**Project To Increase Productivity in Agriculture Using Embedded System (TARGO)**

**Introduction:**

This project introduces **Targo**, an innovative system designed to optimize agricultural efficiency by integrating advanced sensing and analysis technologies. Targo precisely measures critical soil parameters such as water content, temperature, humidity, and essential mineral levels, including potassium, nitrogen, and phosphorus. By leveraging data collected directly from the soil, the system identifies deficiencies and provides realtime feedback to users, enabling prompt corrective actions.

Moreover, Targo employs advanced algorithms to analyze soil conditions and recommend the most suitable soil types for improvement. It also predicts the most productive fruit or vegetable planting options based on current conditions, empowering farmers to make data-driven decisions. With its focus on enhancing productivity and sustainability, Targo represents a cutting-edge approach to modern agriculture, offering actionable insights to maximize yields while preserving environmental health.

**Key Features:**

1. Soil Moisture Sensor:
   * Measures the soil's moisture content with precision.
   * Helps optimize irrigation scheduling to prevent overwatering or water stress.
2. Temperature and Humidity Sensor (DHT11):
   * Monitors air temperature and humidity levels around the crop field.
   * Ensures environmental conditions are suitable for plant growth.
3. DS18B20 Waterproof Temperature Sensor:
   * Provides accurate soil temperature readings.
   * Crucial for monitoring root-zone conditions.
4. Soil Nutrient Sensors:
   * Detects essential mineral levels such as potassium, nitrogen, and phosphorus in the soil.
   * Identifies deficiencies to guide fertilization strategies.

1. Water Level and Rain Sensor:
   * Monitors water availability and rainfall in the area.
   * Supports precise water management by tracking precipitation levels.
2. Gas Detection Sensor (MQ-135):
   * Measures air quality by detecting hazardous gases.
   * Ensures a safe and healthy environment for plants and farmers.
3. OLED Display:
   * Displays real-time data directly on-site for easy monitoring.
   * Offers visual representation of all collected parameters.
4. Communication Module:
   * Transmits collected data to a central control unit or cloud storage.
   * Enables remote monitoring and decision-making.

**Project Objectives:**

The primary objective of the Project to Increase Productivity in Agriculture Using Embedded System (TARGO) is to develop a smart and efficient system that leverages embedded technology to monitor, analyze, and optimize agricultural practices. By integrating real-time data collection, advanced algorithms, and user-friendly interfaces, TARGO aims to empower farmers to make informed decisions that enhance crop productivity, conserve resources, and promote sustainable farming practices.

**Components:**

|  |  |  |
| --- | --- | --- |
| Components | Quantity | |
| 12V Soil Hygrometer Moisture Detection Module | | 1 |
| Soil Moisture Detection Sensor | | 1 |
| Water Level / Rain Sensor | | 1 |
| Water Level / Rain Sensor | | 1 |
| DS18B20 Waterproof Temperature Sensor | | 1 |
| MQ-135 Air Quality Control Sensor Module | | 1 |
| Jumper Cable Male to Male | | 43 |
| Jumper Cable Female to Male | | 37 |
| Jumper Cable Female to Female | | 17 |
| Arduino Mega 2560 R3 | | 1 |
| Arduino MEGA 2560 R3 Plexi Box | | 1 |
| 1.3 inch I2C OLED Display (128x64) SSD1106G | | 1 |
| DHT11 Temperature and Humidity Sensor Module | | 1 |
| Soil NPK Sensor | | 1 |
| ESP8266 WiFi Module ESP-01 | | 1 |
| 330R ¼W Resistor | | 1 |
| 1K ¼W Resistor | | 1 |
| LM2596 Mini Adjustable Voltage Regulator Board 3A (4-35V to 1.2530V) | | 1 |
| TTL-RS485 Serial Converter Board (MAX485 | | 1 |
| 10uF 16V Electrolytic Capacitor | | 1 |

**Desing Measure:**

**Lcd Port Connections:**

Port 1 → GND

Port 2 → VCC

Port 3 → Potentiometer Port 2

Port 4 → Arduino Port 7

Port 5 → GND

Port 6 → Arduino Port 6

Port 11 → Arduino Port 5

Port 12 → Arduino Port 4

Port 13 → Arduino Port 3

Port 14 → Arduino Port 2

Port 15 → VCC with resistor

Port 16 → GND

**Potentiometer Connections:**

Port 1 → VCC

Port 2 → LCD Port 3

Port 3 → GND

**Buzzer Connections:**

Buzzer (+) → Arduino Port 10

Buzzer (-) → GND

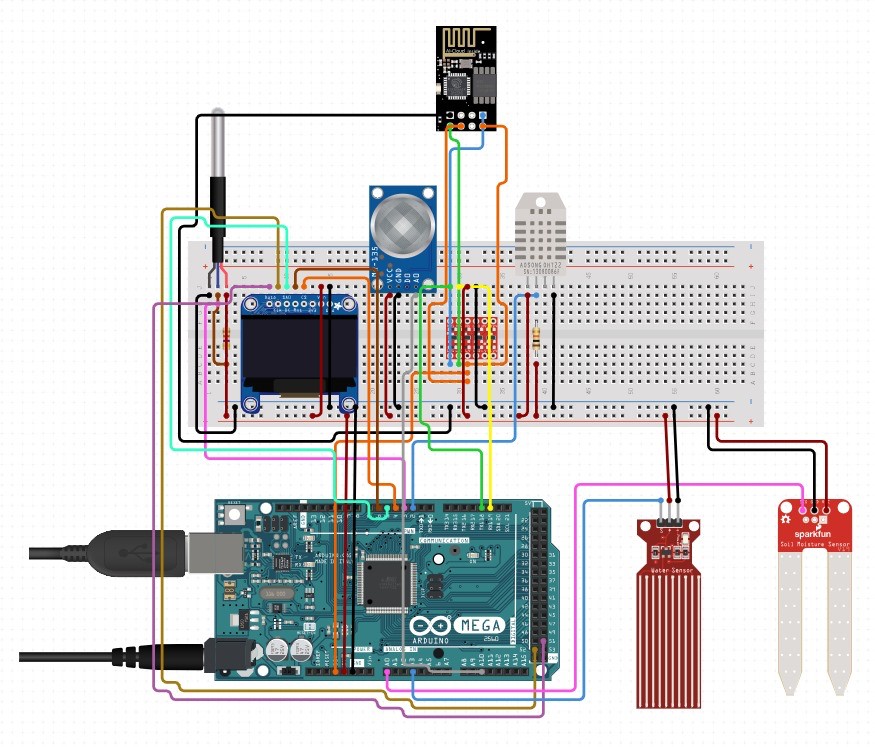
**Power Connections:**

From Arduino 5V → Board Power From Arduino GND → Board GND **Antenna Connections:**

