Plant Wisperer

Monitor your plant health on the go!

॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS
INSTITUTE OF TECHNOLOGY, PUNE

TinkerLab Project

Group 7

AY 2024-25 Semester II

Batch 2024-28

Sr No	Name	PRN
1	Avishi Poddar	24070123026
2	Ananya Swadia	24070123012
3	Shejal Singh	24070123152
4	Anavi Kashyap	24070123013
5	Armaan Majhi	24070123022

Problem Statement

Plants require specific environmental conditions to thrive, including adequate light, temperature, humidity, and soil moisture. However, maintaining these conditions manually can be difficult, especially for individuals with limited gardening experience or busy lifestyles. The "Plant Whisperer" addresses this challenge by continuously monitoring key environmental factors and providing real-time information. This system aims to simplify plant care, prevent over- or under-watering, and promote better growth by giving users actionable insights into their plant’s environment.

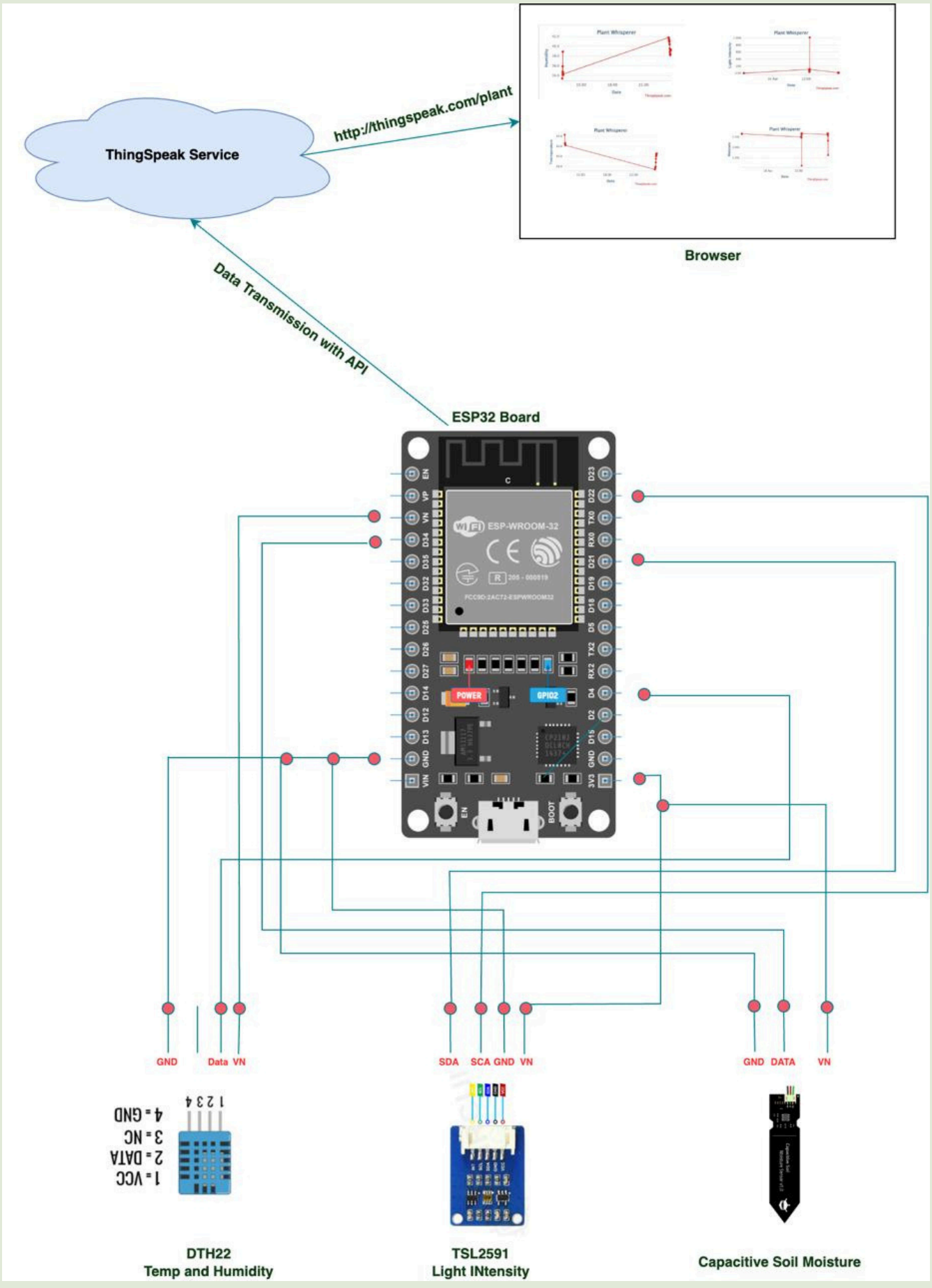
Introduction

The cultivation and maintenance of healthy plants require the careful regulation of several environmental parameters, including light intensity, temperature, humidity, and soil moisture. Inconsistent monitoring and inadequate responses to fluctuations in these conditions often result in suboptimal plant health, particularly in domestic and urban settings where natural conditions may not be ideal. Furthermore, individuals may lack the expertise or time required to manage these variables effectively. To address these challenges, this project proposes the development of the "Plant Whisperer," a smart plant monitoring system designed to measure and report critical environmental factors in real-time. By providing continuous feedback and data-driven insights, the system aims to support users in maintaining optimal conditions for plant growth, thereby enhancing the ease and efficiency of plant care.

In recent years, there has been a growing interest in integrating technology into sustainable and efficient plant care, driven by advancements in sensors, microcontrollers, and Internet of Things (IoT) applications. Smart gardening systems have emerged as practical tools to assist individuals in managing plant health, particularly in environments where traditional gardening practices are less feasible. These systems not only aid hobbyists and urban dwellers, but also have potential applications in agriculture.

Ultimately, this project seeks to contribute to the broader goal of using technology to promote healthier, more sustainable plant care practices

System Component and Wiring Diagram



Components

1. ESP 32 dev module board-

- A powerful Wi-Fi and Bluetooth-enabled microcontroller board.
- Used for IoT projects due to its wireless capabilities and multiple GPIO pins.
- Has a dual-core processor, making it faster and more efficient than many other microcontrollers.

Why ESP 32 -

- wifi enabled -Built-in Wi-Fi and Bluetooth make it perfect for wireless communication.
- Low power consumption, good for battery-powered projects.
- Supports various sensors and modules easily.

Microcontroller of ESP32

- Based on the Tensilica Xtensa LX6 (or LX7) dual-core processor.
- 32-bit MCU with high speed (up to 240 MHz).
- Handles real-time tasks, sensor data, and internet connectivity all in one chip.

2. DHT22 Temperature and Humidity Sensor

- Measures temperature and humidity accurately.
- Digital output – easy to interface with ESP32.
- Temperature range: -40 to +80°C, Humidity range: 0-100%.

4. TSL2591 Light Intensity Sensor

- Detects ambient light levels (lux).
- High sensitivity – works well in very low to very bright light.
- Outputs digital I2C data for easy connection.

5. Soil Moisture Sensor

- Measures moisture level in soil.
- Helps in smart gardening/irrigation systems.
- Output can be analog or digital depending on the version.

6. Connecting Jumper Wires

- Used to make temporary connections between components and the breadboard.
- Types: Male-to-Male, Male-to-Female, Female-to-Female.

7. Mini Breadboard

- A solderless prototyping board for electronics.
- Allows easy circuit building and testing.
- Reusable and ideal for beginners and prototyping.

ThingSpeak

ThingSpeak is an open-source Internet of Things (IoT) platform that allows you to collect, store, analyze, and visualize data from sensors or devices in real time. It's widely used for DIY projects, research, and prototyping where users want to remotely monitor data or trigger actions based on conditions.

