ASSIGNMENT – 2

IoT System and Software

Vishwanath Singh

73975792

FINAL VERSION

DESCRIPTION OF ESP8266 CODE:

For the ESP code, we have an initial state which is waiting to receive a UDP message from the raspberry pi which helps determine the next state.

We have a init function where there is a function that is called to receive a message from the raspberry pi. The next state is basically the running state. Once the ESP has received a message from the raspberry pi it starts by initializing the variables and the array for the light sensor readings and also alternates turning on and off the LED while collecting the sensor readings. Then it averages the values of the sensor readings which are particularly five sensor readings and sends it to raspberry pi why are you DP and then continues to collect data and send it while in running state the output of this is that it converts a flute data into character array and sends it to the raspberry pie using UDP.

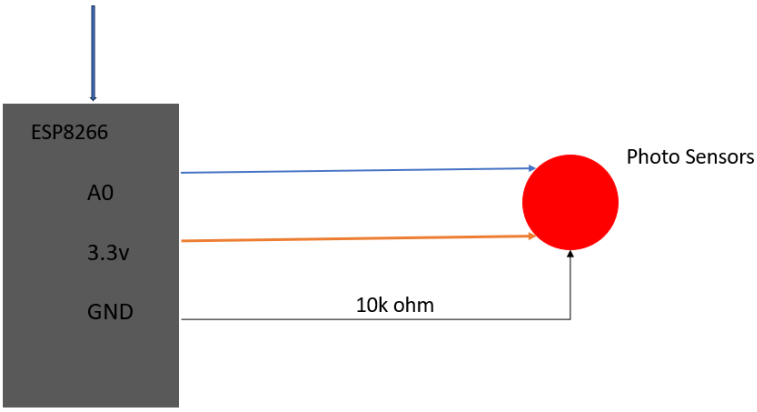

Figure 1: Wiring connection ESP8266 and the photoresistor.

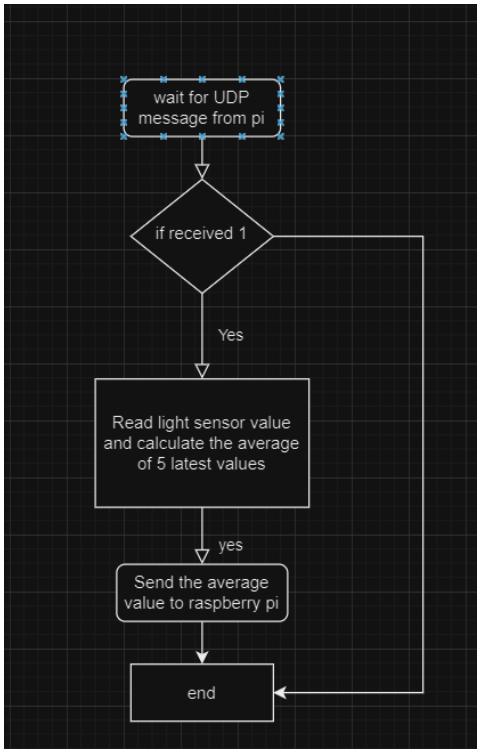

Figure 1: Flowchart of the 3 main states of the ESP8266.

Input:

Listening to the port for UDP packets. Parse the incoming packet to determine the next steps. Stores the IP address and port of the Raspberry Pi for future communication.

Process:

Collects sensor readings from a photoresistor. Calculates the average of the 5 readings taken every second, while flashing on board LED at 0.5sec rate. Sends the average to the Raspberry Pi via UDP.

Output:

Converts the float data to a character array. Sends the data to the Raspberry Pi using UDP.

