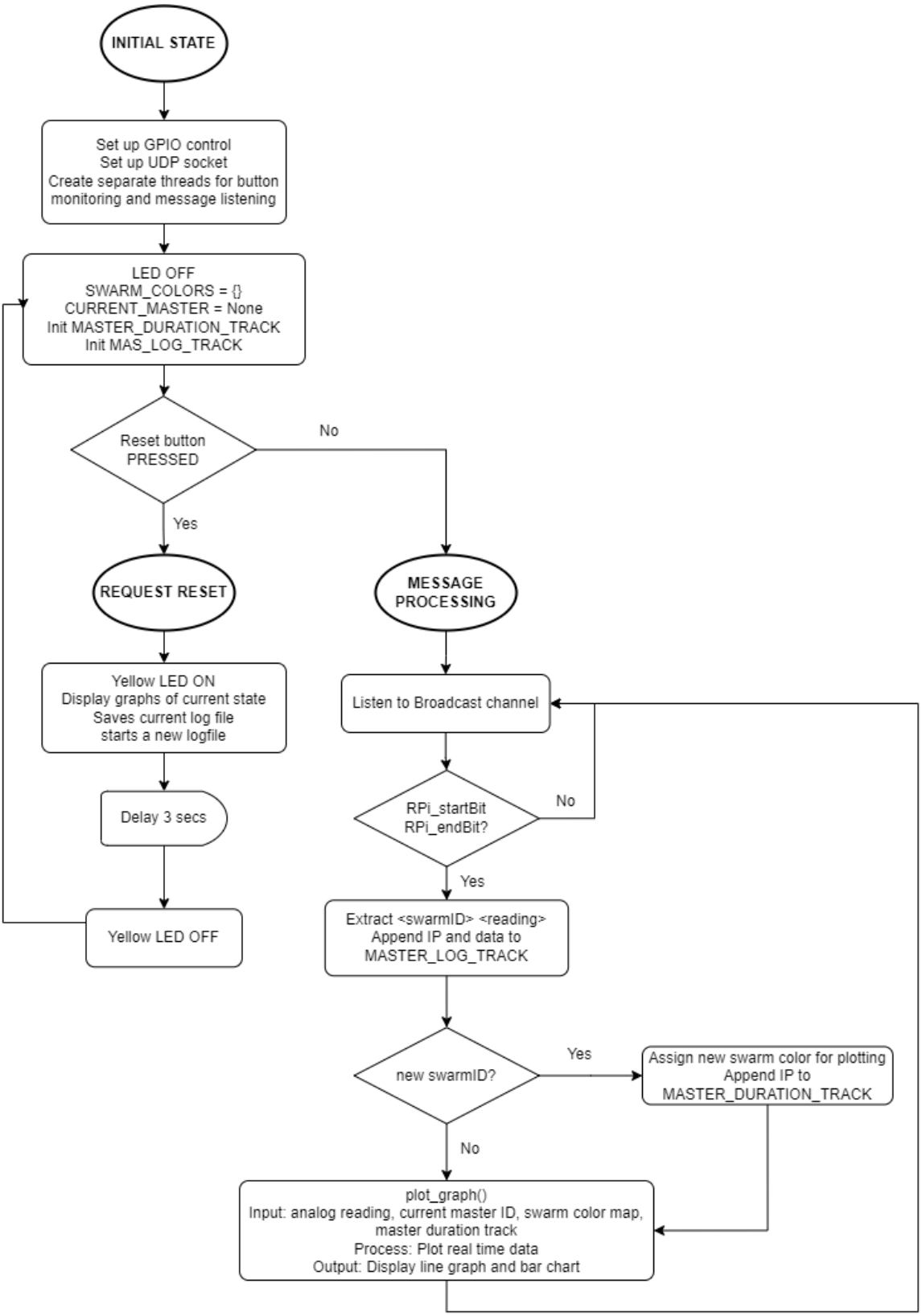
Detailed Message Descriptions

- Sensor Data Message (ESP8266 to ESP8266):
  - Start Bit: "~~~"
  - Payload: "<Swarm ID>,<Analog Reading>"
    - Swarm ID: Unique identifier for each ESP8266 (based on the last digit of its IP address).
    - Analog Reading: Light sensor reading as an integer, used to compare against other modules' readings.
  - End Bit: "---"
  - Example: If an ESP8266 with Swarm ID 4 has an analog reading of 750, the message would look like: "~~~4,750---"
- Sensor Data Message (from ESP8266 Master to RPi5):
  - Start Bit: "+++"
  - Payload: "<Swarm ID>,<Analog Reading>"
  - End Bit: "***"
- Reset Request Message (from RPi5 to all ESP8266): When the button connected to the RPi5 is pressed, it broadcasts a reset request message to all ESP8266 devices. This message has the following structure:
  - Start Bit: "+++"
  - Payload: "RESET_REQUESTED"
  - End Bit: "***"

