VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRICAL ENGINEERING

BECE351E – INTERNET OF THINGS

RESEARCH PROJECT

Name:

1. **C Yagnesh (21BEE0260)**
2. **Shailesh raj (21BEE0186)**
3. **Lokesh Yadav (21BEE0192)**
4. **Debarshee Apurba (21BEE0133)**
5. **Saurish Bharati (21BEE0415)**

* Project Title:

Long-Term Environmental Monitoring Using ESP32

* Introduction:

Long-Term Environmental Monitoring and Graph Visualization using ESP32 is an innovative IoT research project that focuses on continuous monitoring of environmental parameters and the visual representation of data using graphs. The project leverages the capabilities of the ESP32 microcontroller, which combines a powerful processor, built-in Wi-Fi connectivity, and numerous GPIO pins for sensor integration. By collecting and analyzing environmental data over an extended period, valuable insights can be gained, leading to a deeper understanding of environmental conditions, patterns, and trends.

The primary objective of this project is to develop a robust and reliable system for long-term environmental monitoring. By deploying sensors such as temperature, humidity, air quality, and light intensity sensors, the ESP32 can gather real-time data at regular intervals. These sensors play a vital role in capturing key environmental parameters that significantly impact our daily lives, including the quality of the air we breathe, the comfort levels within a space, and the effects of light on our surroundings.

To ensure seamless integration and accurate data acquisition, the ESP32 microcontroller is programmed using the Arduino IDE. The code written for the ESP32 enables it to initialize the sensors, read sensor data periodically, and store the collected data locally or transmit it to a cloud-based platform for further analysis.

The collected environmental data is then subjected to analysis, which involves identifying patterns, trends, and correlations. Statistical metrics such as averages, minimum and maximum values, and standard deviations can be calculated to gain deeper insights into the data. These findings provide valuable information for various applications, including environmental research, smart city planning, and building automation.

In conclusion, the Long-Term Environmental Monitoring Visualization Using ESP32 project presents an exciting opportunity to explore the intersection of IoT, environmental science, and data visualization. By employing the ESP32 microcontroller, integrating various sensors, and utilizing graph visualization techniques, this project enables the acquisition, analysis, and visualization of environmental data over an

extended timeframe. The project's outcomes have the potential to contribute to a deeper understanding of environmental conditions, facilitate data-driven decision-making, and promote sustainable practices in various domains.

* Component Description:

1. ESP32 development board:

ESP32 is a low-cost System on Chip (SoC) Microcontroller from Espressif Systems, the developers of the famous ESP8266 SoC. It is a successor to ESP8266 SoC and comes in both single-core and dual-core variations of the Tensilica's 32bit Xtensa LX6 Microprocessor with integrated Wi-Fi and Bluetooth.

The good thing about ESP32, like ESP8266 is its integrated RF components like Power Amplifier, Low-Noise Receive Amplifier, Antenna Switch, Filters and RF Balun. This makes designing hardware around ESP32 very easy as you require very few external components.

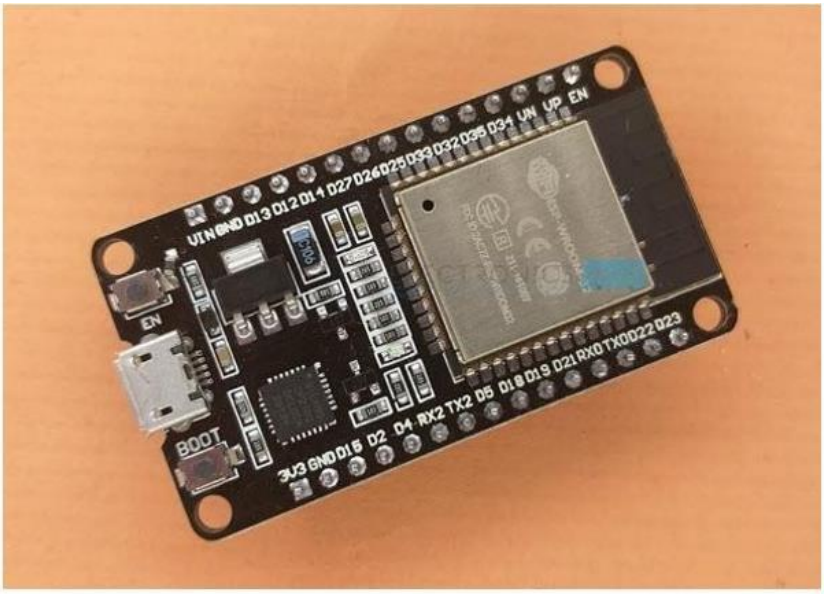
Another important thing to know about ESP32 is that it is manufactured using TSMC's ultra-low-power 40 nm technology. So, designing battery operated applications like wearables, audio equipment, baby monitors, smart watches, etc., using ESP32 should be very easy.