ThingSpeak is a cloud-based platform developed by MathWorks (the same folks behind MATLAB), and it primarily offers:
Data logging in channels (with fields for different sensor readings).
Visualization tools like line graphs, bar charts, and maps.
Data processing using MATLAB code directly in the platform.
Trigger actions, such as sending alerts or webhooks.
APIs (RESTful) for reading/writing data.

🚀 Why Use **ThingSpeak** with an ESP32?
The ESP32 is a powerful microcontroller with built-in Wi-Fi and Bluetooth, making it ideal for IoT applications. Here's why you'd pair it with ThingSpeak:
✅ 1. Remote Data Monitoring
- Use ESP32 to collect data from sensors (like temperature, humidity, motion, etc.).
- Send that data to ThingSpeak over Wi-Fi.
- View the data live from anywhere via the ThingSpeak web interface.

✅ 2. Data Logging & History
- ThingSpeak stores data in "channels" and keeps historical logs.
- Great for long-term monitoring or tracking trends over time.

✅ 3. Visualization
- Quickly graph your sensor data with no coding required.
- Helpful for debugging or understanding data patterns.

✅ 4. Triggers & Alerts
- Set up rules to trigger actions like sending emails, tweeting, or calling a webhook when certain conditions are met.

✅ 5. Ease of Use
- ThingSpeak is beginner-friendly and integrates well with Arduino IDE, which many ESP32 projects use.
- Supports HTTPS and API keys for security.

🔐 Security Note
- Always secure your ESP32 to ThingSpeak connection using HTTPS and write API keys.
- Avoid exposing your keys in public repositories.

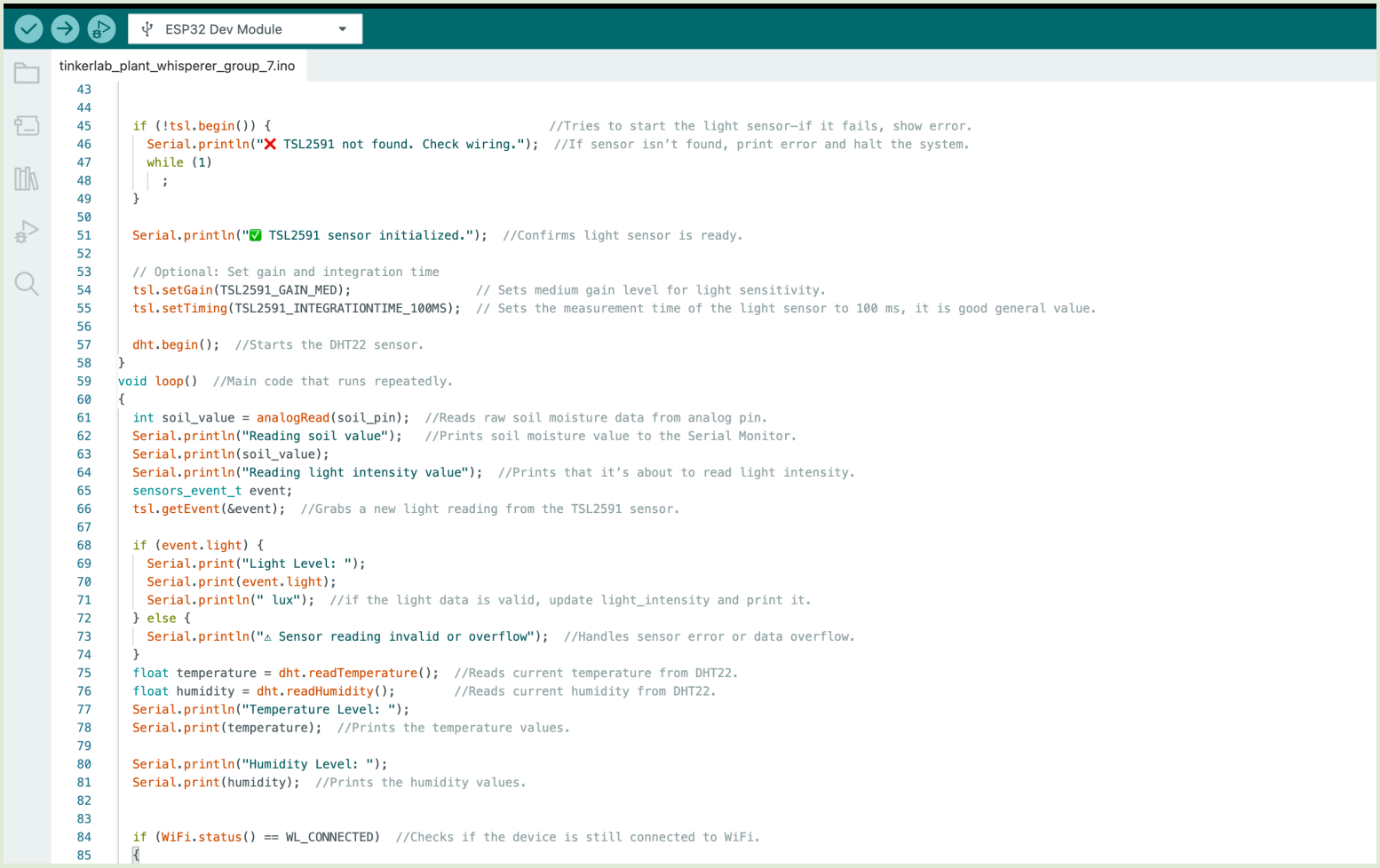
We have used this as ESP32 is wifi enabled and its easy for sending data and displaying it in a proper manner.

