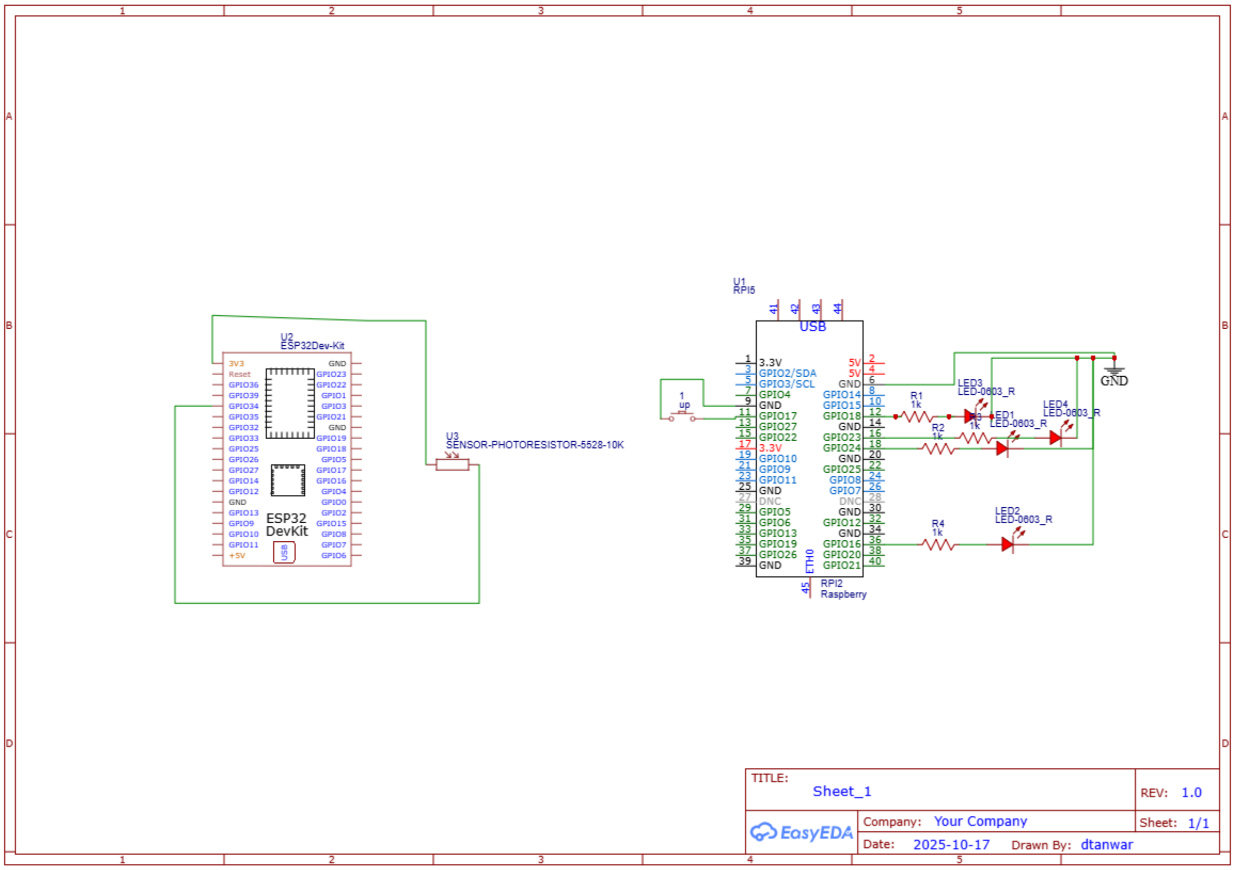
Video Link:
https://drive.google.com/file/d/1Iyfum8h0a8bYc1gP6p4GwSR5RQ6uivGz/view?usp=sharing