Message Type	Direction	Start Bit	Payload	End Bit
Sensor Data	ESP <> ESP	"~~~"	"<SwarmID>,<AnalogReading>"	"---"
Master Data	ESP → RPi	"+++"	"<SwarmID>,<AnalogReading>"	"***"
Reset Request	RPi → ESP	"+++"	"RESET_REQUESTED"	"***"

PART 1 - RASPBERRY PI WIFI SETUP AND PACKET DELIVERY

Flowchart



Main Functionality of Raspberry Pi Code

- Input:
    - Button Press: Detects button presses to trigger log saving, log file creation, and system reset.
    - UDP Messages: Listens for incoming messages containing analog sensor data and Swarm IDs. These messages provide data for the master tracking and analog readings.
- Process:
    - Three threads: button_thread, receive_thread, graph_thread
    - GPIO Management: Monitors button presses (to save logs and trigger resets) and controls an LED to signal reset status.
    - Message Parsing and Logging: Processes received UDP messages to log Swarm ID, analog readings, and tracks the master Swarm ID.
    - Swarm Master Tracking: Keeps track of the current master Swarm, their durations, and logs all relevant data.
    - Real-Time Graphing: Updates a real-time graph displaying both analog readings and the total time each Swarm device has been the master.
- Output:
    - Logs: Generates and saves logs to a file that includes a summary of master device durations and raw data for each master.
    - LED Indicator: Controls the yellow LED to show the reset status.
    - Real-Time Graphs: Displays live graphs of analog sensor readings and master device durations in a two-panel plot.