Code

ESP32 Dev Module

tinkerlab_plant_whisperer_group_7.ino

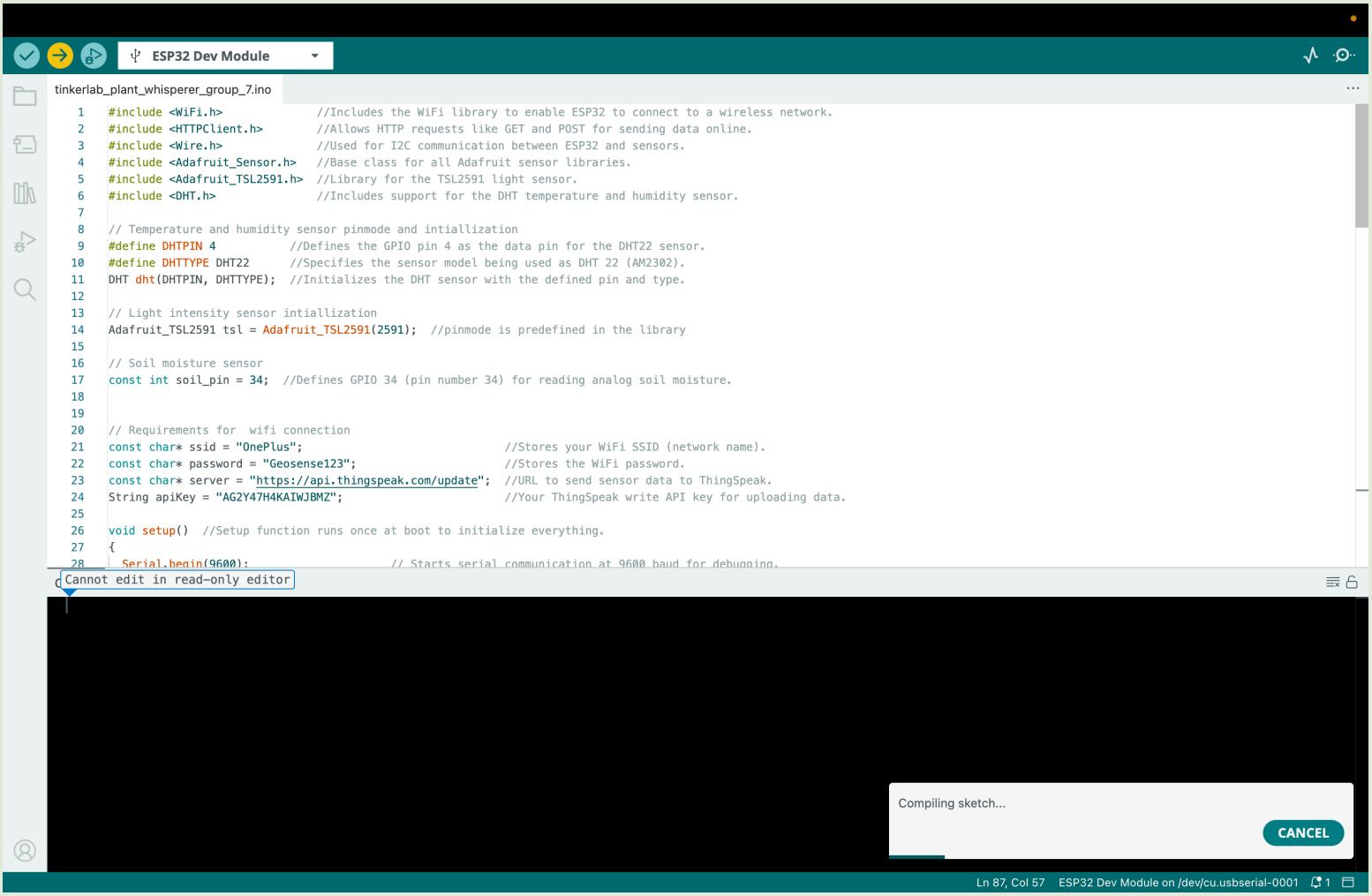
```
#include <WiFi.h>              //Includes the WiFi library to enable ESP32 to connect to a wireless network.
#include <HTTPClient.h>        //Allows HTTP requests like GET and POST for sending data online.
#include <Wire.h>              //Used for I2C communication between ESP32 and sensors.
#include <Adafruit_Sensor.h>   //Base class for all Adafruit sensor libraries.
#include <Adafruit_TSL2591.h>  //Library for the TSL2591 light sensor.
#include <DHT.h>               //Includes support for the DHT temperature and humidity sensor.

// Temperature and humidity sensor pinmode and intiallization
#define DHTPIN 4           //Defines the GPIO pin 4 as the data pin for the DHT22 sensor.
#define DHTTYPE DHT22      //Specifies the sensor model being used as DHT 22 (AM2302).
DHT dht(DHTPIN, DHTTYPE);  //Initializes the DHT sensor with the defined pin and type.

// Light intensity sensor intiallization
Adafruit_TSL2591 tsl = Adafruit_TSL2591(2591);  //pinmode is predefined in the library

// Soil moisture sensor
const int soil_pin = 34;  //Defines GPIO 34 (pin number 34) for reading analog soil moisture.


// Requirements for  wifi connection
const char* ssid = "OnePlus";                              //Stores your WiFi SSID (network name).
const char* password = "Geosense123";                      //Stores the WiFi password.
const char* server = "https://api.thingspeak.com/update";  //URL to send sensor data to ThingSpeak.
String apiKey = "AG2Y47H4KAIWJBMZ";                        //Your ThingSpeak write API key for uploading data.

void setup()  //Setup function runs once at boot to initialize everything.
{
  Serial.begin(9600);                    // Starts serial communication at 9600 baud for debugging.
  WiFi.disconnect(true);                 //Disconnects and resets any previous WiFi connections.
  delay(3000);                           //Waits 3 seconds (3000 milli seconds) before starting WiFi connection.
  WiFi.begin(ssid, password);            //Starts connecting to the specified WiFi network/
  Serial.println("connecting to wifi "); //Prints connection status message.
  while (WiFi.status() != WL_CONNECTED)
    ;  //Keeps looping until the ESP32 is connected to WiFi.
  {
    delay(500);         //Tries connecting to wifi in the time interval of 500 milli seconds.
    Serial.print(".");  //Prints dots and connection status during WiFi attempt.
    Serial.print(WiFi.status());
    Serial.println("connected successfully");  //Confirmation that WiFi is connected.
  }

  pinMode(soil_pin, INPUT);  //Sets soil sensor pin as input for analog readings.

