EMBEDDED SYSTEM WITH IOT

EXPERIMENT NO. 1

IoT based Automated Street Lighting System

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE162P-4/E01

Date of Performance: 05/07/2025
Date of Submission: 05/09/2025

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

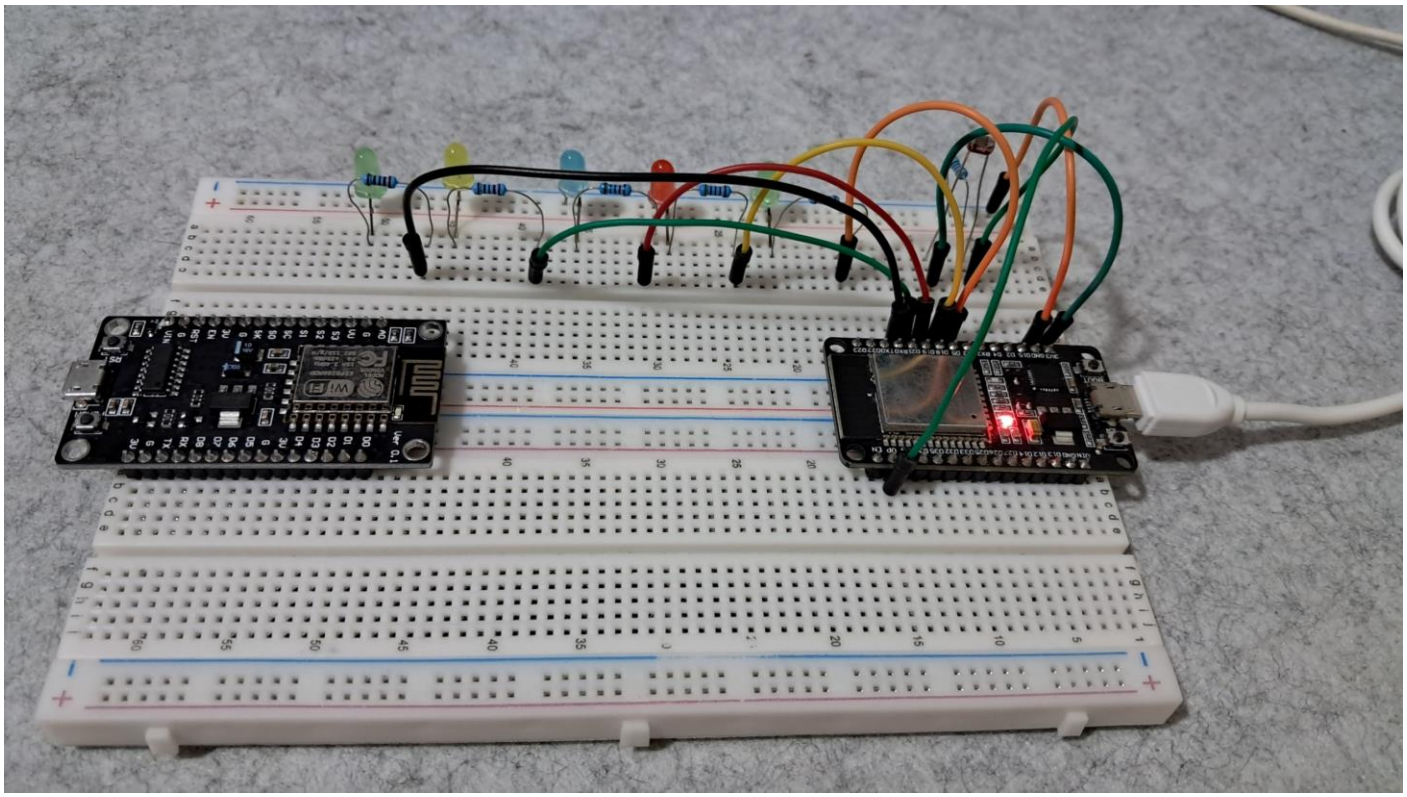

FIGURE 1: THE CONNECTION OF THE LEDS, AND PHOTORESISTOR SENSOR TO THE ESP32 DEV MODULE

This experiment is about building an automated street light system using an IoT device called the ESP32 dev module. The esp32 is a low-cost, low-power microcontroller with built-in Wi-Fi and Bluetooth, making it ideal for IoT projects. its Wi-Fi feature allows it to send data over the internet. The first step in the experiment was to build the physical setup of the street lights and connect all the components properly. The main parts used were a photoresistor sensor, LEDs, and the ESP32. Each LED had a 220-ohm resistor to protect it from receiving too much current. One of the goals of this project was to send data to ThingSpeak for monitoring and analysis. To make the setup more creative, I used different colored LEDs to look like festive street lights during the holiday season. For the automation part,

the photoresistor detects the light level - if it s bright, the LEDs turn off; if it s dark, they turn on automatically.

Navarro_EXP1_CPE162P.ino

```
#include <WiFi.h>
#include <ThingSpeak.h>

const char* ssid = "PLDTHOMEFIBRt6ucX";
const char* password = "PLDTWIFItLK72";

const int ldrPin = A0;
const int Green1 = 5;
const int Red1 = 18;
const int Blue1 = 19;
const int Yellow1 = 21;
const int Green2 = 3;

//LED status variables
int Green1Status = 0;
int Red1Status = 0;
int Blue1Status = 0;
int Yellow1Status = 0;
int Green2Status = 0;

// ThingSpeak setup
WiFiClient client;
long myChannelNumber = 2954171 ;
const char myWriteAPIKey[] = "7U4W9J6MTU110C7S";
```