Overview - Schematic and Communication Protocol

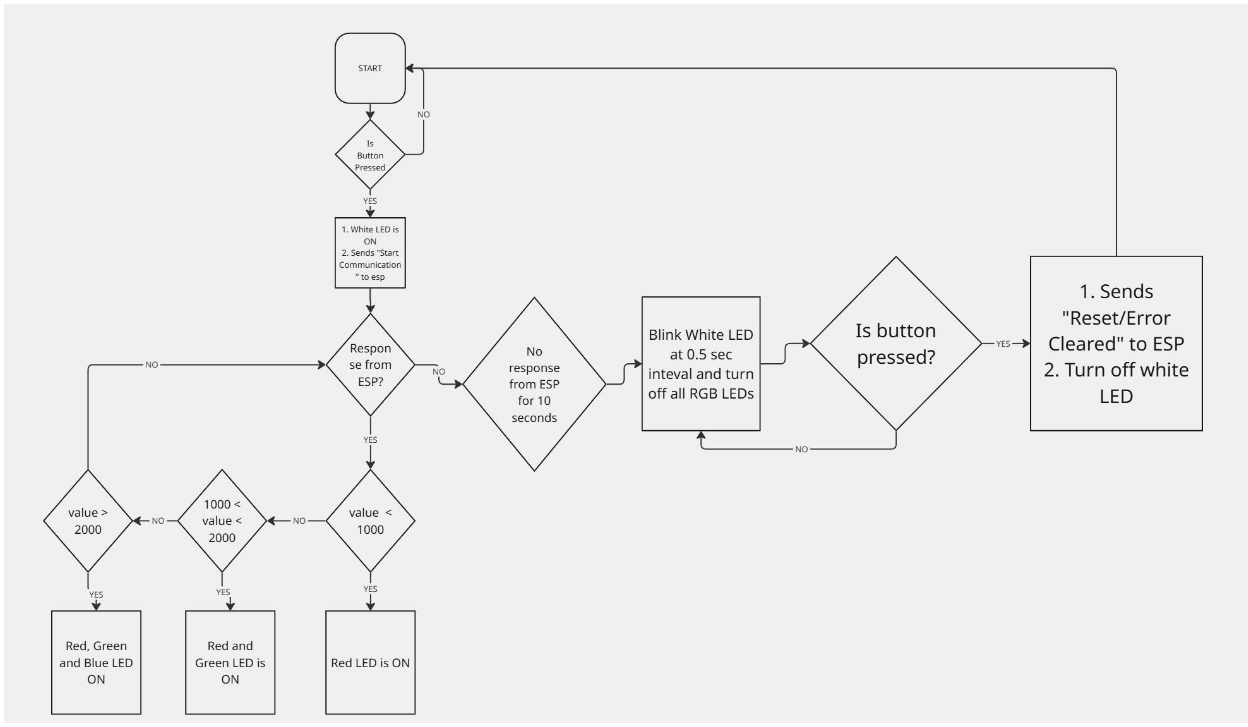
1. Schematic



2. For UDP communication, the ESP32 is configured to send data to the Raspberry Pi at IP 192.168.137.101 on port 8888, while the Raspberry Pi sends commands to the ESP32 at IP 192.168.137.219 on port 9999.
When both programs start, the ESP32 waits for a signal from the Raspberry Pi. Once the user presses the button on the Raspberry Pi, it sends the message “Start Communication” to the ESP32. The ESP32 then begins collecting LDR sensor readings and periodically sends the results back to the Raspberry Pi in the format:
[From 192.168.137.219:9999] Average Light=<value>
where <value> represents the averaged light intensity measurement.