```

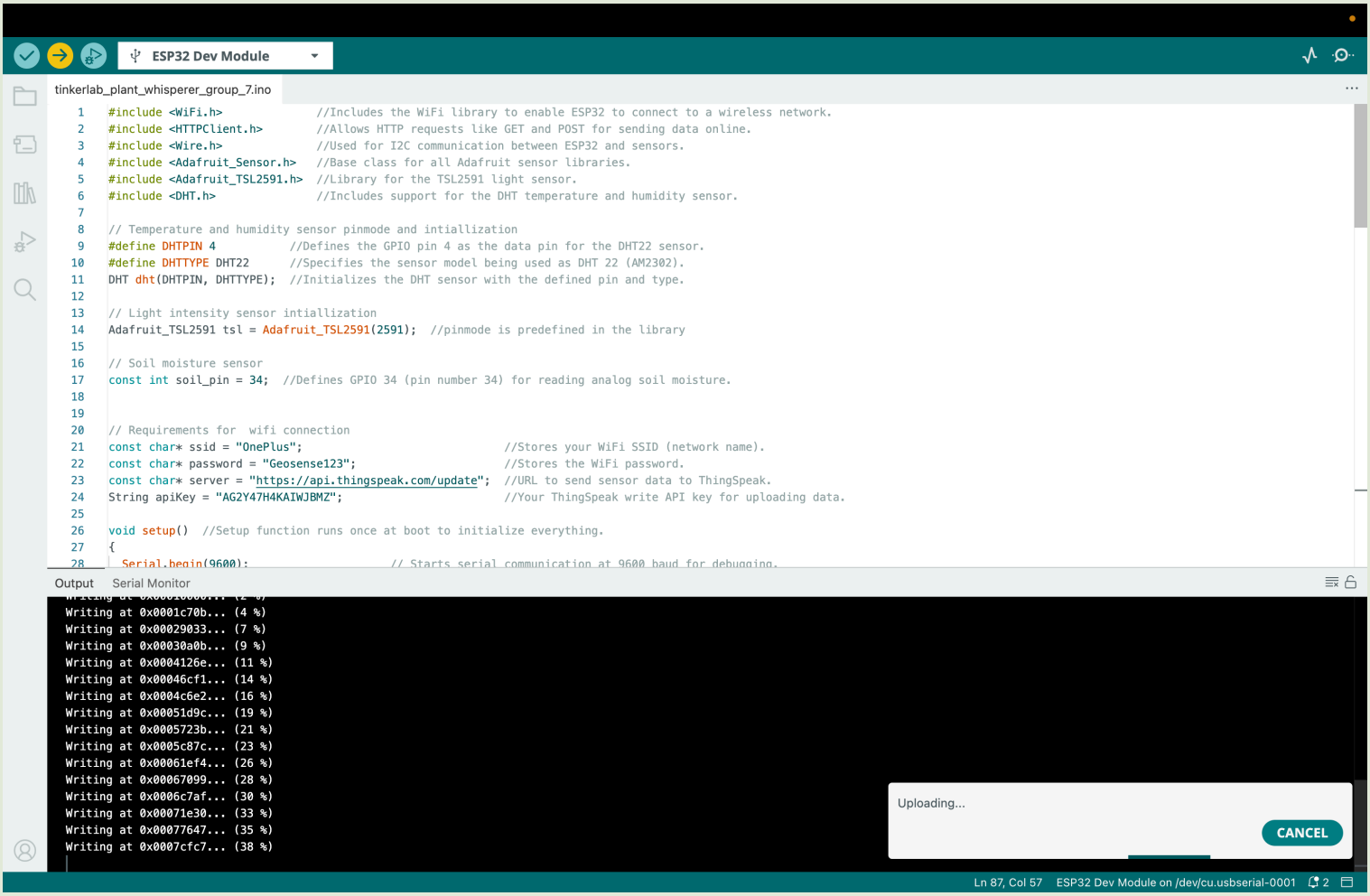


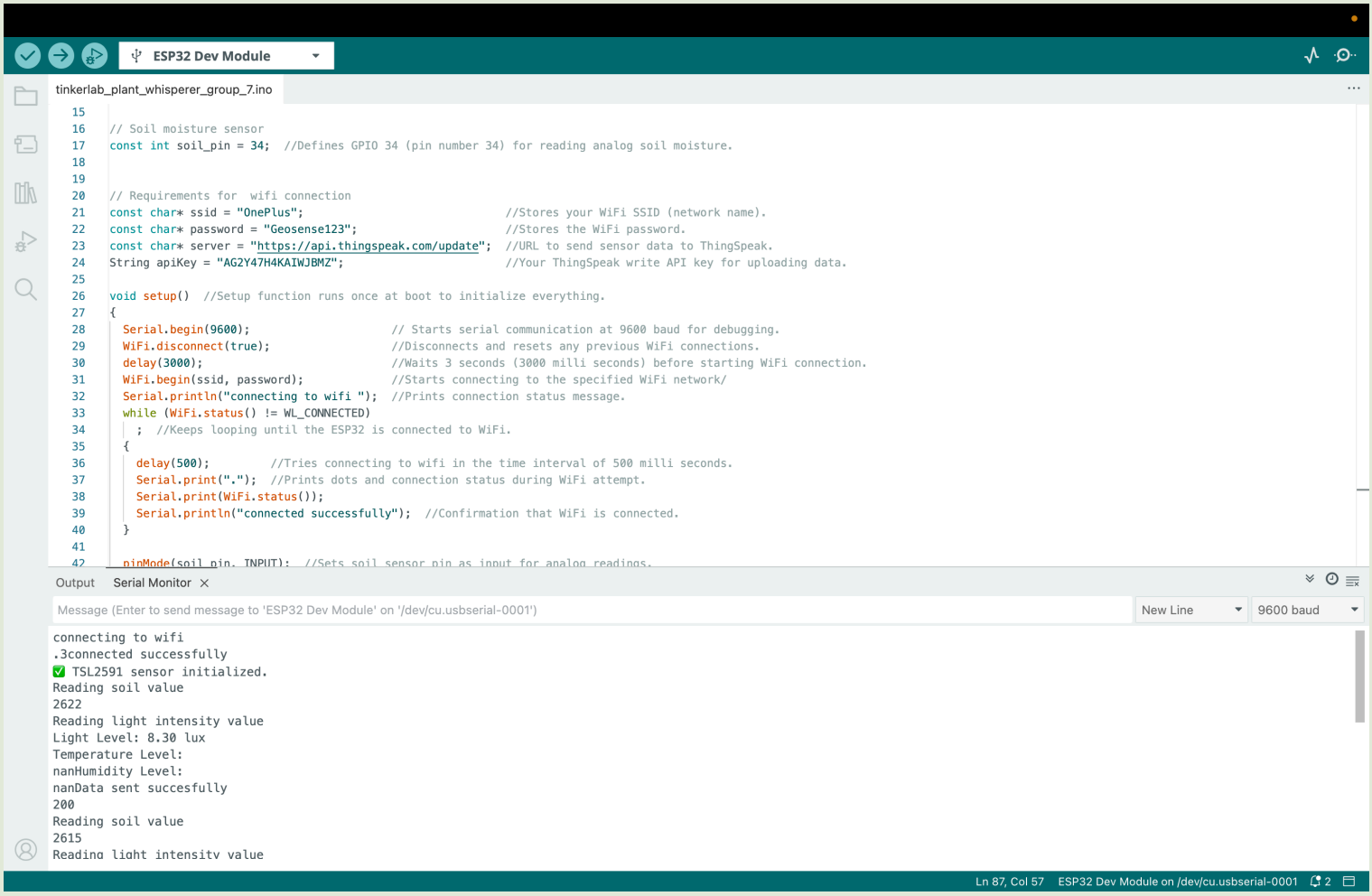

```
43
44
45    if (!tsl.begin()) {                                    //Tries to start the light sensor—if it fails, show error.
46      Serial.println("❌ TSL2591 not found. Check wiring.");  //If sensor isn't found, print error and halt the system.
47      while (1)
48        ;
49    }
50
51    Serial.println("✅ TSL2591 sensor initialized.");  //Confirms light sensor is ready.
52
53    // Optional: Set gain and integration time
54    tsl.setGain(TSL2591_GAIN_MED);                 // Sets medium gain level for light sensitivity.
55    tsl.setTiming(TSL2591_INTEGRATIONTIME_100MS);  // Sets the measurement time of the light sensor to 100 ms, it is good general value.
56
57    dht.begin();  //Starts the DHT22 sensor.
58  }
59  void loop()  //Main code that runs repeatedly.
60  {
61    int soil_value = analogRead(soil_pin);  //Reads raw soil moisture data from analog pin.
62    Serial.println("Reading soil value");   //Prints soil moisture value to the Serial Monitor.
63    Serial.println(soil_value);
64    Serial.println("Reading light intensity value");  //Prints that it's about to read light intensity.
65    sensors_event_t event;
66    tsl.getEvent(&event);  //Grabs a new light reading from the TSL2591 sensor.
67
68    if (event.light) {
69      Serial.print("Light Level: ");
70      Serial.print(event.light);
71      Serial.println(" lux");  //if the light data is valid, update light_intensity and print it.
72    } else {
73      Serial.println("⚠ Sensor reading invalid or overflow");  //Handles sensor error or data overflow.
74    }
75    float temperature = dht.readTemperature();  //Reads current temperature from DHT22.
76    float humidity = dht.readHumidity();        //Reads current humidity from DHT22.
77    Serial.println("Temperature Level: ");
78    Serial.print(temperature);  //Prints the temperature values.
79
80    Serial.println("Humidity Level: ");
81    Serial.print(humidity);  //Prints the humidity values.
82
83
84    if (WiFi.status() == WL_CONNECTED)  //Checks if the device is still connected to WiFi.
85    {
```