VCC → 5V

GND → GND

CE → Port 10



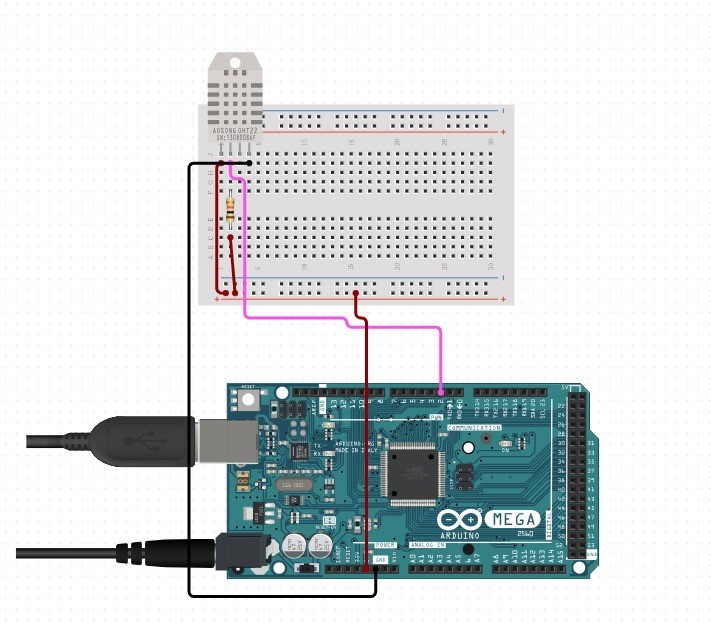


Figure 1: DHT11 (Temperature and Humidity Sensor)

Measures the surrounding temperature and humidity levels.

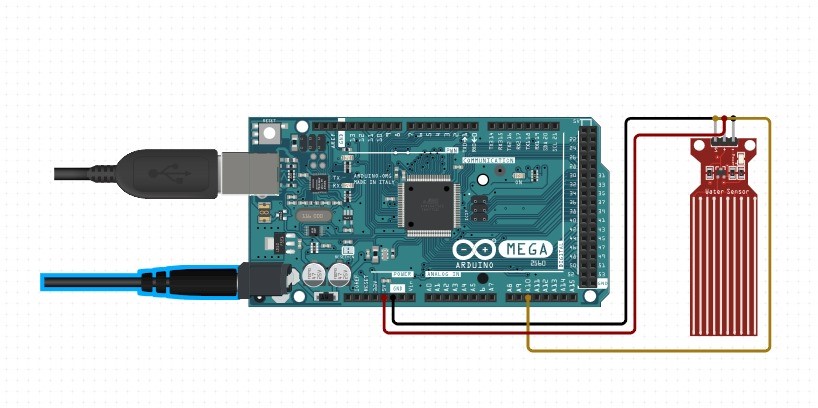


Figure 2: Water Level

Measures the surrounding temperature and humidity levels.

Detects the water level in a plant.