Why are they so popular? Mainly because of the following features:

Low-cost: you can get an ESP32 starting at $6, which makes it easily accessible to the general public;
Low-power: the ESP32 consumes very little power compared with other microcontrollers, and it supports low-power mode states like deep sleep to save power;
Wi-Fi capabilities: the ESP32 can easily connect to a Wi-Fi network to connect to the internet (station mode), or create its own Wi-Fi wireless network (access point mode) so other devices can connect to it—this is

essential for IoT and Home Automation projects—you can have multiple devices communicating with each other using their Wi-Fi capabilities;
Bluetooth: the ESP32 supports Bluetooth classic and Bluetooth Low Energy

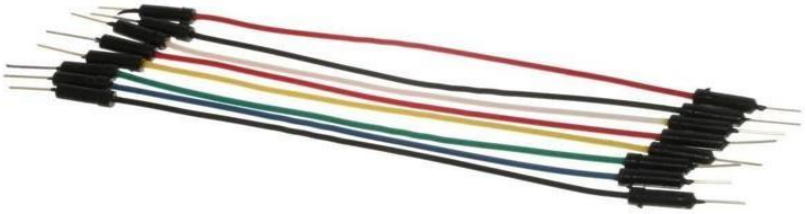
(BLE)—which is useful for a wide variety of IoT applications;
Dual-core: most ESP32 are dual-core— they come with 2 Xtensa 32-bit LX6 microprocessors: core 0 and core 1.



2. Breadboard and jumper wires:

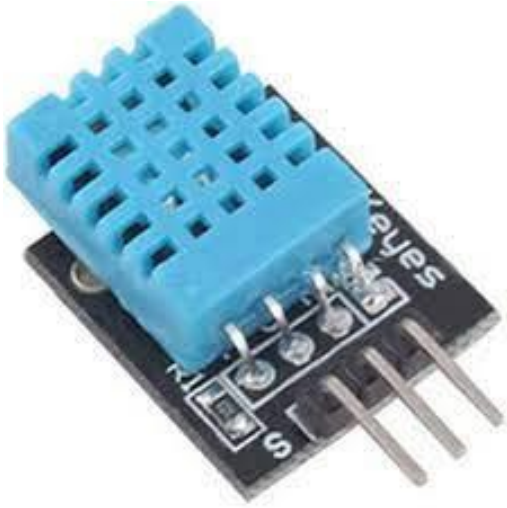
Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires.
Though jumper wires come in a variety of colours, the colours don't mean anything. This means that a red jumper wire is technically the same as a black one. But the colours can be used to your advantage to differentiate between types of connections, suchas ground or power.
There are different types of jumper wires. Some have the same type of electrical connecter at both ends, while others have different connectors. Some common connectors are:

Solid tips – are used to connect on/with a breadboard or female header connector. The arrangement of the elements and ease of insertion on a breadboard allows increasing the mounting density of both components and jump wires without fear of short-circuits. The jump wires vary in size and colour to distinguish the different working signals.



3. DHT11:

The DHT11 is a popular digital temperature and humidity sensor module. It is commonly used in various electronic projects and IoT devices to measure environmental conditions. The sensor integrates a calibrated digital signal output, making it easy to interface with microcontrollers and other digital circuits. DHT11 provides accurate and reliable readings for both temperature and humidity, and it comes in a compact form factor with a simple 3-pin interface (VCC, data, and GND). It is widely utilized in home automation, weather stations, and climate control systems due to its affordability and ease of use.



4. MQ135:

The MQ135 is a gas sensor that can detect a wide range of harmful gases, including ammonia, nitrogen oxides, benzene, smoke, and various other volatile organic compounds (VOCs). This sensor is commonly employed in air quality monitoring systems to assess indoor and outdoor air pollution levels. The MQ135 operates based on the principle of the resistance change in response to different gas concentrations. The sensor's analog output can be connected to ADC modules or microcontrollers to interpret gas concentration levels and trigger appropriate actions or alerts. It finds applications in smart homes, environmental monitoring, and health-related projects.