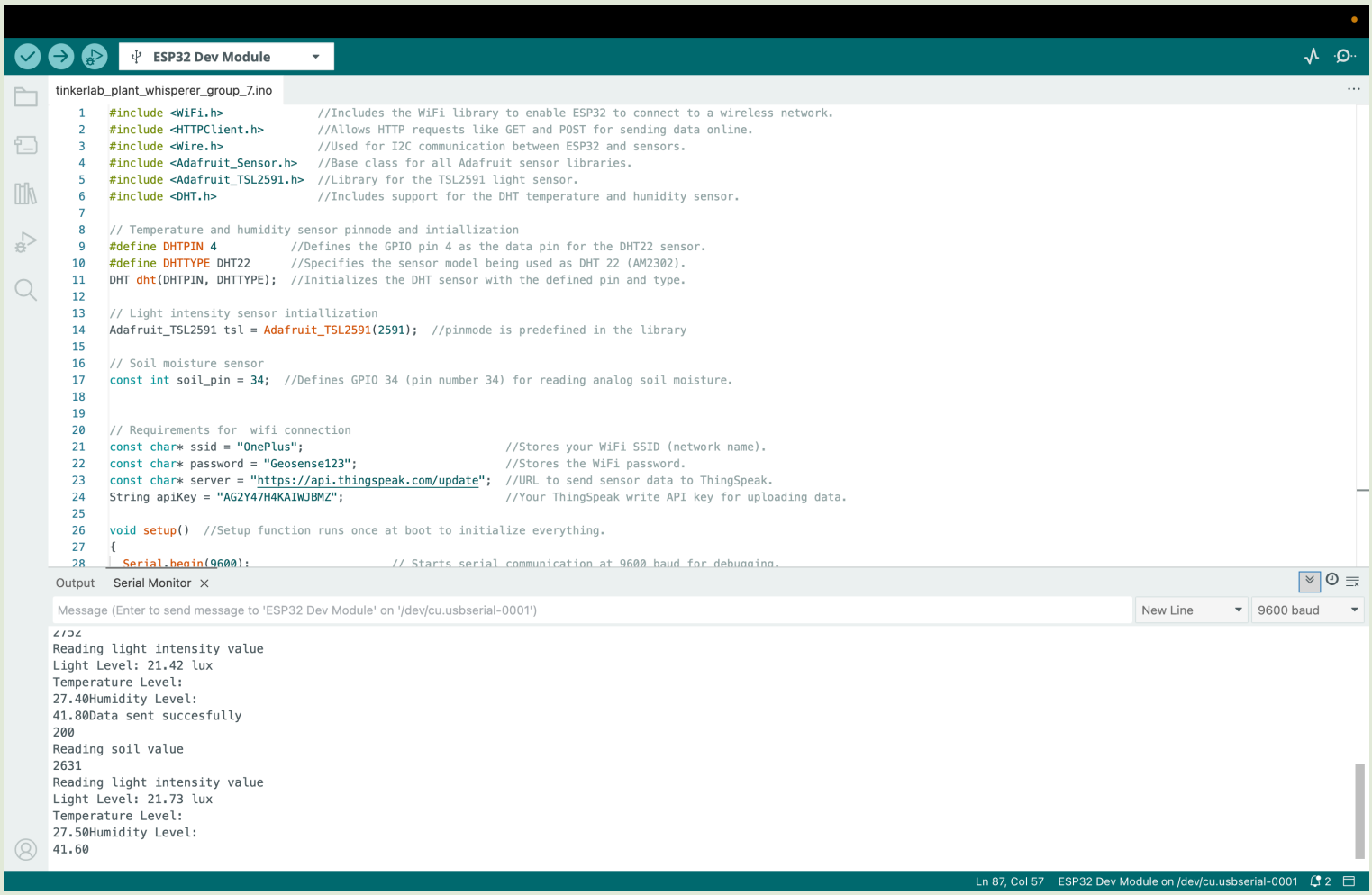
```
85    {
86      HTTPClient http;  //Creates an HTTP client for making requests.
87      //Starts the HTTP request and sends the GET request.
88      String url = String(server) + "?api_key=" + apiKey + "&field1=" + String(temperature) + "&field2=" + String(soil_value) + "&field3=" + String(event.light) + "&field4=" + String(humidit
89      http.begin(url);
90      int httpCode = http.GET();
91      if (httpCode == 200) {
92        Serial.println("Data sent succesfully");
93        Serial.println(httpCode);
94      }  //Checks if data upload was successful or not and prints status.
95      else {
96        Serial.println("Couldn't send data");
97      }
98      http.end();  //Ends the HTTP connection.
99    } else {
100     Serial.println("WiFi was not connected");  //If WiFi failed, print a warning.
101   }
102 }
103
```