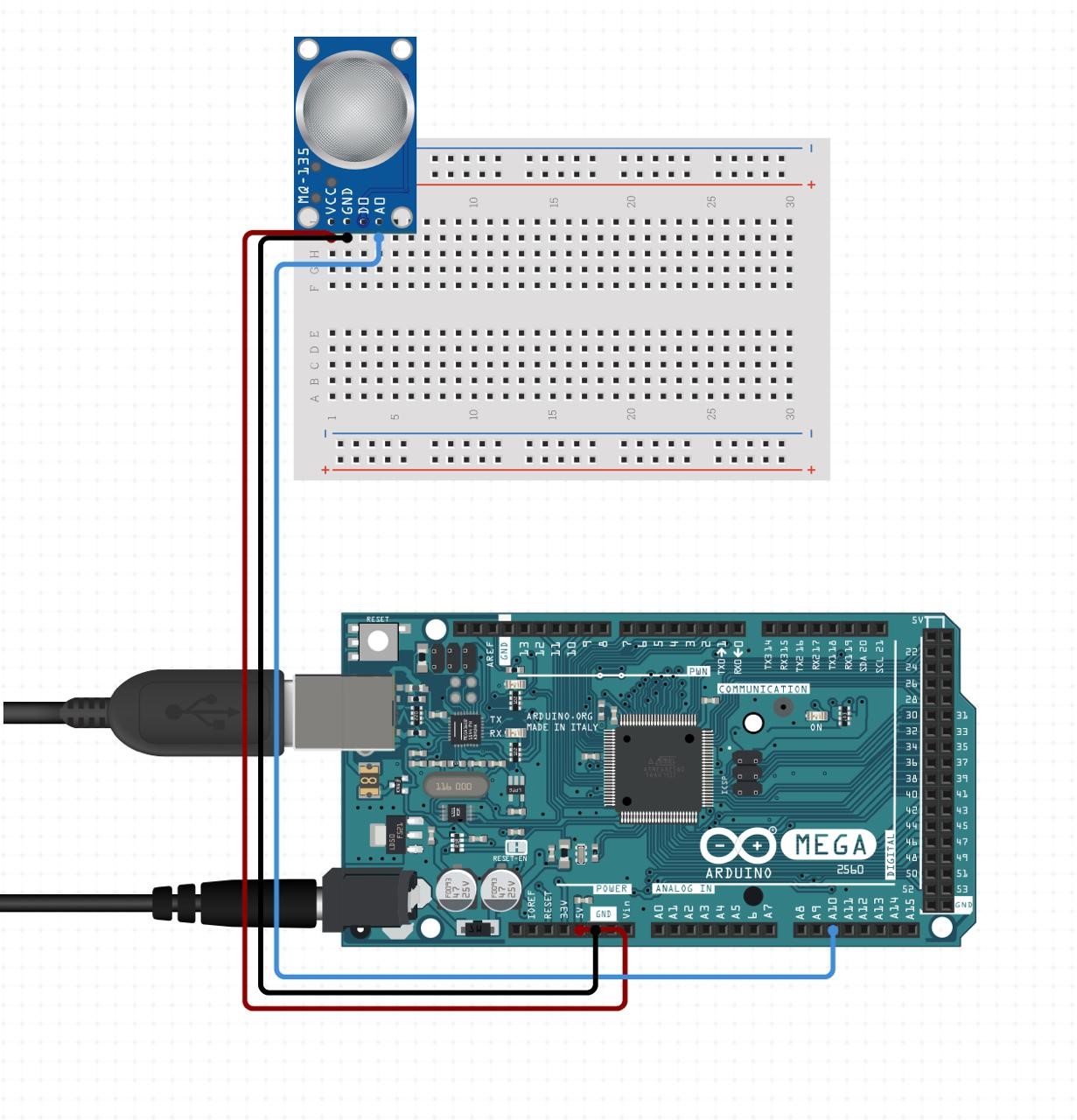


Figure 3:MQ-135

Measures air quality by detecting harmful gases like NH3, CO2, and smoke.

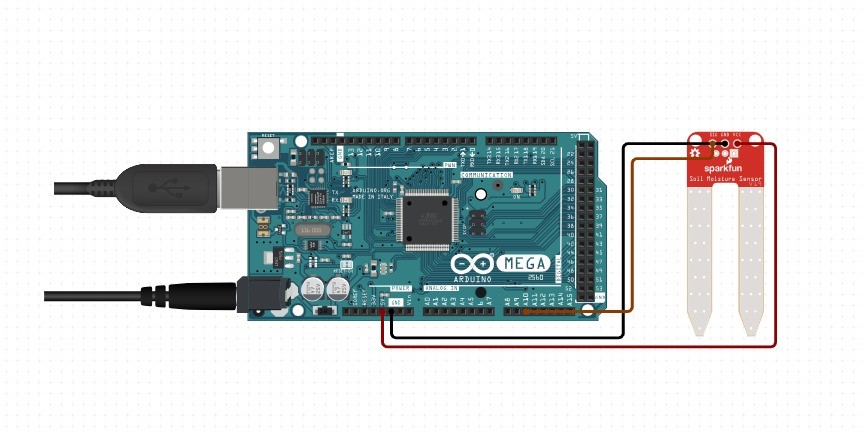


Figure 4:Soil Measure

Measures moisture content in the soil.

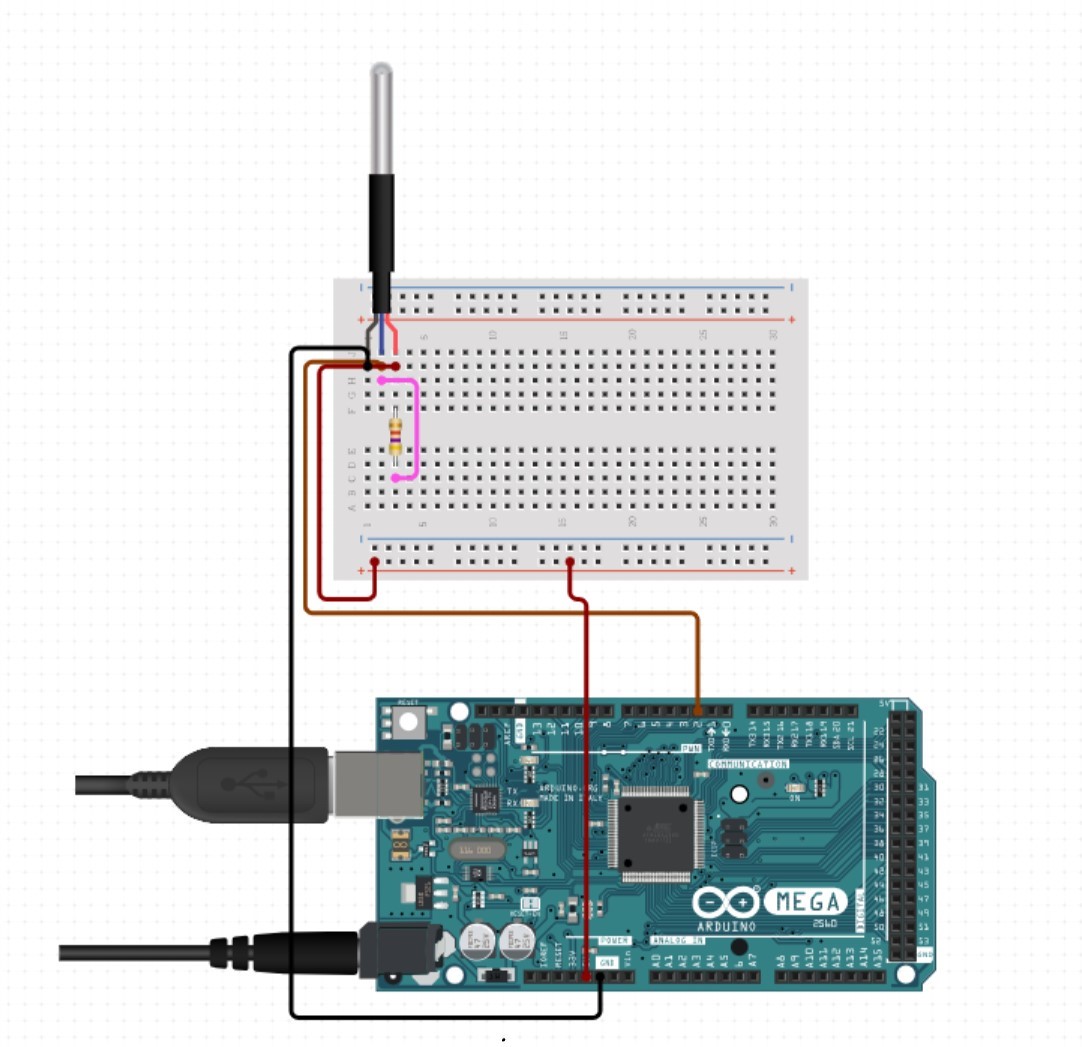


Figure 5:DS18B20 Waterproof Temperature Sensor

It measures the temperature inside the soil with its waterproof feature.