5. Raindrop Sensor:

A raindrop sensor, as the name suggests, is a weather sensor used to detect rain or the presence of water on its surface. These sensors typically consist of conductive tracks on a substrate that form a capacitor. When raindrops or water come into contact with the sensor, the capacitance changes, and this change is detected by the connected circuitry. Raindrop sensors are widely used in weather stations, automated sprinkler systems, and rain detection systems for automobiles.

6. Light Sensor (LDR - Light Dependent Resistor):

An LDR, or Light Dependent Resistor, is a type of photoresistor whose resistance varies based on the amount of light falling on its surface. The higher the light intensity, the lower the resistance of the LDR, and vice versa. This property allows LDRs to be used as simple light sensors in various applications. They are commonly found in automatic streetlight control systems, camera exposure control, outdoor lighting, and many consumer electronics. When used in combination with other components, LDRs can facilitate a wide range of light-based automation tasks. Their ease of use, low cost, and reliability make them a popular choice in the world of electronics and DIY projects.

* Code Sample:

```
#include <WiFi.h>

#include <HTTPClient.h>

#include <Adafruit_Sensor.h>

#include <DHT.h>

#include <MQUnifiedsensor.h>


#define DHTPIN 4     // DHT11 sensor pin

#define DHTTYPE DHT11   // DHT sensor type

#define MQ_PIN A0    // MQ135 sensor pin
```

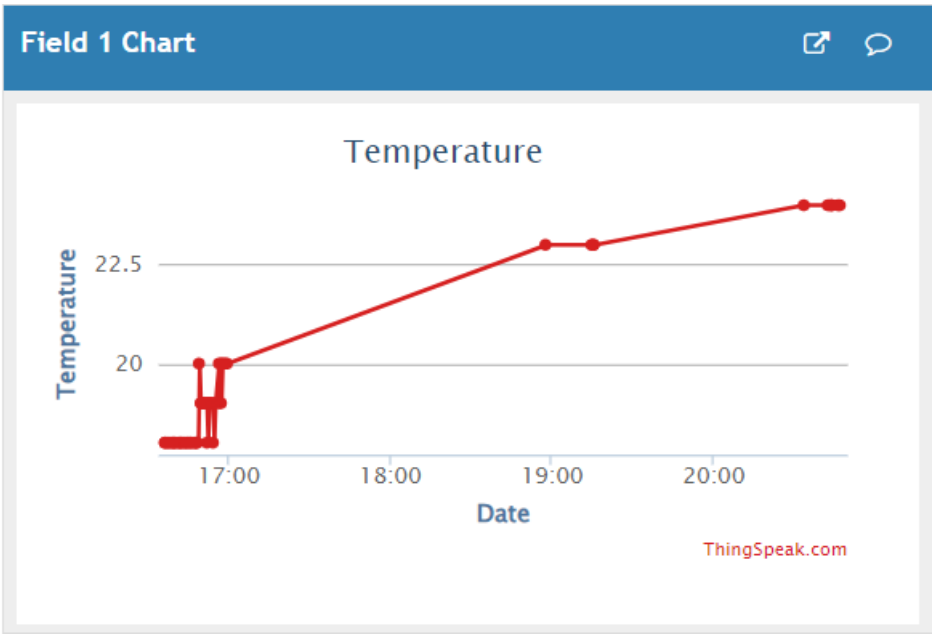
```
#define RL_VALUE 20.0 // MQ135 resistance RL value

#define RO_CLEAN_AIR_FACTOR 9.83 // MQ135 clean air factor

#define LDR_PIN 36   // LDR sensor pin

#define RAIN_PIN 39  // Raindrop sensor pin

#define WIFI_SSID "Yagnesh"

#define WIFI_PASSWORD "yagnesh27"

#define THINGSPEAK_API_KEY "VVFHPHXQPZ3D47ZW"

#define THINGSPEAK_CHANNEL_ID "2224707"


DHT dht(DHTPIN, DHTTYPE);

MQUnifiedsensor   gas(MQ_PIN,   MODE_ANALOG,   RESOLUTION_ADC_5V,
RL_VALUE, RO_CLEAN_AIR_FACTOR);

int ldrValue = 0;

int rainValue = 0;


void setup() {

  Serial.begin(115200);


  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {

    delay(1000);

    Serial.println("Connecting to WiFi...");

  }

  Serial.println("Connected to WiFi");
```