Arduino IDE Working

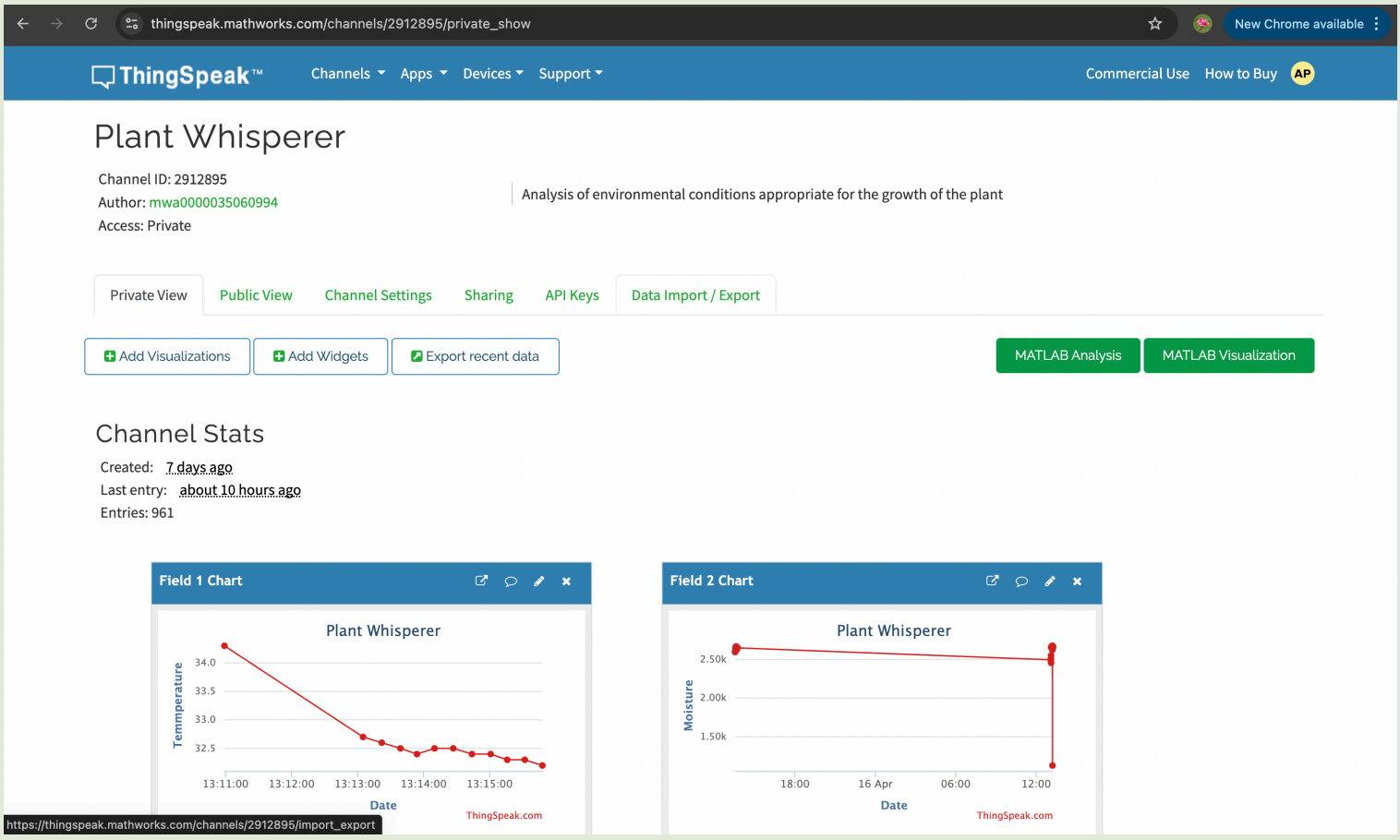


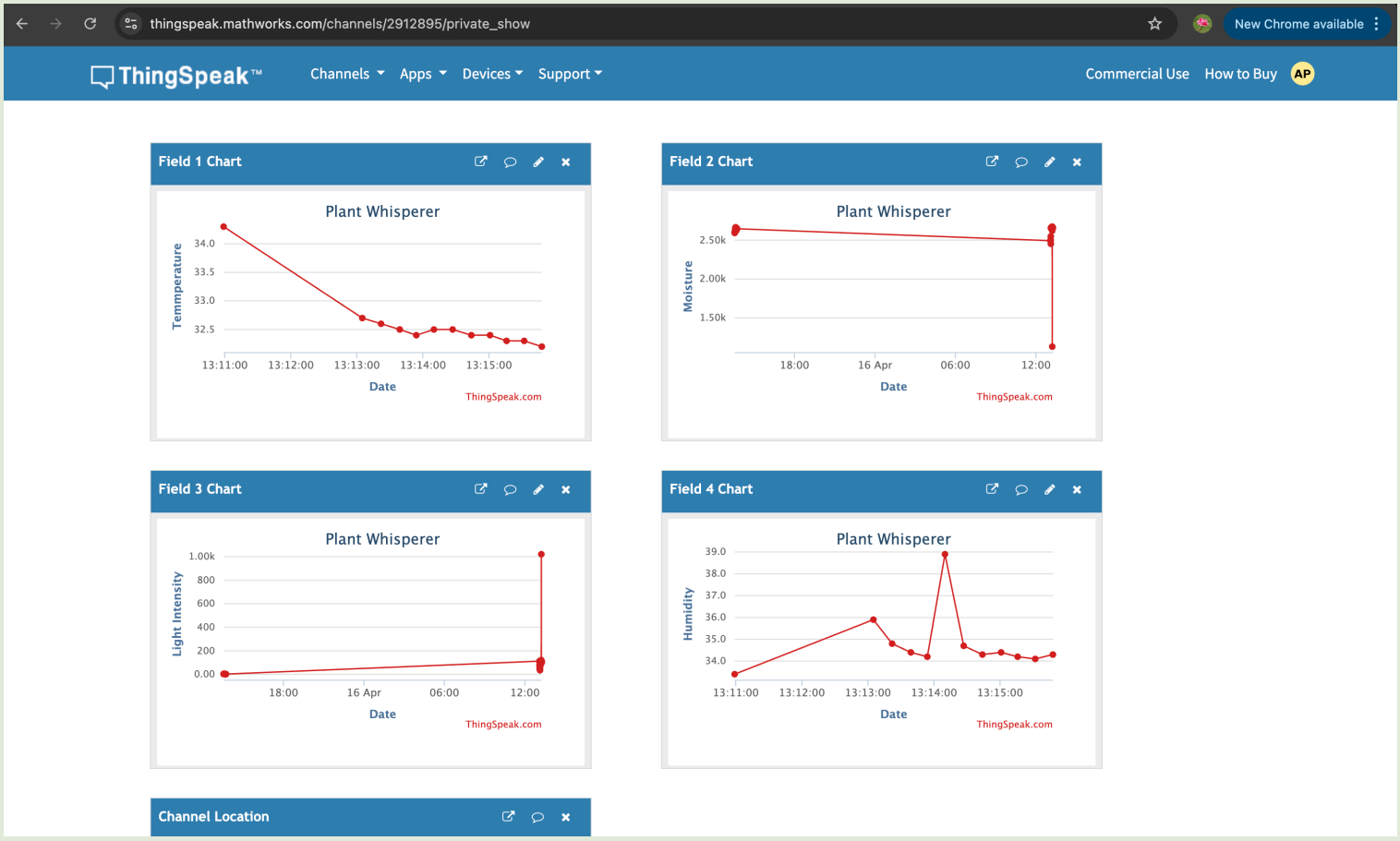


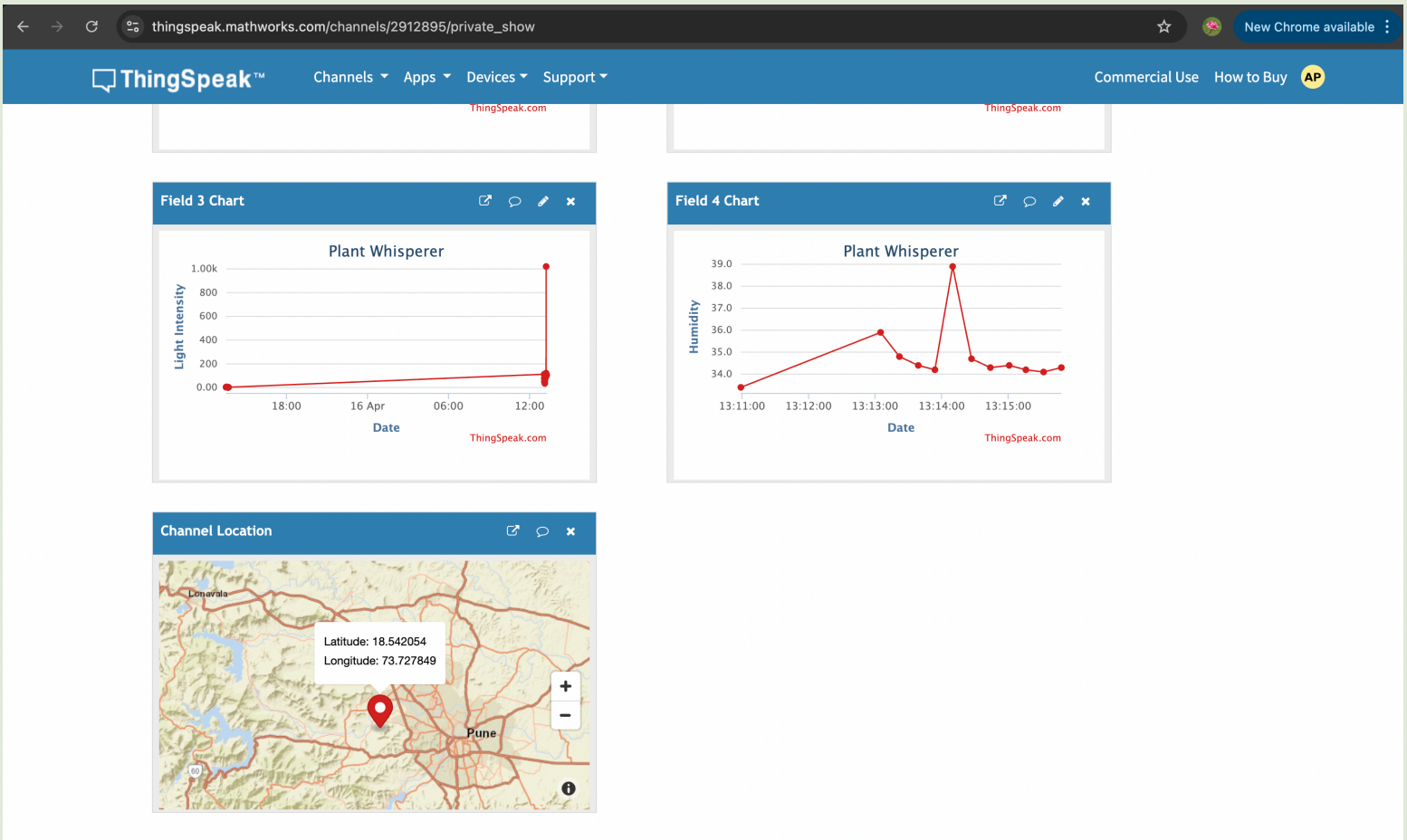

ESP32 Dev Module
tinkerlab_plant_whisperer_group_7.ino
// Soil moisture sensor
const int soil_pin = 34; //Defines GPIO 34 (pin number 34) for reading analog soil moisture.
// Requirements for wifi connection
const char* ssid = "OnePlus"; //Stores your WiFi SSID (network name).
const char* password = "Geosense123"; //Stores the WiFi password.
const char* server = "https://api.thingspeak.com/update"; //URL to send sensor data to ThingSpeak.
String apiKey = "AG2Y47H4KAIWJBMZ"; //Your ThingSpeak write API key for uploading data.
void setup() //Setup function runs once at boot to initialize everything.
{
Serial.begin(9600); // Starts serial communication at 9600 baud for debugging.
WiFi.disconnect(true); //Disconnects and resets any previous WiFi connections.
delay(3000); //Waits 3 seconds (3000 milli seconds) before starting WiFi connection.
WiFi.begin(ssid, password); //Starts connecting to the specified WiFi network/
Serial.println("connecting to wifi "); //Prints connection status message.
while (WiFi.status() != WL_CONNECTED)
; //Keeps looping until the ESP32 is connected to WiFi.
{
delay(500); //Tries connecting to wifi in the time interval of 500 milli seconds.
Serial.print("."); //Prints dots and connection status during WiFi attempt.
Serial.print(WiFi.status());
Serial.println("connected successfully"); //Confirmation that WiFi is connected.
}
pinMode(soil_pin, INPUT); //Sets soil sensor pin as input for analog readings.
Output
Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')
New Line
9600 baud
connecting to wifi
.3connected successfully
TSL2591 sensor initialized.
Reading soil value
2622
Reading light intensity value
Light Level: 8.30 lux
Temperature Level:
nanHumidity Level:
nanData sent succesfully
200
Reading soil value
2615
Reading light intensity value
Ln 87, Col 57
ESP32 Dev Module on /dev/cu.usbserial-0001
2