Figure 6: ESP8266 Wifi Module

Provides a wireless connection to transmit data from the sensors to the server.

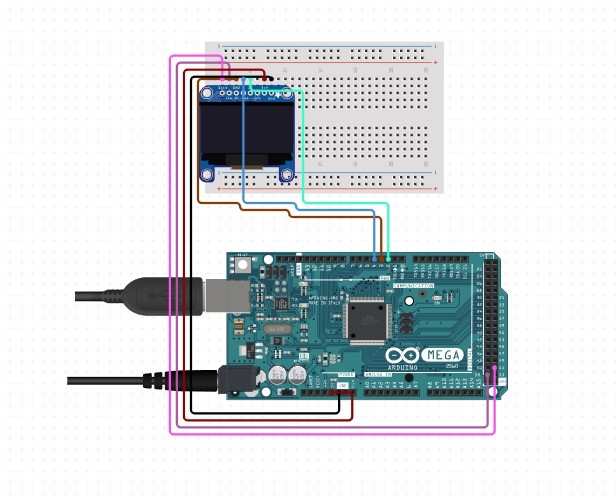


Figure 7: 1.3 inch I2C OLED Display

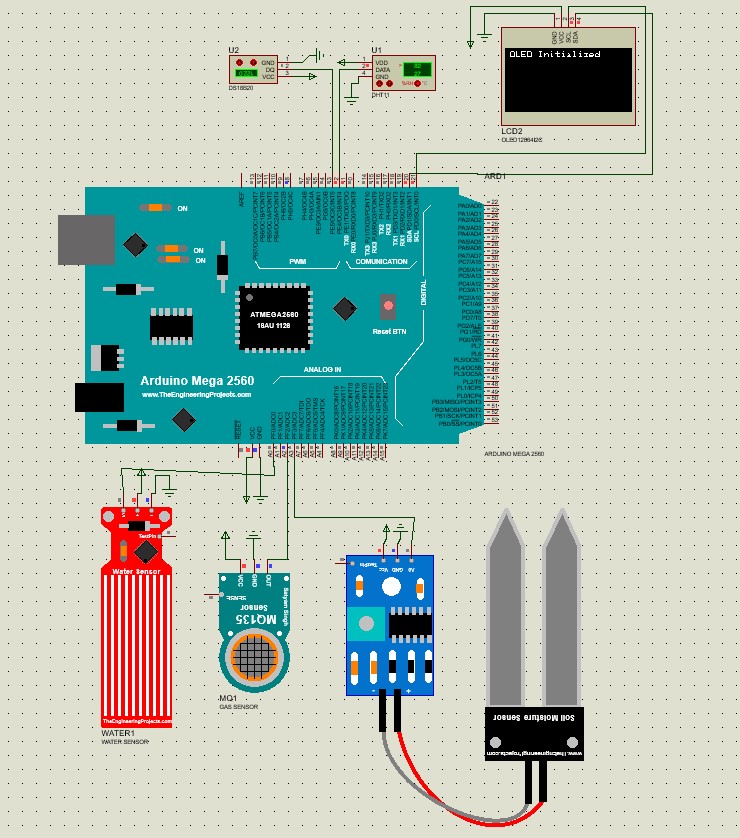
Provides a wireless connection to transmit data from the sensors to the server.

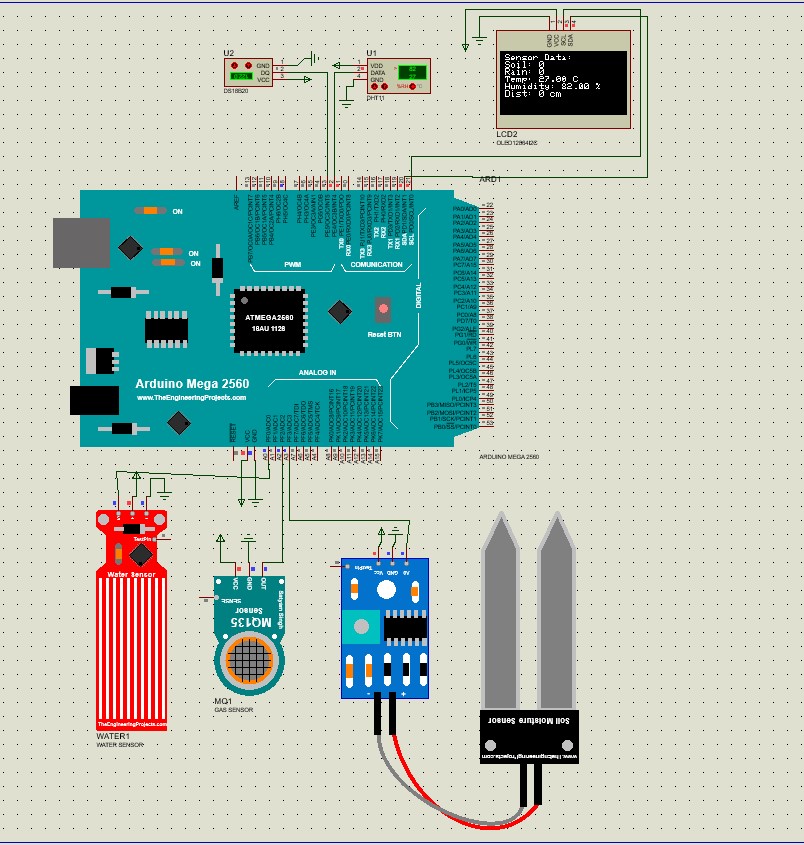


Figure 8: Soil NPK Sensor

Measures the concentration of Nitrogen (N), Phosphorus (P), and Potassium (K) in the soil.