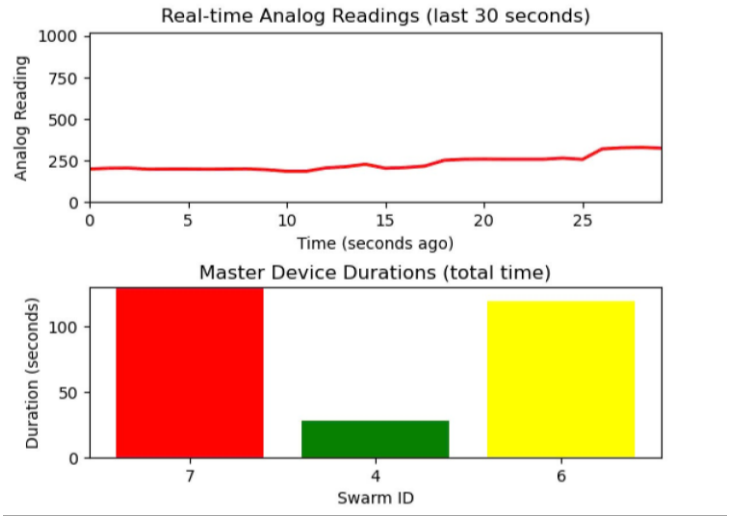
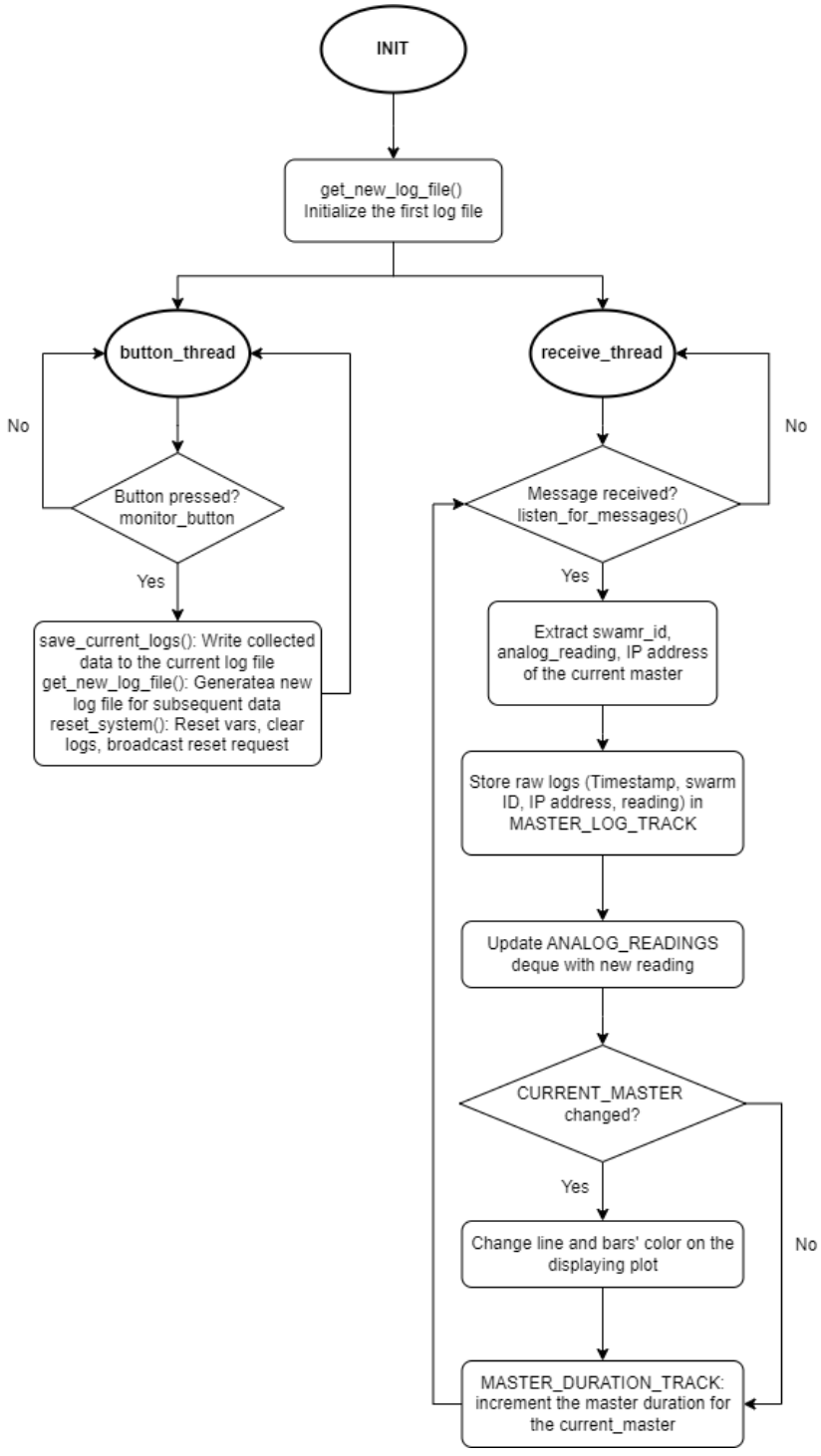


Figure 2. Real-time graphs

Data Structures/Log Files



- Key Data Structures and Fields:
    - MASTER_LOG_TRACK (dict):
        - Stores logs of data received from each device (IP-based).
        - Fields: IP address, log entry (timestamp, swarm_id, analog reading).
    - MASTER_DURATION_TRACK (defaultdict):

        - Tracks the duration (in seconds) each Swarm ID has been the master.
        - Fields: swarm_id, total duration (seconds).
    - SWARM_COLORS (dict):
        - Assigns a color to each Swarm ID.
        - Fields: swarm_id, assigned color (red, green, yellow).
    - ANALOG_READINGS (deque):
        - Stores the last 30 analog readings from the devices.
        - Field: analog reading value.

- Log File:
    - LOG_FILE (str): The current log file name, dynamically generated based on the timestamp of file creation.
    - Log Content: Includes a summary of all devices that became masters (IP addresses), how long each device was a master (from the beginning), and raw data from each master.