FIGURE 2: THE INITIALIZATION OF VARIABLES AND COMPONENTS

In the beginning of the code, I declared the variables needed for the experiment. These include the Wi-Fi name and password so the ESP32 can connect to the internet, the pin numbers for the photoresistor or light dependent resistor (LDR) and the LEDs, and status variables to check if each LED is on or off. I also included the details for ThingSpeak, like the channel number and API key, so that the ESP32 can send data to the online platform. This part sets the foundation for how the system will behave and connect to the network.

Navarro_EXP1_CPE162P.ino

```
26  void setup() {
27    Serial.begin(9600);
28    delay(1000);
29
30    pinMode(ldrPin, INPUT);
31    pinMode(Green1, OUTPUT);
32    pinMode(Red1, OUTPUT);
33    pinMode(Blue1, OUTPUT);
34    pinMode(Yellow1, OUTPUT);
35    pinMode(Green2, OUTPUT);
36
37    ledClear();
38
39    Serial.println();
40    Serial.print("Connecting to ");
41    Serial.println(ssid);
42    WiFi.begin(ssid, password);
43    while (WiFi.status() != WL_CONNECTED){
44      delay(500);
45      Serial.print(".");
46    }
47    Serial.println();
48    Serial.println("WiFi connected");
49    Serial.print("IP address: ");
50    Serial.println(WiFi.localIP());
51
52    // Start ThingSpeak
53    ThingSpeak.begin(client);
54  }
```

FIGURE 3: THE SETUP OF THE COMPONENTS

Inside the setup() function, I started by opening the serial monitor using Serial.begin(9600) so I can see outputs while testing. then I used pinMode() to set each pin as either input or output. The LDR is an input because it reads light levels, while the LEDs are outputs because we control them. I also added ledClear() to make sure all the LEDs are off at the start. Then, I connected the ESP32 to the wi-fi using WiFi.begin() and printed the IP address to confirm the connection. In addition, I initialized ThingSpeak so it s ready to send data.