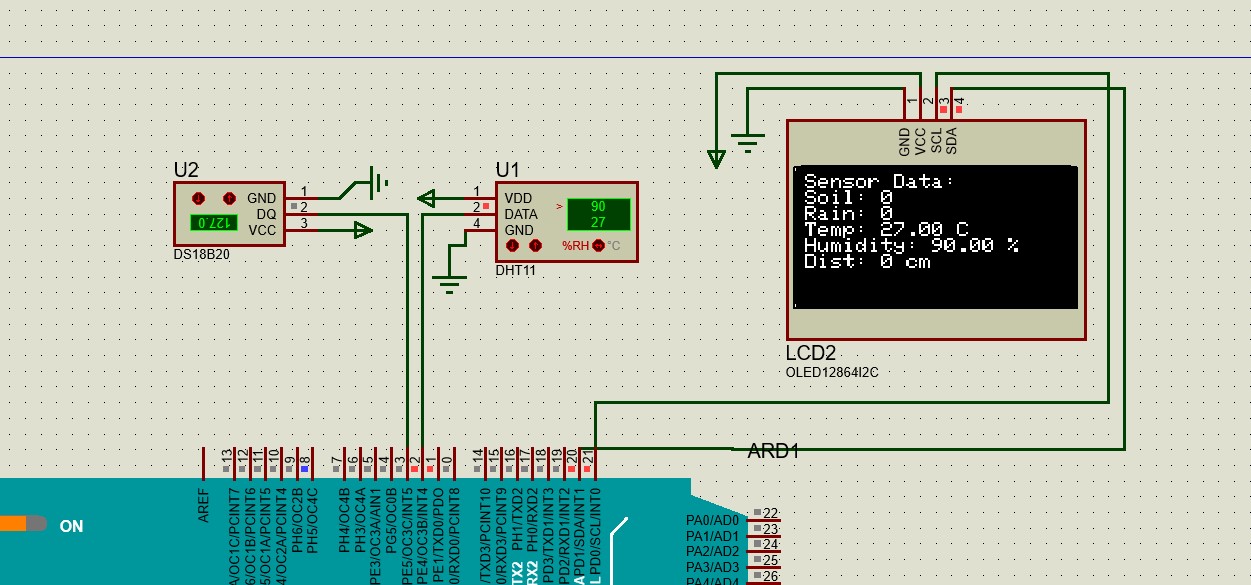
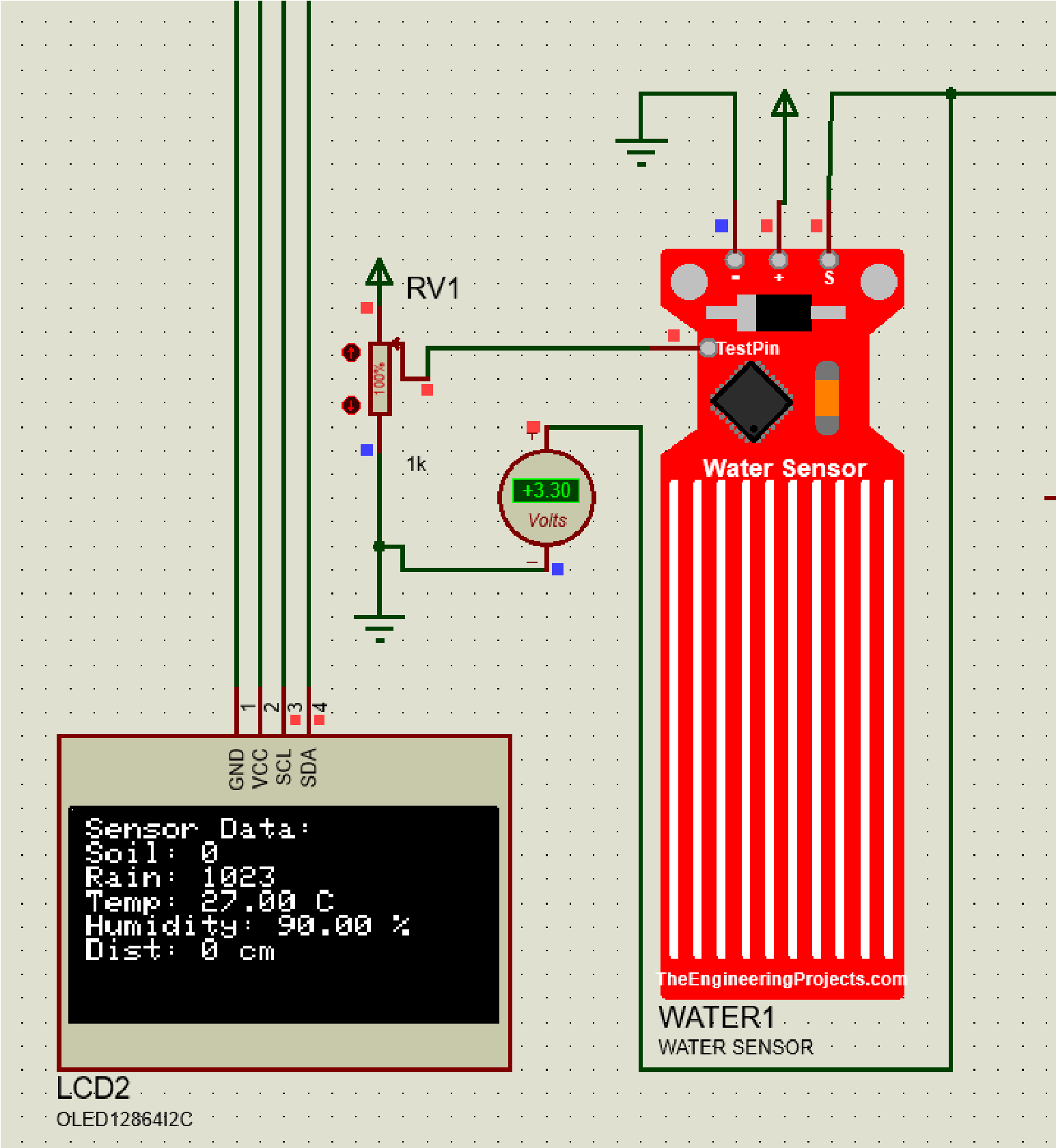
**Proteus Details:**