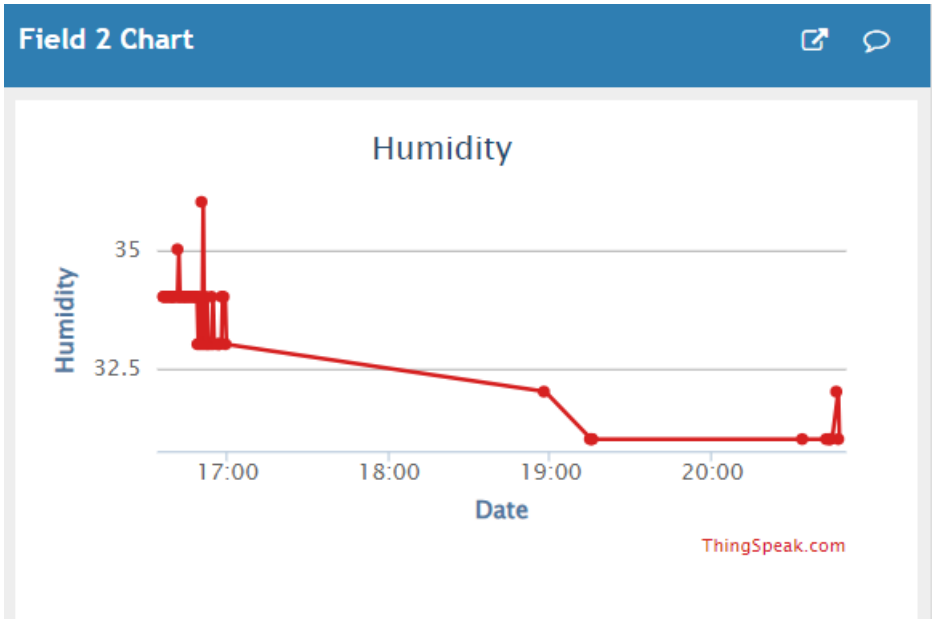
```
  dht.begin();
  gas.begin();
  pinMode(LDR_PIN, INPUT);
  pinMode(RAIN_PIN, INPUT);
}

void loop() {
  delay(2000); // Delay between readings

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  float gasDensity = gas.readDensity();

  ldrValue = analogRead(LDR_PIN);
  rainValue = digitalRead(RAIN_PIN);

  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  if (gasDensity >= 0) {
    Serial.print("Gas Density: ");
    Serial.print(gasDensity);
    Serial.println(" ppm");
```

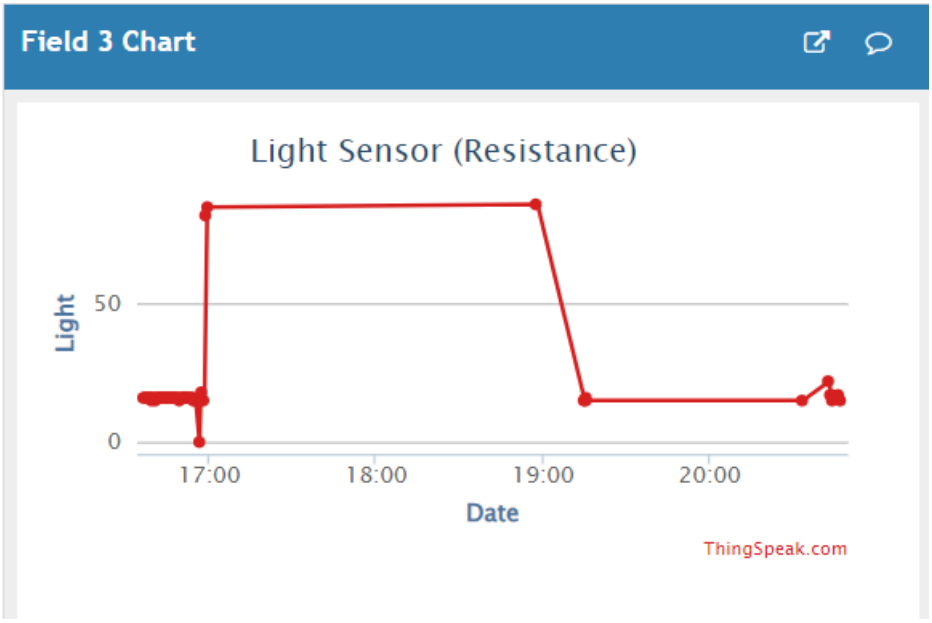
```
  } else {
    Serial.println("Failed to read from MQ135 sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C, Humidity: ");
  Serial.print(humidity);
  Serial.print(" %, LDR Value: ");
  Serial.print(ldrValue);
  Serial.print(", Raindrop Value: ");
  Serial.println(rainValue);

  sendToThingSpeak(temperature, humidity, gasDensity, ldrValue, rainValue);
}

void sendToThingSpeak(float temperature, float humidity, float gasDensity, int ldrValue, int rainValue) {
  HTTPClient http;
  String url = "http://api.thingspeak.com/update?api_key=" + String(THINGSPEAK_API_KEY) +
               "&field1=" + String(temperature) +
               "&field2=" + String(humidity) +
               "&field3=" + String(gasDensity) +
```