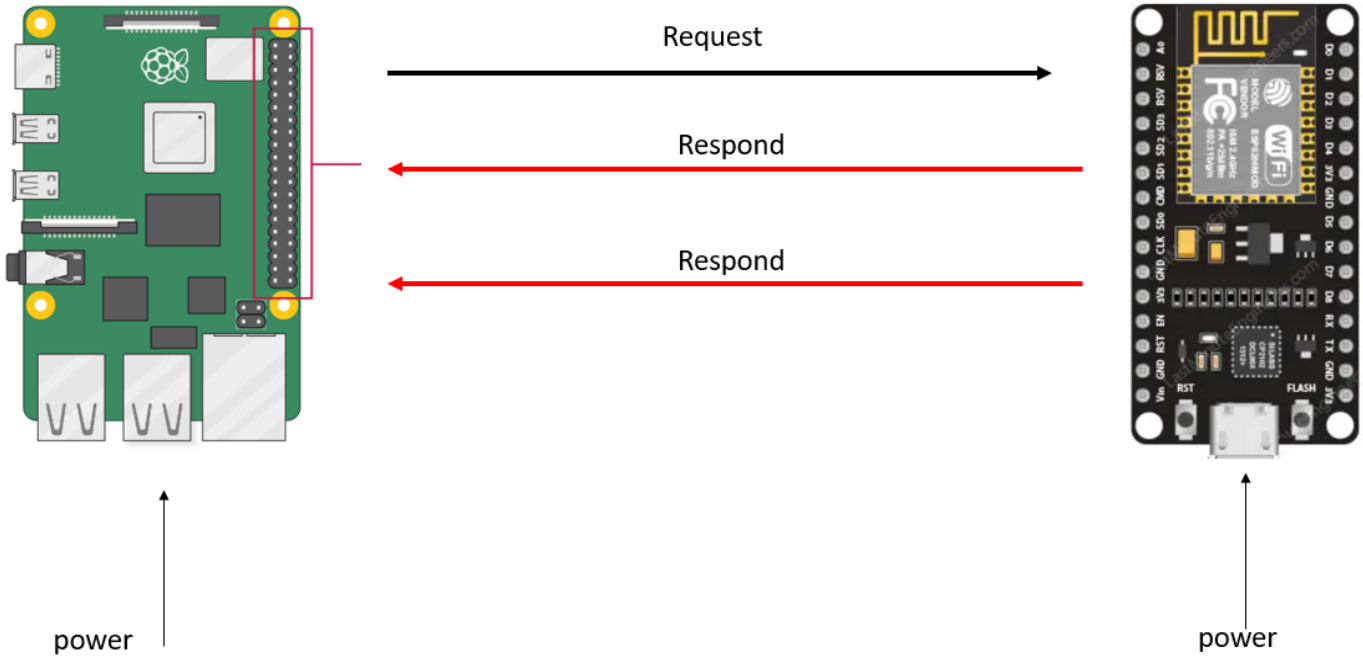

Figure 4: UDP communication between Rpi and ESP, Rpi request data from ESP.

DESCRIPTION OF RASPBERRY PI CODE:

For the raspberry pi code, we initially start by configuring the pins for all the four LEDs and pushbutton. We declare a function, in which we basically initialize all of the pins and the button. Then we have a function which basically takes in the input from the user, which is basically a push of a button. Inside that function when the button is pressed for the first time, a message is sent to ESP to start the communication and if the button is pressed another time, it will send a different message that will stop the communication.

The next function is to send a message from raspberry pi to ESP. Then we have a receive function which takes data that the ESP sends to raspberry pi and suppose if raspberry pi stops receiving data (connection is interrupted) when it is expected to receive data, it will wait for 10 seconds before flashing a white LED multiple time.

We have a function for flashing the LED, in the case that I had mentioned above. The next function is to turn on the LED based on threshold values. We have three threshold values, high medium and low. When the sensor value goes above Low 1 LED is turned on when the sensor value goes above the medium to 2 LEDs are turned on sensor value goes above High three, LEDs are turned on. This is the code for raspberry pie along with the description of all its functions.

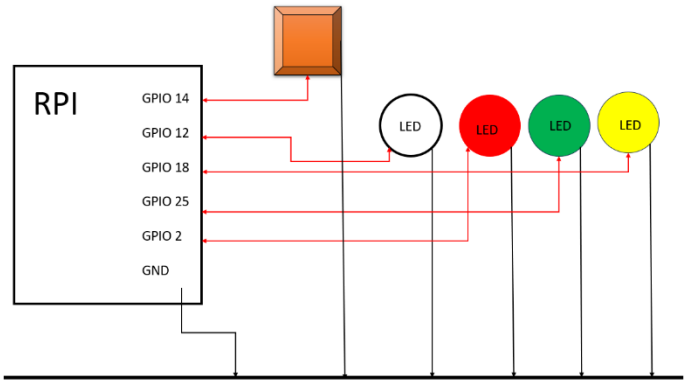

Figure 4: Wiring connection for raspberry pi 4 and white, green, red, and yellow LED.

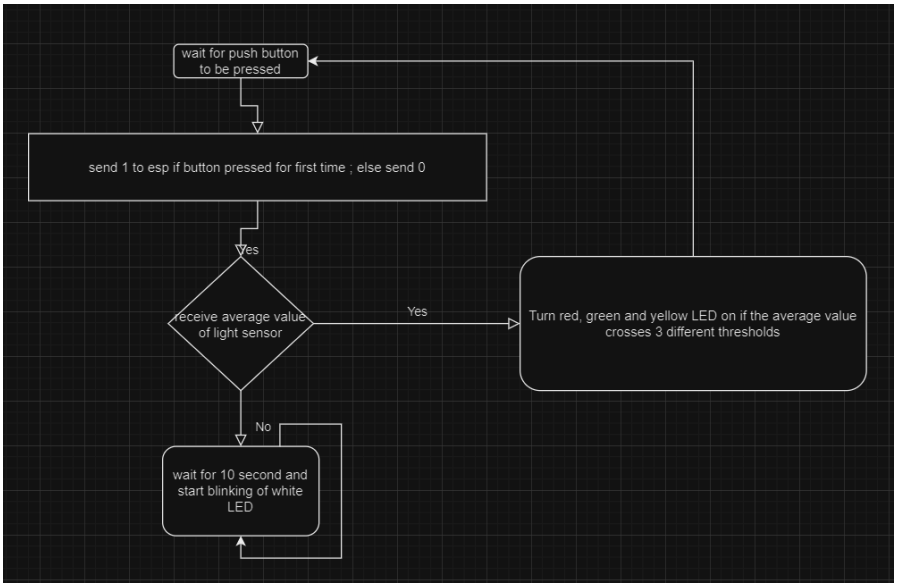

Figure 5: Flowchart of the states and transition for Raspberry pi.

Input:

Monitors if the button is pressed. Initiates messages ("1" or "0") based on button presses.

Process:

Sends the message to ESP once it has detected a button press. Listens to incoming packet from ESP. Using value of light, checks how many LED need to be turned on. If the communication is interrupted at any time, the pi waits for 10 seconds, after which it starts to flash the white LED until the communication starts again.

Output:

GPIO pins are controlled to turn on/off LEDs (white, red, yellow, green) based on light sensor reading sent from ESP8266.

VIDEO: **https://drive.google.com/file/d/14HadSISJ-XW4apCbrPe9l7Vy6LqBFcI3/view?usp=sharing**