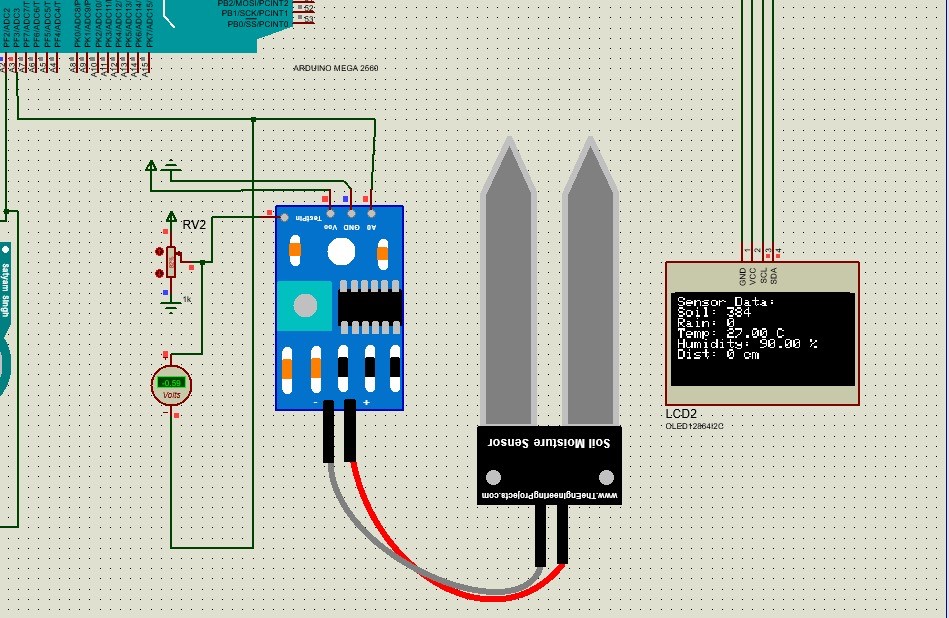




**Sensor Overview:**

We have 5 components that we can show in the proteus.





* DS18B20 waterproof temperature sensor is used to measure soil temperature.
* DHT11 temperature and humidity sensor monitors environmental temperature and humidity levels.
* MQ-135 air quality sensor measures harmful gas levels, ensuring safety.
* Water level and rain sensor tracks water availability and rainfall.
* Soil moisture sensor detects the soil's moisture level, aiding in irrigation management.

* The simulation tested the accuracy of sensor data generation.
* Real-time data was visualized on the OLED screen within the Proteus environment.
* Sensor data was successfully transmitted to the cloud using the

Thingspeak API.

|  |
| --- |
| #include <SoftwareSerial.h>  #include "DHT.h"  #include <OneWire.h>  #include <DallasTemperature.h>  #include <Wire.h>  #include <Adafruit\_GFX.h>  #include <Adafruit\_SH110X.h>    // RE and DE Pins set the RS485 module  // to Receiver or Transmitter mode  #define RE 8  #define DE 7    #define ONE\_WIRE\_BUS 3 // DS18B20 sensor pin  #define DHTPIN 2 // DHT sensor pin  #define DHTTYPE DHT11 // DHT11 type  #define GAS\_MQ A2 // MQ135 gas sensor  #define WATER A0 // Water Rain sensor  #define SOIL\_WATER A3 // Soil water sensor  #define PUMP 24 // PUMP    #define i2c\_Address 0x3c // OLED ekran adresi    #define SCREEN\_WIDTH 128 // OLED ekran genişliği, piksel cinsinden  #define SCREEN\_HEIGHT 64 // OLED ekran yüksekliği, piksel cinsinden  #define OLED\_RESET -1 // QT-PY / XIAO    // Modbus RTU requests for reading NPK values  const byte nitro[] = {0x01, 0x03, 0x00, 0x1E, 0x00, 0x01,  0xE4, 0x0C};  const byte phos[] = {0x01, 0x03, 0x00, 0x1F, 0x00, 0x01,  0xB5, 0xCC};  const byte pota[] = {0x01, 0x03, 0x00, 0x20, 0x00, 0x01, 0x85, 0xC0};    byte values[7]; // A variable used to store NPK values |

String networkAddressName = "Wowa"; //

Network adress name

String networkPassword = "deneme12345"; //

Network password

String ipAddress = "184.106.153.149"; //

Thingspeak ip address

float weatherTemperature, insideSoilTemperature, moistureSensorData, gasSensorData, waterRainSensorData, nitrogenData, phosphorousData, potassiumData,

soilWaterSensorData; int yPos = 0, sending = 0;

bool check = false, getDatasSW = true;

DHT dht(DHTPIN, DHTTYPE);

SoftwareSerial mod(12, 13);

OneWire oneWire(ONE\_WIRE\_BUS); // Setup a oneWire instance to communicate with any OneWire device

DallasTemperature sensors(&oneWire); // Pass oneWire reference to DallasTemperature library

Adafruit\_SH1106G display = Adafruit\_SH1106G(SCREEN\_WIDTH,

SCREEN\_HEIGHT, &Wire, OLED\_RESET);

void setup() {

Serial.begin(9600); // Seri port communication started

Serial.println("Started");

/-------------- OLED Ekranı Başlat --------------/ delay(250); // OLED'in açılması için bekleyin if (!display.begin(i2c\_Address, true)) { // Adresi 0x3C olarak başlat

Serial.println(F("OLED ekran bulunamadi")); while (1);

}

Serial.println(F("OLED ekran basariyla baslatildi"));

display.clearDisplay();

/\*-------------- OLED Ekranı Bitiş-------------- \*/

// Initialize ESP8266 using hardware serial

Serial1.begin(115200);

Serial1.println("AT");