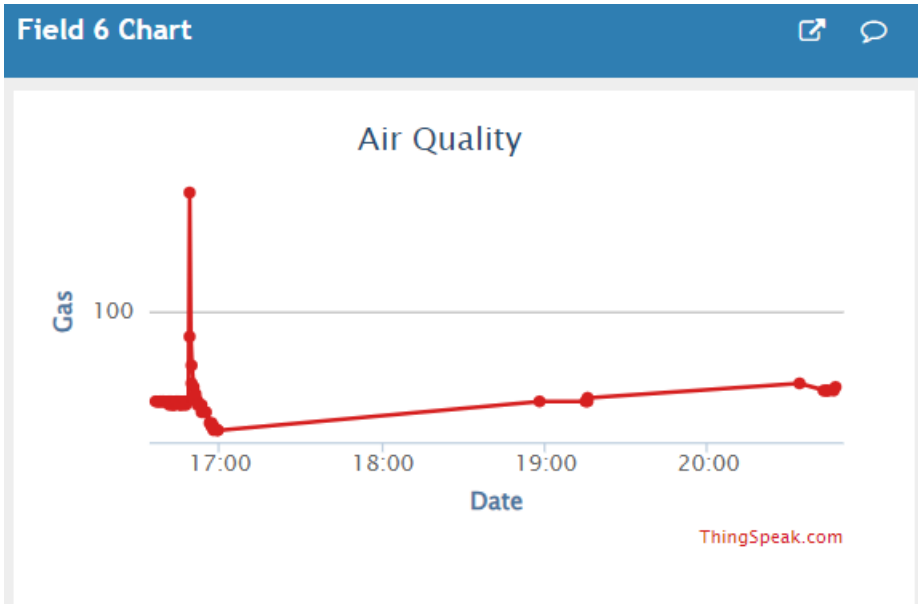
```
                "&field4=" + String(ldrValue) +
                "&field5=" + String(rainValue);

  http.begin(url);
  int httpResponseCode = http.GET();
  if (httpResponseCode == 200) {
    Serial.println("Data sent to ThingSpeak successfully");
  } else {
    Serial.print("Error sending data to ThingSpeak. HTTP response
code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}
```