Navarro_EXP1_CPE162P.ino

```
56  void loop() {
57    int ldrValue = analogRead(ldrPin);
58    Serial.print("LDR Value: ");
59    Serial.println(ldrValue);
60
61    // Control LEDs based on LDR value
62    if(ldrValue <= 500){
63      ledLights();
64      Serial.println("LEDs are ON");
65    }else {
66      ledClear();
67      Serial.println("LEDs are OFF");
68    }
69
70    Green1Status = digitalRead(Green1);
71    Red1Status = digitalRead(Red1);
72    Blue1Status = digitalRead(Blue1);
73    Yellow1Status = digitalRead(Yellow1);
74    Green2Status = digitalRead(Green2);
75
76    ThingSpeak.setField(1, ldrValue);
77    ThingSpeak.setField(2, Green1Status);
78    ThingSpeak.setField(3, Red1Status);
79    ThingSpeak.setField(4, Blue1Status);
80    ThingSpeak.setField(5, Yellow1Status);
81    ThingSpeak.setField(6, Green2Status);
82
83    int statusCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
84    if(statusCode == 200){
85      Serial.println("Channel update successful.");
86    }else {
87      Serial.print("ThingSpeak update error. Code: ");
88      Serial.println(statusCode);
89    }
90    delay(15000);
```

```
94  // Turn all LEDs OFF
95  void ledClear(){
96    digitalWrite(Green1, LOW);
97    digitalWrite(Red1, LOW);
98    digitalWrite(Blue1, LOW);
99    digitalWrite(Yellow1, LOW);
100   digitalWrite(Green2, LOW);
101 }
102
103 // Turb all LEDs ON
104 void ledLights(){
105   digitalWrite(Green1, HIGH);
106   digitalWrite(Red1, HIGH);
107   digitalWrite(Blue1, HIGH);
108   digitalWrite(Yellow1, HIGH);
109   digitalWrite(Green2, HIGH);
110 }
```

FIGURE 4: LOOP FUNCTION AND OTHER NECESSARY FUNCTIONS

The loop() function is the heart of the automation. it starts by reading the LDR value using analogRead() and prints it to the serial monitor. If the LDR value is low (dark), the LEDSs turn on by calling ledLights(). If it's bright, the LEDs turn off using ledClear(). then, the code checks which LEDs are currently on or off and saves that information into the status variables. Finally, it sends all this data including the LDR value and led status to ThingSpeak every 15 seconds. This allows us to monitor and analyze the system in real-time.

Navarro_EXP1_CPE162P.ino

```
#include <WiFi.h>
#include <ThingSpeak.h>

const char* ssid = "PLDTHOMEFIBRt6ucX";
const char* password = "PLDTWIFItLK72";

const int ldrPin = A0;
const int Green1 = 5;
const int Red1 = 18;
const int Blue1 = 19;
const int Yellow1 = 21;
const int Green2 = 3;

//LED status variables
int Green1Status = 0;
int Red1Status = 0;
int Blue1Status = 0;
int Yellow1Status = 0;
int Green2Status = 0;
```

```
// Control LEDs based on LDR value
if(ldrValue <= 500){
  ledLights();
  Serial.println("LEDs are ON");
}else {
  ledClear();
  Serial.println("LEDs are OFF");
}
```

FIGURE 5: FIXING THE ERRORS

While testing the code, one issue I encountered was incorrect LED behavior. At first, some LEDs didn't light up because I forgot to match the physical wiring with the pin numbers in the code. After double-checking both the code and the breadboard, I fixed the pin assignments. I also noticed I had a typo in a comment (turb instead of turn) which I corrected to make the code easier to understand.