Serial.println("Sending AT command to check ESP8266");

display.clearDisplay();

display.setTextSize(1); // text size '1' display.setTextColor(SH110X\_WHITE); // text color display.setCursor(0, yPos); // Started position yPos += 16;

display.println(F("Sending AT command to check ESP8266")); display.display();

while (!Serial1.find("OK")) {

Serial1.println("AT");

Serial.println("ESP8266 not found. Retrying..."); display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("ESP8266 not found. Retrying...")); display.display(); yPos += 16;

if (yPos > SCREEN\_HEIGHT - 10) { yPos = 0;

display.clearDisplay();

}

delay(1000);

} yPos = 0;

display.clearDisplay();

Serial.println("ESP8266 found and ready"); display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("ESP8266 found and ready")); display.display(); yPos += 16;

Serial1.println("AT+CWMODE=1"); while (!Serial1.find("OK")) { Serial1.println("AT+CWMODE=1");

Serial.println("Setting ESP8266 mode to client...");

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("Setting ESP8266 mode to client...")); display.display(); yPos += 16;

if (yPos > SCREEN\_HEIGHT - 10) { yPos = 0;

display.clearDisplay();

}

delay(1000);

}

Serial.println("ESP8266 set to client mode");

yPos = 0;

display.clearDisplay();

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("ESP8266 set to client mode")); display.display();

Serial.println("Connecting to Wi-Fi...");

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("Connecting to Wi-Fi...")); display.display(); yPos += 16;

Serial1.println("AT+CWJAP=\"" + networkAddressName +

"\",\"" + networkPassword + "\""); while (!Serial1.find("OK")) {

Serial.println("Failed to connect to Wi-Fi. Retrying...");

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("Failed to connect to Wi-Fi. Retrying...")); display.display();

yPos += 16;

if (yPos > SCREEN\_HEIGHT - 10) { yPos = 0;

display.clearDisplay();

} delay(2000);

}

Serial.println("Connected to Wi-Fi"); yPos = 0;

display.clearDisplay();

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("Connected to Wi-Fi")); display.display(); yPos += 16;

dht.begin();

Serial.println("DHT sensor initialized.");

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("DHT sensor initialized.")); display.display(); yPos += 16;

mod.begin(9600); pinMode(RE, OUTPUT); pinMode(DE, OUTPUT);

Serial.println("NPK sensor initialized."); display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("NPK sensor initialized.")); display.display(); yPos += 16;

sensors.begin();

Serial.println("DS18B20 sensor initialized."); display.setTextSize(1); display.setTextColor(SH110X\_WHITE);

display.setCursor(0, yPos); display.println(F("DS18B20 sensor initialized.")); display.display();

yPos = 0;

display.clearDisplay();

pinMode(SOIL\_WATER, INPUT);

Serial.println("SOIL\_WATER sensor initialized.");

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos);

display.println(F("SOIL\_WATER sensor initialized.")); display.display(); yPos += 16;

// pinMode(PUMP, OUTPUT);

// Serial.println("PUMP sensor initialized.");

// digitalWrite(PUMP, HIGH);

// display.setTextSize(1);

// display.setTextColor(SH110X\_WHITE);

// display.setCursor(0, yPos);

// display.println(F("PUMP sensor initialized."));

// display.display();

// yPos += 16;

Serial.println("All sensors initialized.");

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, yPos); display.println(F("All sensors initialized.")); display.display();

delay(2000);

display.clearDisplay();

}

void loop() {

Serial1.println("AT+CIPSTART=\"TCP\",\"" + ipAddress +

"\",80");

if (Serial1.find("Error")) {

Serial.println("Failed to start TCP connection"); delay(2000); return;

}

if(getDatasSW == false){ Serial.print(getDatas()); getDatasSW = true;

}

sensors.requestTemperatures(); weatherTemperature = dht.readTemperature(); moistureSensorData = dht.readHumidity(); gasSensorData = analogRead(GAS\_MQ); waterRainSensorData = analogRead(WATER); insideSoilTemperature = sensors.getTempCByIndex(0); soilWaterSensorData = analogRead(SOIL\_WATER);

nitrogenData = nitrogen(); delay(250);

phosphorousData = phosphorous(); delay(250);

potassiumData = potassium(); delay(250);

yPos = 0;

display.setTextSize(1); // Metin boyutu 1 display.setTextColor(SH110X\_WHITE); // Beyaz metin rengi display.setCursor(0, 0); // Başlangıç pozisyonu

display.print(F("Temp : ")); display.println(weatherTemperature); display.print(F("ISoil: ")); display.println(insideSoilTemperature); display.print(F("Mois : ")); display.println(moistureSensorData); display.print(F("Rain : ")); display.println(waterRainSensorData); display.print(F("WSoil: ")); display.println(soilWaterSensorData); display.print(F("Gas : ")); display.println(gasSensorData); display.setCursor(80, 0);

display.print(F("N: ")); display.println(nitrogenData); display.setCursor(80, 14); display.print(F("P: ")); display.println(phosphorousData); display.setCursor(80, 28); display.print(F("K: ")); display.println(potassiumData);

Serial.println("--------------------------");

Serial.print("Weather Temperature: ");

Serial.print(weatherTemperature);

Serial.println("C ");

Serial.print("Inside Soil Temperature: ");

Serial.print(insideSoilTemperature);

Serial.println("C ");

Serial.print("Moisture Sensor Data: ");

Serial.println(moistureSensorData);

Serial.print("Gas Sensor Data: ");

Serial.println(gasSensorData);

Serial.print("Water Rain Sensor Data: ");

Serial.println(waterRainSensorData);

Serial.print("Soil Water Sensor Data: ");

Serial.println(soilWaterSensorData);

Serial.println("--------------------------");

Serial.print("Nitrogen: ");

Serial.print(nitrogenData);

Serial.println(" kg/ha");

Serial.print("Phosphorous: ");

Serial.print(phosphorousData);

Serial.println(" kg/ha");

Serial.print("Potassium: ");

Serial.print(potassiumData);

Serial.println(" kg/ha");