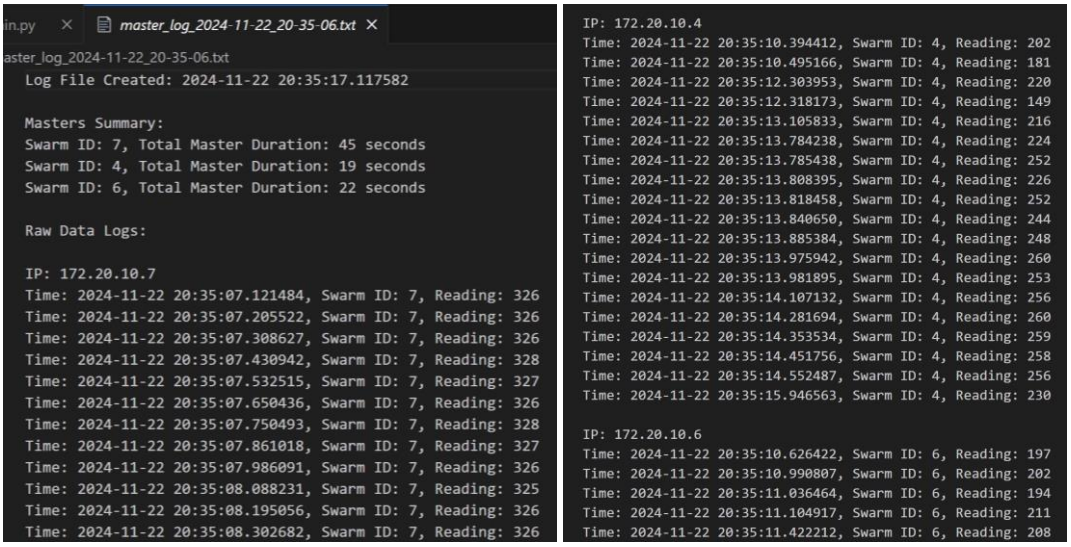
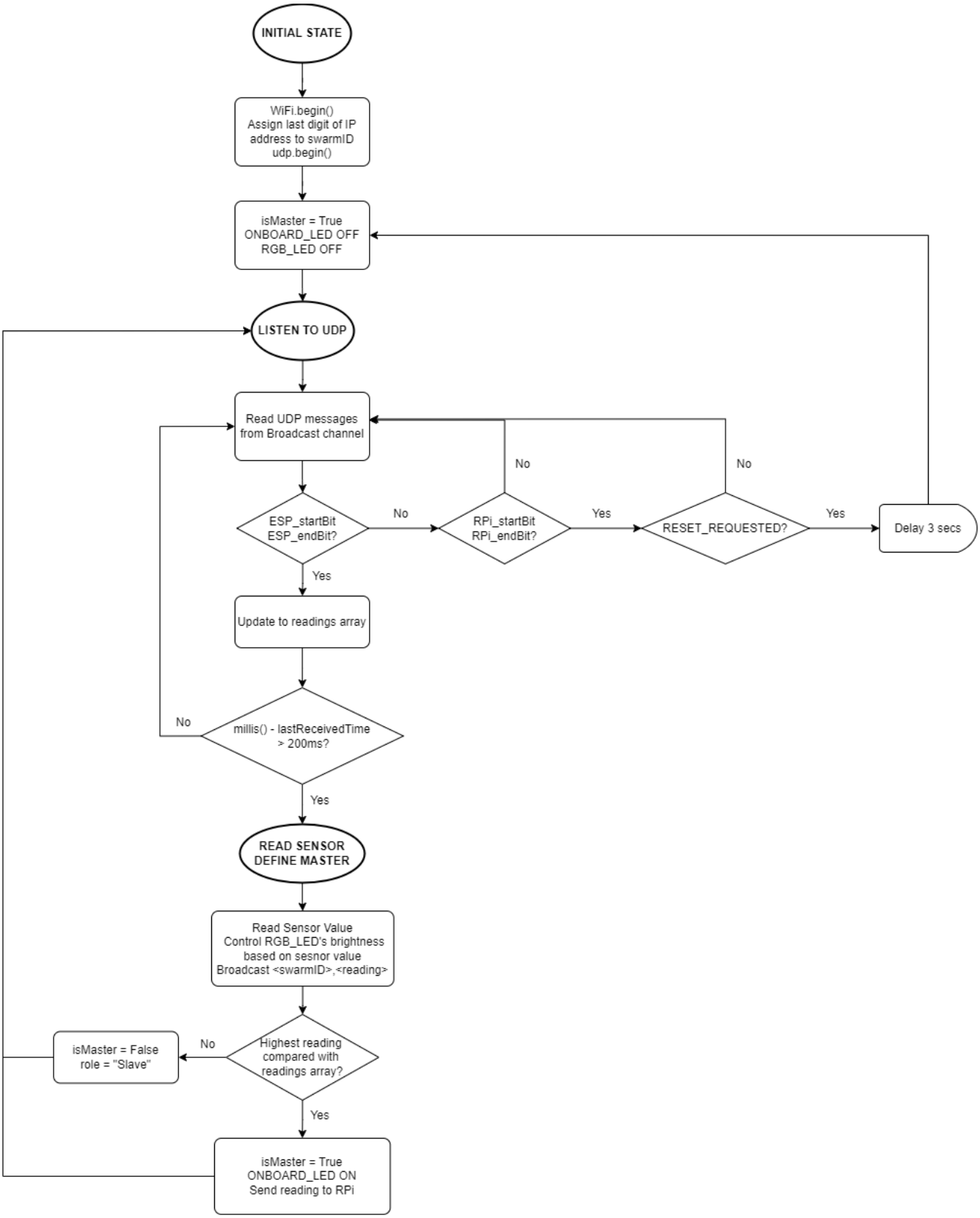