* **Output Graph –**

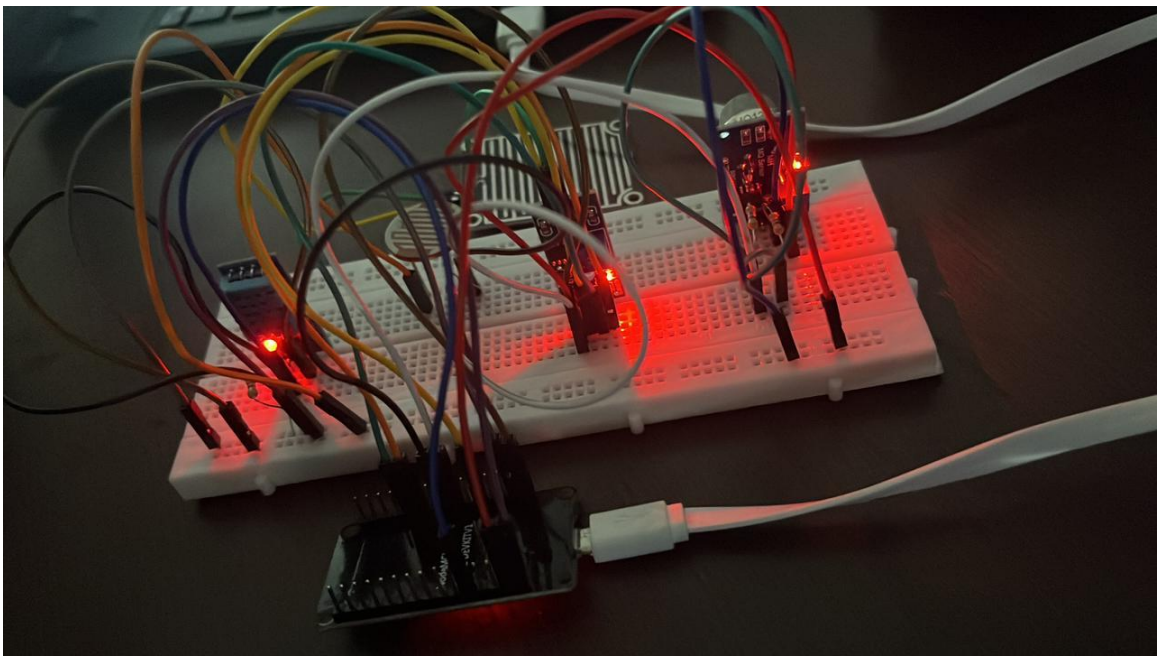


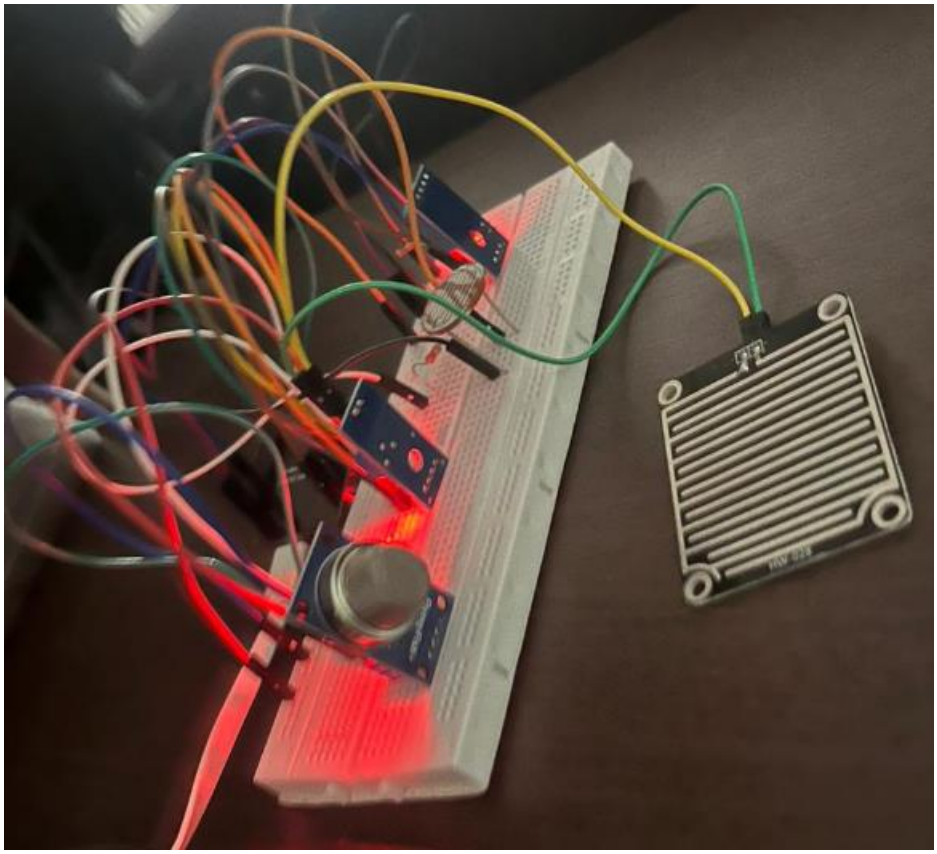

Field 2 Chart
Humidity
35
32.5
Humidity
17:00
18:00
19:00
20:00
Date
ThingSpeak.com