ESP32 Dev Module
tinkerlab_plant_whisperer_group_7.ino
#include <WiFi.h> //Includes the WiFi library to enable ESP32 to connect to a wireless network.
#include <HTTPClient.h> //Allows HTTP requests like GET and POST for sending data online.
#include <Wire.h> //Used for I2C communication between ESP32 and sensors.
#include <Adafruit_Sensor.h> //Base class for all Adafruit sensor libraries.
#include <Adafruit_TSL2591.h> //Library for the TSL2591 light sensor.
#include <DHT.h> //Includes support for the DHT temperature and humidity sensor.
// Temperature and humidity sensor pinmode and intiallization
#define DHTPIN 4 //Defines the GPIO pin 4 as the data pin for the DHT22 sensor.
#define DHTTYPE DHT22 //Specifies the sensor model being used as DHT 22 (AM2302).
DHT dht(DHTPIN, DHTTYPE); //Initializes the DHT sensor with the defined pin and type.
// Light intensity sensor intiallization
Adafruit_TSL2591 tsl = Adafruit_TSL2591(2591); //pinmode is predefined in the library
// Soil moisture sensor
const int soil_pin = 34; //Defines GPIO 34 (pin number 34) for reading analog soil moisture.
// Requirements for wifi connection
const char* ssid = "OnePlus"; //Stores your WiFi SSID (network name).
const char* password = "Geosense123"; //Stores the WiFi password.
const char* server = "https://api.thingspeak.com/update"; //URL to send sensor data to ThingSpeak.
String apiKey = "AG2Y47H4KAIWJBMZ"; //Your ThingSpeak write API key for uploading data.
void setup() //Setup function runs once at boot to initialize everything.
{
Serial.begin(9600); // Starts serial communication at 9600 baud for debugging.
Output
Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on '/dev/cu.usbserial-0001')
New Line
9600 baud
2732
Reading light intensity value
Light Level: 21.42 lux
Temperature Level:
27.40Humidity Level:
41.80Data sent succesfully
200
Reading soil value
2631
Reading light intensity value
Light Level: 21.73 lux
Temperature Level:
27.50Humidity Level:
41.60
Ln 87, Col 57
ESP32 Dev Module on /dev/cu.usbserial-0001
2

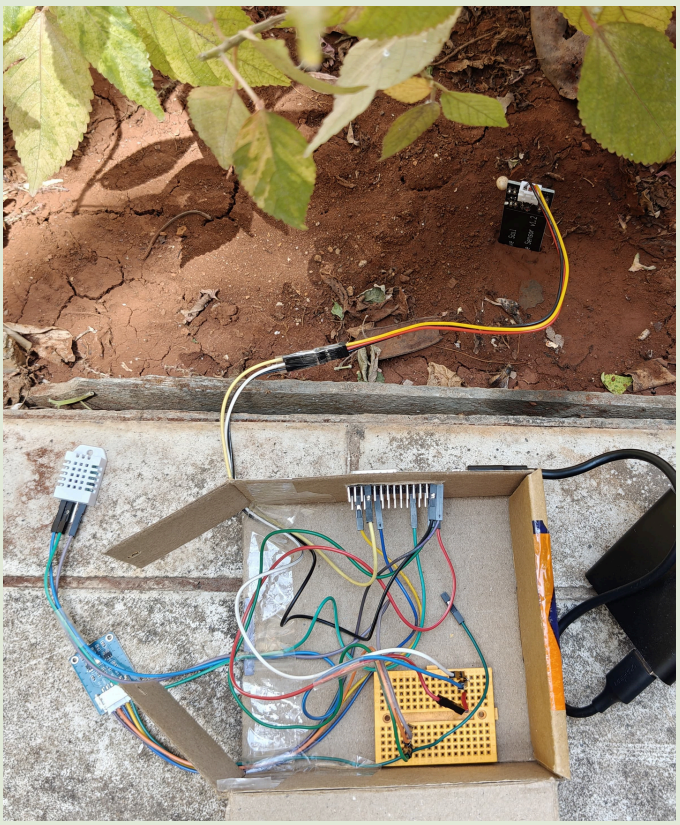
Measured Parameters Plotted On ThingSpeak