Part 1 - Raspberry Pi WiFi setup and packet delivery

1. Flow chart

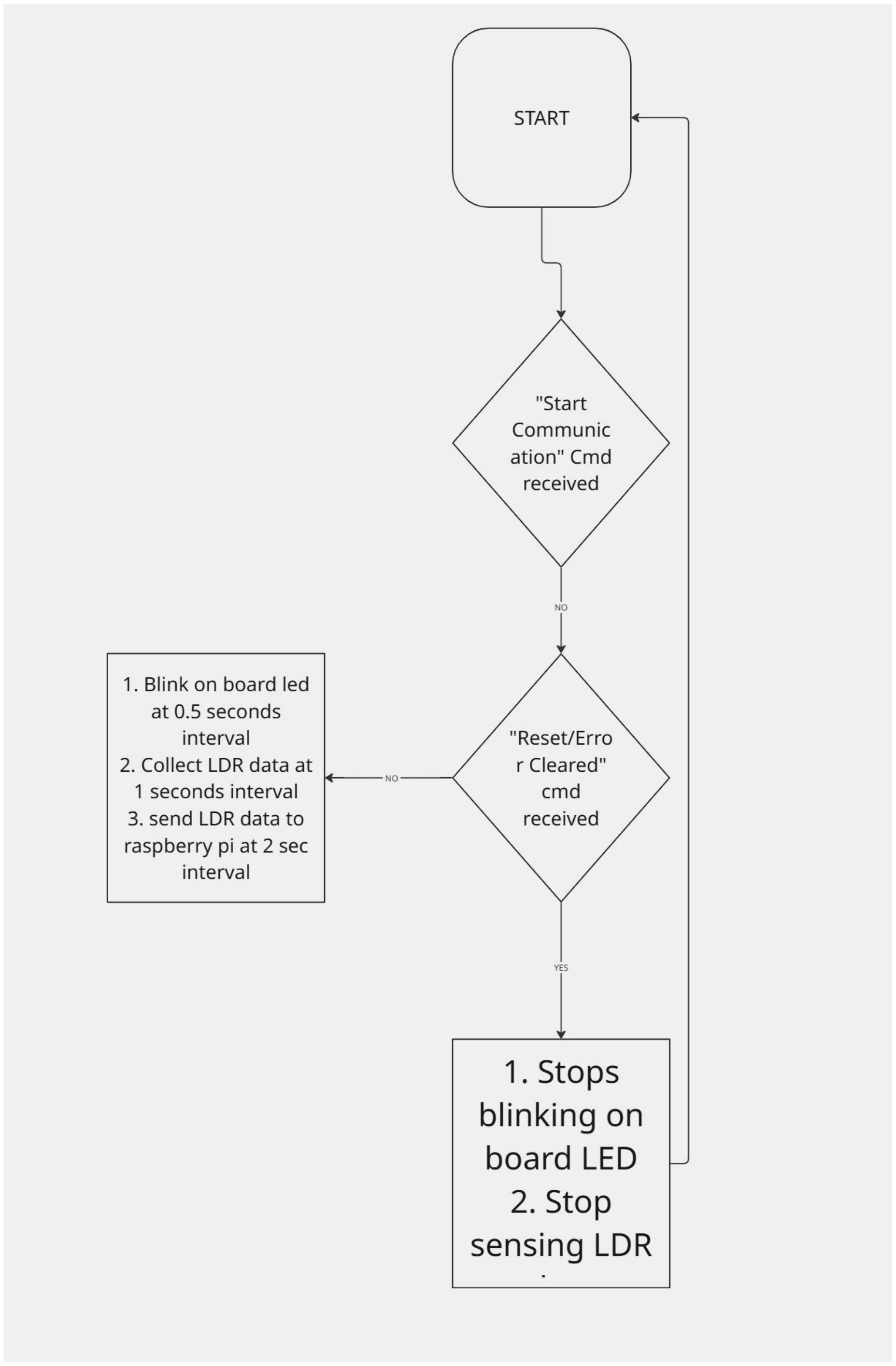

START
NO
Is Button Pressed
YES
1. White LED is ON 2. Sends "Start Communication" to esp
Response from ESP?
NO
No response from ESP for 10 seconds
Blink White LED at 0.5 sec inteval and turn off all RGB LEDs
Is button pressed?
YES
1. Sends "Reset/Error Cleared" to ESP 2. Turn off white LED
NO
NO
YES
value > 2000
NO
1000 < value < 2000
NO
value < 1000
YES
YES
YES
Red, Green and Blue LED ON
Red and Green LED is ON
Red LED is ON

2. The entire setup begins in a waiting state until the button on the Raspberry Pi is pressed. When pressed, the Raspberry Pi sends a “Start Communication” message to the ESP32 and waits for its response. Upon receiving light sensor data from the ESP32, the Raspberry Pi updates the RGB LEDs based on the received values. If no data is received from the ESP32 for 10 seconds, the system enters an error mode where the white LED blinks every 0.5 seconds and all RGB LEDs turn off. Pressing the button again sends a “Reset/Error Cleared” message to the ESP32, stops the blinking LED, resets the system, and returns it to the initial waiting state for the next button press.

3. The Raspberry Pi code enables communication with the ESP32 using UDP. The input functions (on_button_press() and receiver()) handle button presses and receive sensor data from the ESP32. The processing functions (watchdog() and update_rgb_leds()) monitor connection status and determine which LEDs should light up based on the received light values. The output functions (led_control() and update_rgb_leds()) control the main LED for communication status and RGB LEDs to visually represent light intensity levels.

Part 2 - ESP WiFi setup and packet delivery

1. Flow chart


START
"Start Communication" Cmd received
NO
"Reset/Error Cleared" cmd received
NO
1. Blink on board led at 0.5 seconds interval
2. Collect LDR data at 1 seconds interval
3. send LDR data to raspberry pi at 2 sec interval
YES
1. Stops blinking on board LED
2. Stop sensing LDR

2. The ESP32 remains idle until it receives a “Start Communication” command from the Raspberry Pi. Once received, it begins blinking its onboard LED every 0.5 seconds, reads the LDR sensor every 1 second, and sends the average of 5 readings to the Raspberry Pi every 2 seconds. If the ESP32 receives a “Reset/Error Cleared” command, it turns off the onboard LED, stops all operations, and waits for the next “Start Communication” command to resume.

3. The ESP32 code establishes a Wi-Fi and UDP connection with the Raspberry Pi to exchange sensor data and commands.
   The input functions (udp.parsePacket() and udp.read()) receive UDP commands from the Raspberry Pi, such as “Start Communication” or “Reset/Error Cleared.”
   The processing functions (loop(), sampling logic, and averaging) read LDR sensor values periodically, store recent samples, and compute their average.
   The output functions (udp.beginPacket()/udp.print()/udp.endPacket() and digitalWrite(LED_PIN, …)) send the average light values back to the Raspberry Pi and control the onboard LED to indicate active operation.