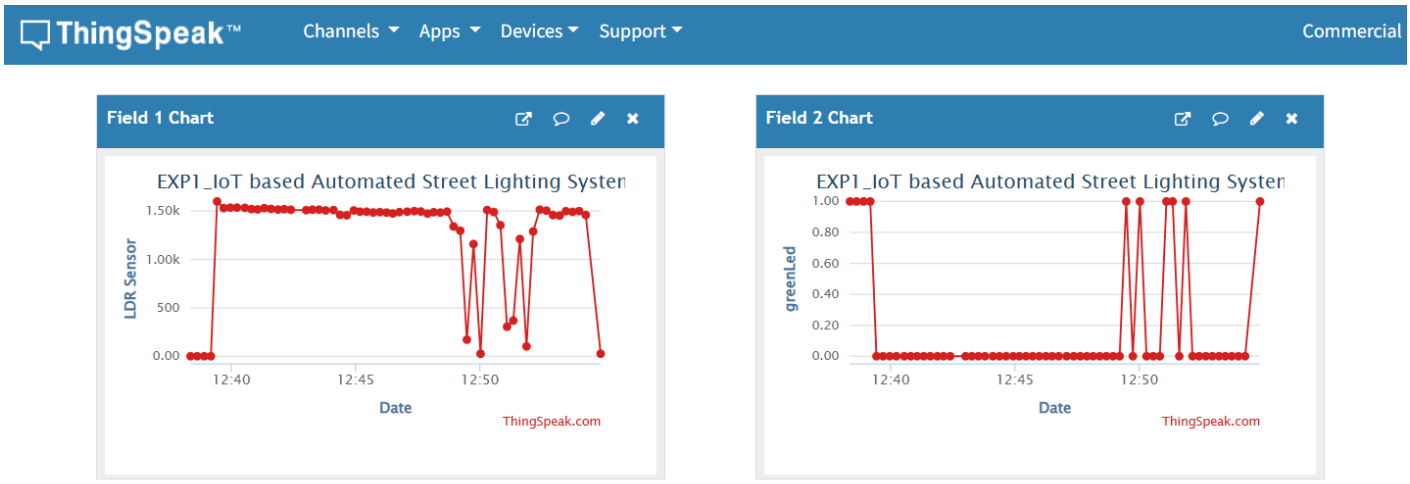

FIGURE 6: THINGSPEAK GRAPH INTERPRETATION: LDR SENSOR AND GREEN LED GRAPHS

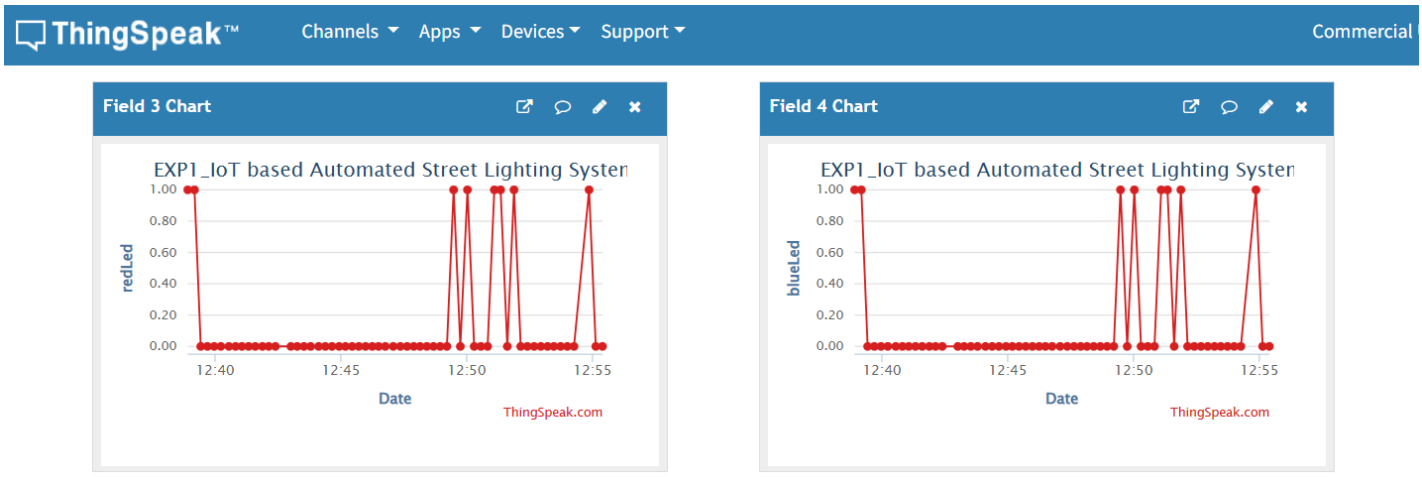

FIGURE 7: THINGSPEAK GRAPH INTERPRETATION: RED LED AND BLUE LED GRAPHS

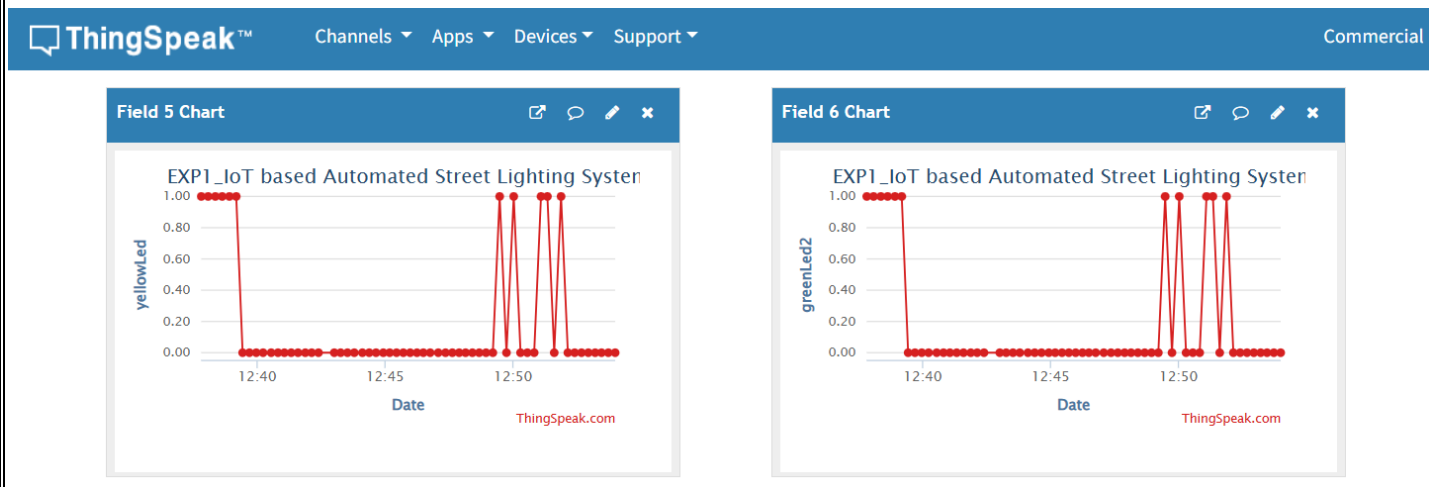

FIGURE 8: THINGSPEAK GRAPH INTERPRETATION: YELLOW LED AND GREEN2 LED GRAPHS

In ThingSpeak IoT, I was able to view graphs in my channel showing the LDR values and LED status over time. When it was dark, the LDR values dropped below 500, and I saw that all the LED fields (green, red, blue, yellow, and green2) showed a value of 1, meaning they were on. When there was light, the LDR values increased, and the LED fields showed 0, meaning the LEDs were off. These graphs helped me clearly see that the automation was working and that the system was responding correctly to changes in light. For future improvements of the system it could also notify the user if some of the lights on the street are not working so that they can take action immediately.