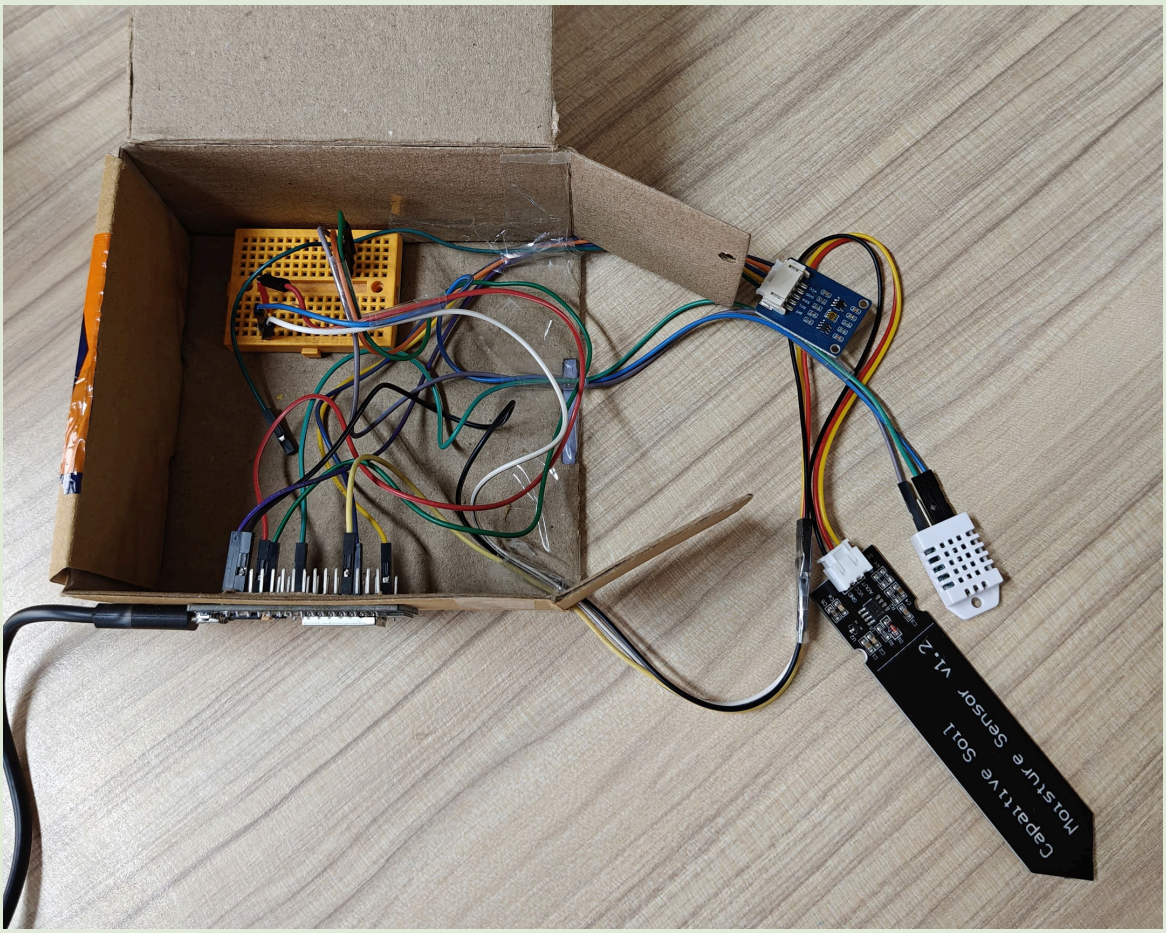


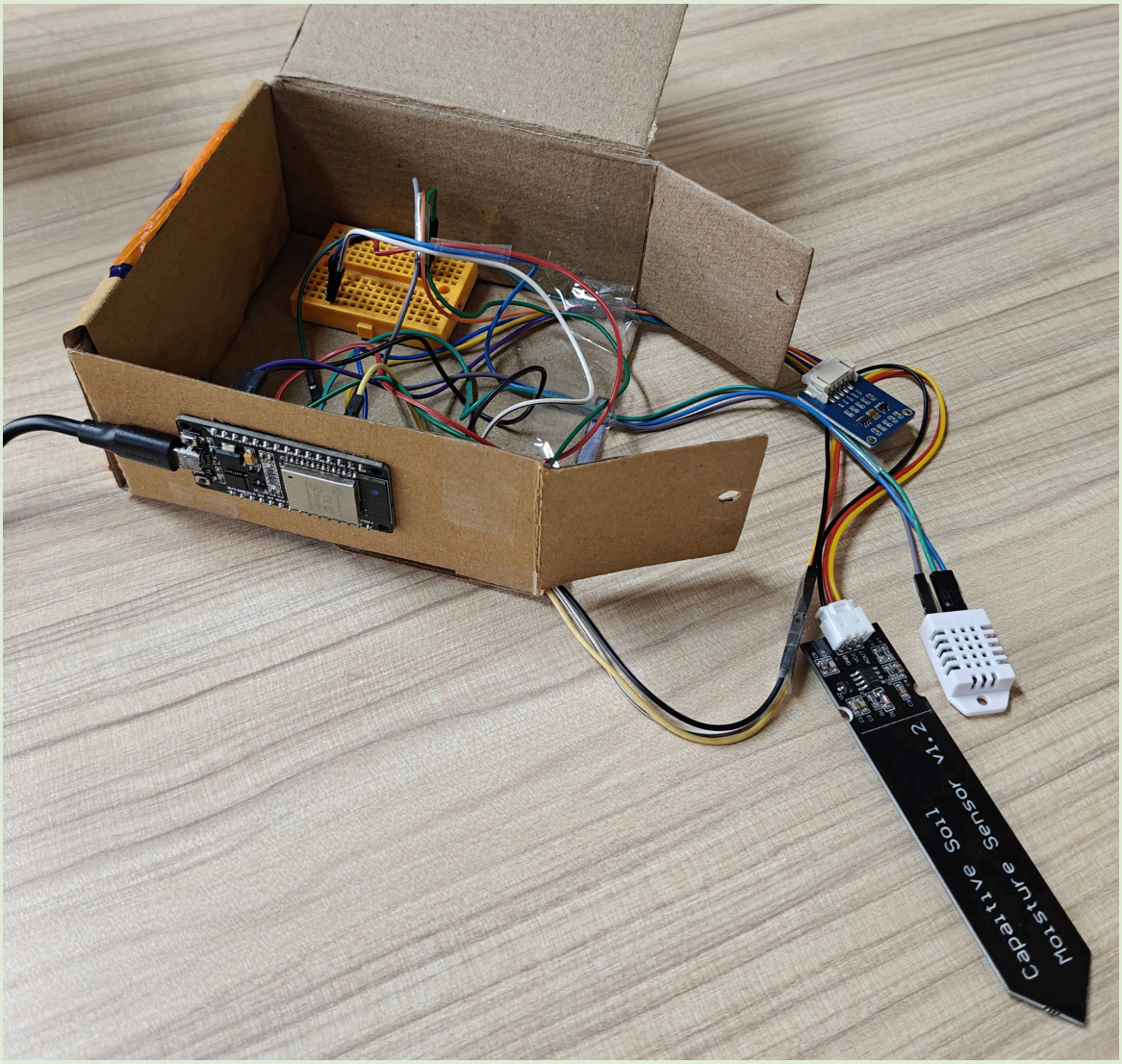

thingspeak.mathworks.com/channels/2912895/private_show
New Chrome available
ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
AP
Field 1 Chart
Plant Whisperer
Temmperature
34.0
33.5
33.0
32.5
13:11:00
13:12:00
13:13:00
13:14:00
13:15:00
Date
ThingSpeak.com
Field 2 Chart
Plant Whisperer
Moisture
2.50k
2.00k
1.50k
18:00
16 Apr
06:00
12:00
Date
ThingSpeak.com
Field 3 Chart
Plant Whisperer
Light Intensity
1.00k
800
600
400
200
0.00
18:00
16 Apr
06:00
12:00
Date
ThingSpeak.com
Field 4 Chart
Plant Whisperer
Humidity
39.0
38.0
37.0
36.0
35.0
34.0
13:11:00
13:12:00
13:13:00
13:14:00
13:15:00
Date
ThingSpeak.com
Channel Location


thingspeak.mathworks.com/channels/2912895/private_show
New Chrome available
ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
AP
ThingSpeak.com
ThingSpeak.com
Field 3 Chart
Plant Whisperer
Light Intensity
1.00k
800
600
400
200
0.00
18:00
16 Apr
06:00
12:00
Date
ThingSpeak.com
Field 4 Chart
Plant Whisperer
Humidity
39.0
38.0
37.0
36.0
35.0
34.0
13:11:00
13:12:00
13:13:00
13:14:00
13:15:00
Date
ThingSpeak.com
Channel Location
Lonavala
Latitude: 18.542054
Longitude: 73.727849
Pune
60

Pictures Of Testing




Capacitive Soil
Moisture Sensor V1.2


Capacitive Soil
Moisture Sensor V1.2

CONCLUSION

The "Plant Whisperer" system offers an effective and user-friendly solution for maintaining optimal plant health by automating the monitoring of essential environmental conditions. By providing real-time data on light intensity, temperature, humidity, and soil moisture, it empowers users to make informed decisions about plant care. This not only reduces the chances of over- or under-watering but also promotes healthier growth, making plant maintenance easier for both beginners and busy individuals. Ultimately, the project enhances the overall plant care experience, encouraging more sustainable and successful gardening practices.

Future Scope

The current project, which involves monitoring temperature, humidity, soil moisture, and light intensity for a plant using an ESP32 microcontroller, presents numerous opportunities for future development and real-world applications.

Some of the potential enhancements and directions for expansion are outlined below:

- Automated Irrigation System: By integrating an electric water pump and a relay module, the system can be upgraded to automate irrigation. Watering can be triggered based on real-time soil moisture readings, thereby reducing manual intervention and ensuring optimal hydration of the plant.
- Data Logging and Analysis: Incorporating data logging functionality can enable the system to store historical sensor readings. Analyzing this data over time can provide valuable insights into environmental trends and plant behavior, leading to improved decision-making for plant care.
- Machine Learning Integration: Future iterations of the project can include machine learning models trained on historical data to predict watering needs, detect anomalies, or suggest optimal environmental conditions. This would enhance the intelligence and adaptability of the system.
- Scalability for Multiple Plants: The project can be extended to monitor multiple plants by interfacing additional sensors and using a central controller. This feature would make the system suitable for greenhouses, nurseries, or small-scale farming operations.