Figure 3. Log content example

- Data Flow:
    - Input: Messages received from devices (IP address, Swarm ID, analog reading).
    - Process: Log the data, assign a color to Swarm IDs, track the master device, update readings.
    - Output:
        - Log file containing master durations and raw data logs.
        - Real-time plots with updated readings and master durations.

PART 2 - ESP WIFI SETUP AND PACKET DELIVERY

Flowchart



State Descriptions

1. **State 1: Initial Setup**

   The ESP8266 connects to the WiFi network, assigns a unique swarmID based on the last digit of the IP address, and initializes UDP communication. The LED pins are set up, and the slope and intercept for controlling the brightness of the RGB LED are calculated in advance.

2. **State 2: Sensor Reading and Master Determination**

   The ESP8266 continuously reads the light sensor value and controls the brightness of the RGB LED according to the sensor reading. If the device is the Master, the Master status LED flashes as well. It then checks for incoming UDP messages from other devices or the RPi5.

3. **State 3: Broadcasting and Receiving Data**

   The device broadcasts its sensor reading over UDP if no message has been received for a set time (silentTime = 100ms). It also processes incoming messages from other ESP8266 devices (to update readings) and from the RPi5 (to handle reset requests). The Master device sends its reading and ID to the RPi5 and adjusts its status based on the highest sensor reading in the swarm.

Main Functionality

- Input:
  - WiFi credentials and UDP communication setup for network interaction.
  - Analog light sensor readings (analogValue) from the photoresistor connected to pin A0.
  - UDP packets from other ESP devices (light sensor readings) and from the Raspberry Pi (reset requests).
- Process:
  - WiFi Connection: The ESP8266 connects to the specified WiFi network and initializes UDP communication on a given port.
  - LED Brightness Control: Based on the light sensor readings (then deduce to be a linear equation), calculate the brightness level for the LED based on the analog input. The formula scales the input value and adjusts it with a constant offset.
    - Common range of sensor readings: 0 – 1024
    - Range of analogWrite(): 0 – 255
      - ➔ Slope = 0.25 → Intercept = -6
  - Swarm ID Assignment: Each ESP assigns itself a unique swarmID derived from its IP address, ensuring no conflicts in the swarm of devices.

- Data Broadcasting: If no UDP messages are received for a specified silent period, the device broadcasts its light sensor reading along with its swarmID to other devices.
- Role Determination (Master/Slave): Each ESP device compares its reading to others in the swarm. The device with the highest sensor reading becomes the Master, and it broadcasts this information. If it's not the Master, it assumes the Slave role.
- Message Handling: The device listens to incoming UDP messages, processes ESP-to-ESP data (light sensor readings), and handles reset requests from the Raspberry Pi (resetting the Master role and LED states).

- Output:
  - LED Indicators: The RGB_LED's brightness is adjusted based on the light sensor readings, and the ONBOARD_LED turns on when possessing the highest photoresistor's reading, providing visual feedback on the sensor's intensity and Master role.
  - UDP Messages:
    - ESP-to-ESP: The device broadcasts its sensor reading, swarmID, and status.
    - RPi Communication: The Master sends its reading and ID to the RPi5 to log and display the data. If a reset is requested, the Master role is reset, and LEDs are switched off.