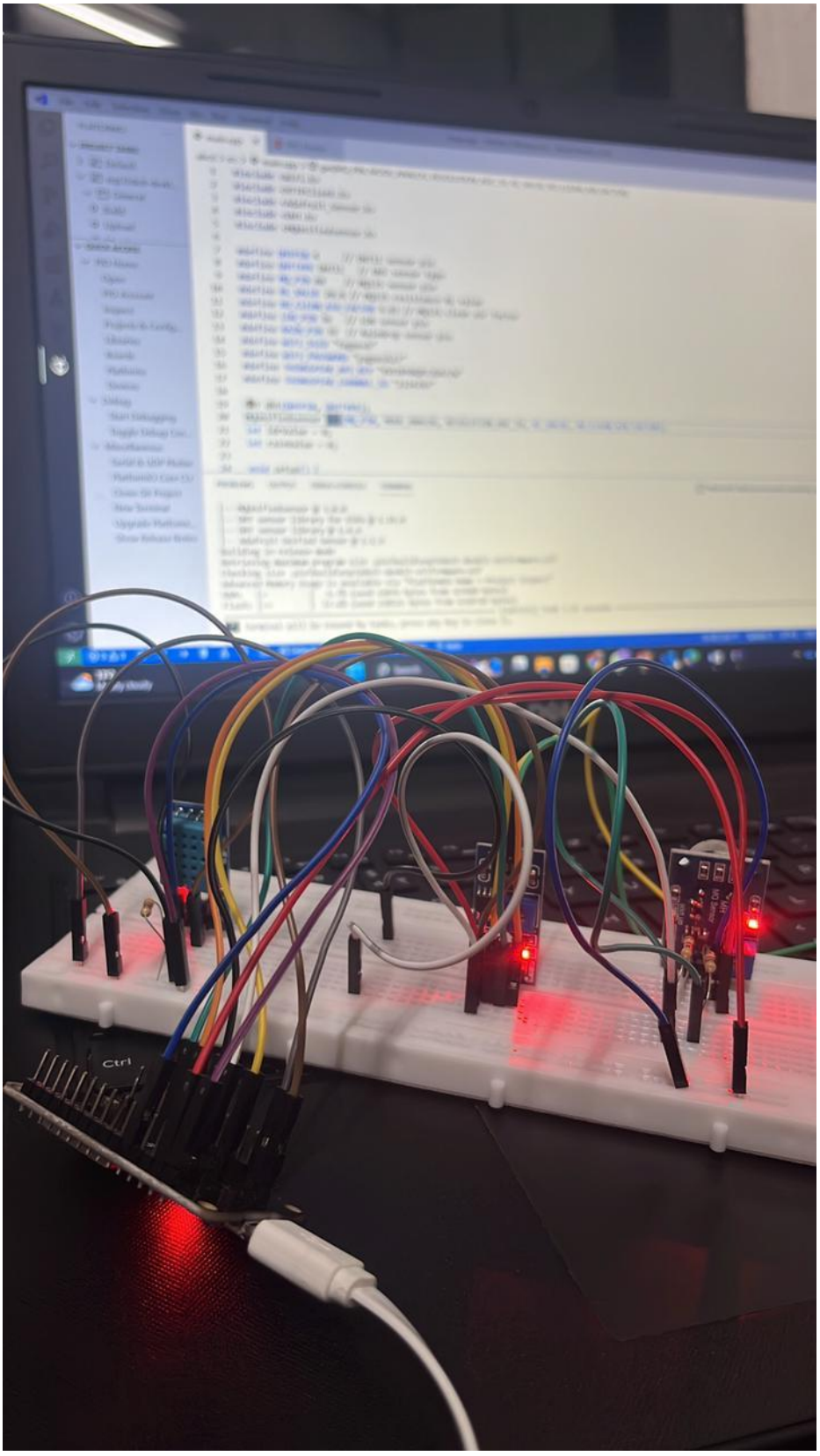

Field 3 Chart
Light Sensor (Resistance)
50
0
Light
17:00
18:00
19:00
20:00
Date
ThingSpeak.com


Field 6 Chart
Air Quality
100
Gas
17:00
18:00
19:00
20:00
Date
ThingSpeak.com

* **Hardware Images –**






Ctrl