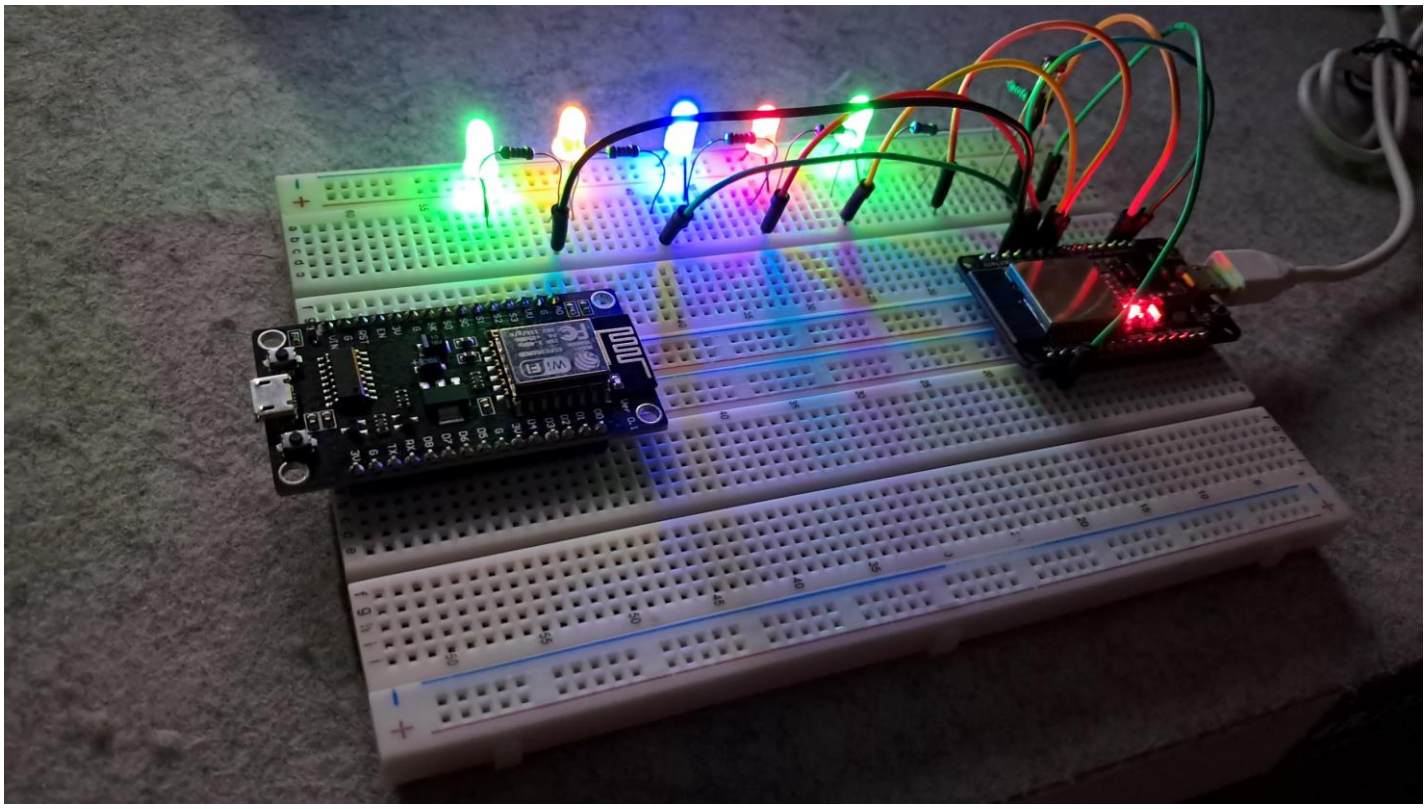

FIGURE 9: WORKING PROTOTYPE: ALL LEDS TURNED ON - LDR SENSOR VALUES LESS THAN OR EQUAL TO 500

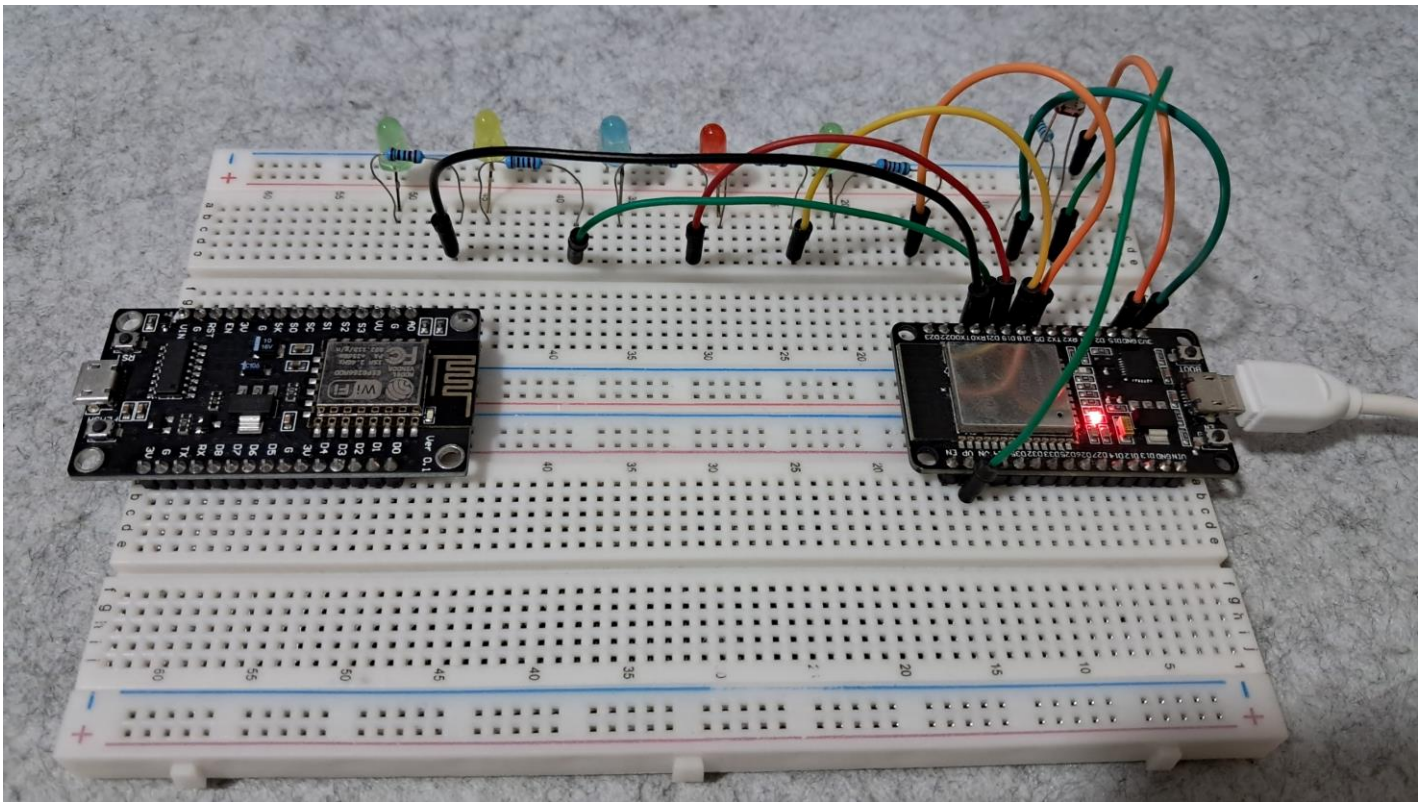

FIGURE 10: WORKING PROTOTYPE: ALL LEDS TURNED OFF - LDR SENSOR VALUES GREATER THAN 500

This is the final working prototype of the IoT based automated street lighting system experiment, which successfully meets all the requirements.

CONCLUSION

In this experiment, I successfully built an IoT-based automated street lighting system using the ESP32 dev module. by combining LEDs, a photoresistor sensor, and internet connectivity through wi-fi, I was able to create a system that automatically turns the lights on when it s dark and off when it s bright. I also connected the project to ThingSpeak, where I could monitor real-time data and graphs that showed the status of each led and the light level from the sensor. I added colorful LEDs to make it look more festive, and I was happy to see that the automation worked just as expected.

While doing this project, I learned how important it is to double-check wiring and code, especially the pin numbers. I made a few mistakes at first, but I was able to fix them and improve my code. I also understood how IoT platforms like ThingSpeak can help in monitoring systems remotely. this project helped me apply what I learned in class to a real-world example, and I believe it can be improved further by adding features like alerts when a light is not working.