String datas = "GET

https://api.thingspeak.com/update?api\_key=VOM7OMIE8X1V1F0M "; //Thingspeak komutu. Key kısmına kendi api keyimizi yazıyoruz. datas += "&field1=";

datas += String(weatherTemperature); datas += "&field2=";

datas += String(moistureSensorData);

datas += "&field3=";

datas += String(soilWaterSensorData); datas += "&field4=";

datas += String(waterRainSensorData); datas += "&field5=";

datas += String(insideSoilTemperature); datas += "&field6="; datas += String(nitrogenData); datas += "&field7="; datas += String(phosphorousData); datas += "&field8="; datas += String(potassiumData); datas += "\r\n\r\n";

Serial1.print("AT+CIPSEND="); Serial1.println(datas.length() + 2); delay(2000); if (Serial1.find(">")) {

Serial1.print(datas);

Serial.println("Data sent to Thingspeak: " + datas);

if(check){ sending = 0;

} else { sending = 1;

}

check = !check;

display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(0, 55); display.print(F("Send: ")); display.println(sending); display.setCursor(64, 55); display.print(F("Wifi")); display.display(); display.clearDisplay(); delay(1000);

} else {

Serial.println("ESP8266 not ready to send data"); display.setTextSize(1); display.setTextColor(SH110X\_WHITE); display.setCursor(20, 55); display.print(F("Wifi problem...")); display.display(); display.clearDisplay();

}

Serial.println("Closing connection");

Serial1.println("AT+CIPCLOSE");

if(soilWaterSensorData >= 700){ digitalWrite(PUMP, LOW);

//delay(2000);

} delay(2000);

} float nitrogen() { return readSensor(nitro);

}

float phosphorous() { return readSensor(phos);

} float potassium() { return readSensor(pota);

}

float readSensor(const byte\* command) { digitalWrite(DE, HIGH); digitalWrite(RE, HIGH); delay(10);

mod.write(command, 8);

digitalWrite(DE, LOW); digitalWrite(RE, LOW); delay(10);

unsigned long startTime = millis();

int index = 0;

while (index < 7 && millis() - startTime < 1000) { if (mod.available()) { values[index++] = mod.read();

}

}

if (index == 7) {

for (int i = 0; i < 7; i++) { Serial.print(values[i], HEX);

Serial.print(" ");

}

Serial.println(); return values[4] \* 10.0;

} else {

Serial.println("Error: Incomplete response"); return 0;

}

}

String getDatas() {

String rest = "AT+CIPSEND=90"; rest += "\r\n"; sendData(rest, 2000, 0); //Gönderilecek Karakter Sayısı.

( "AT+CIPSEND=90" )

String hostt = "GET

/channels/2576973/feeds.json?api\_key=COGQ8EN5OM4Q0S67"; hostt += "\r\n";

hostt += "Host:api.thingspeak.com"; hostt += "\r\n\r\n\r\n\r\n\r\n";

String Altin = sendData(hostt, 2000, 1); // GET request ( GET

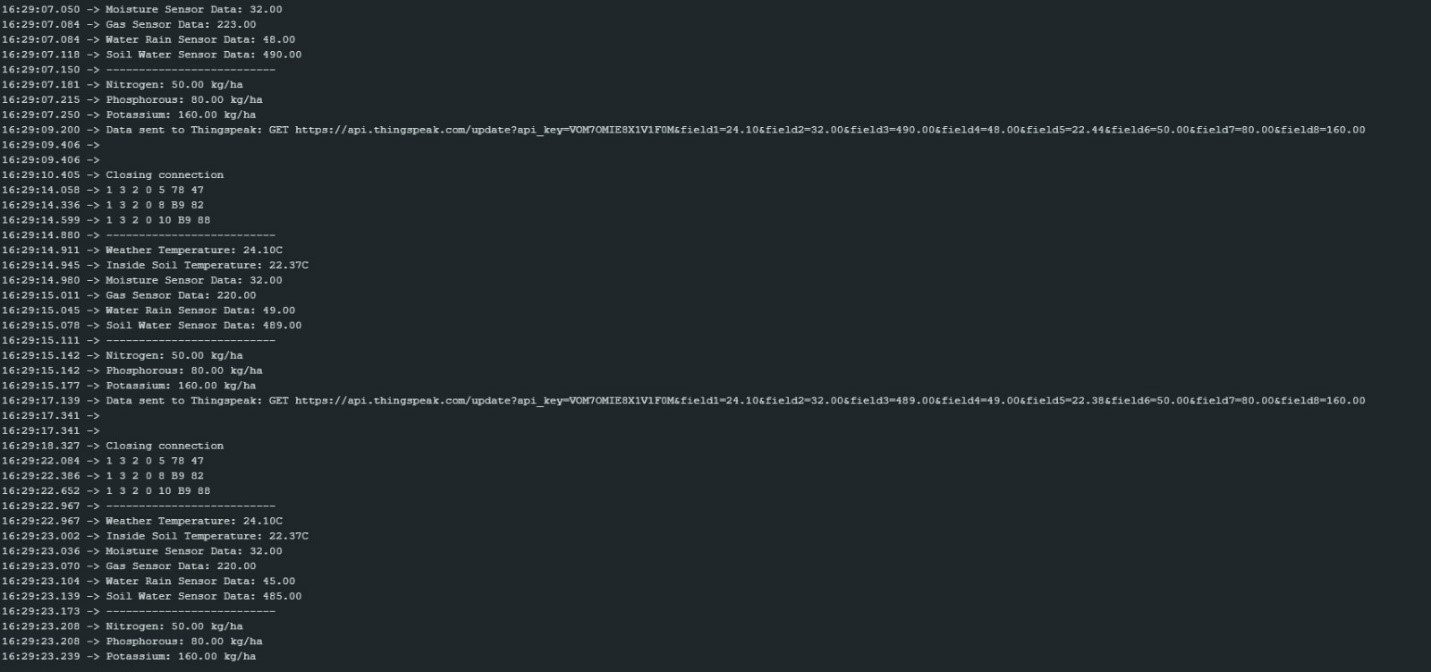
/apps/thinghttp/send\_request?api\_key=XXXXXXXXXXXXXXXX

//

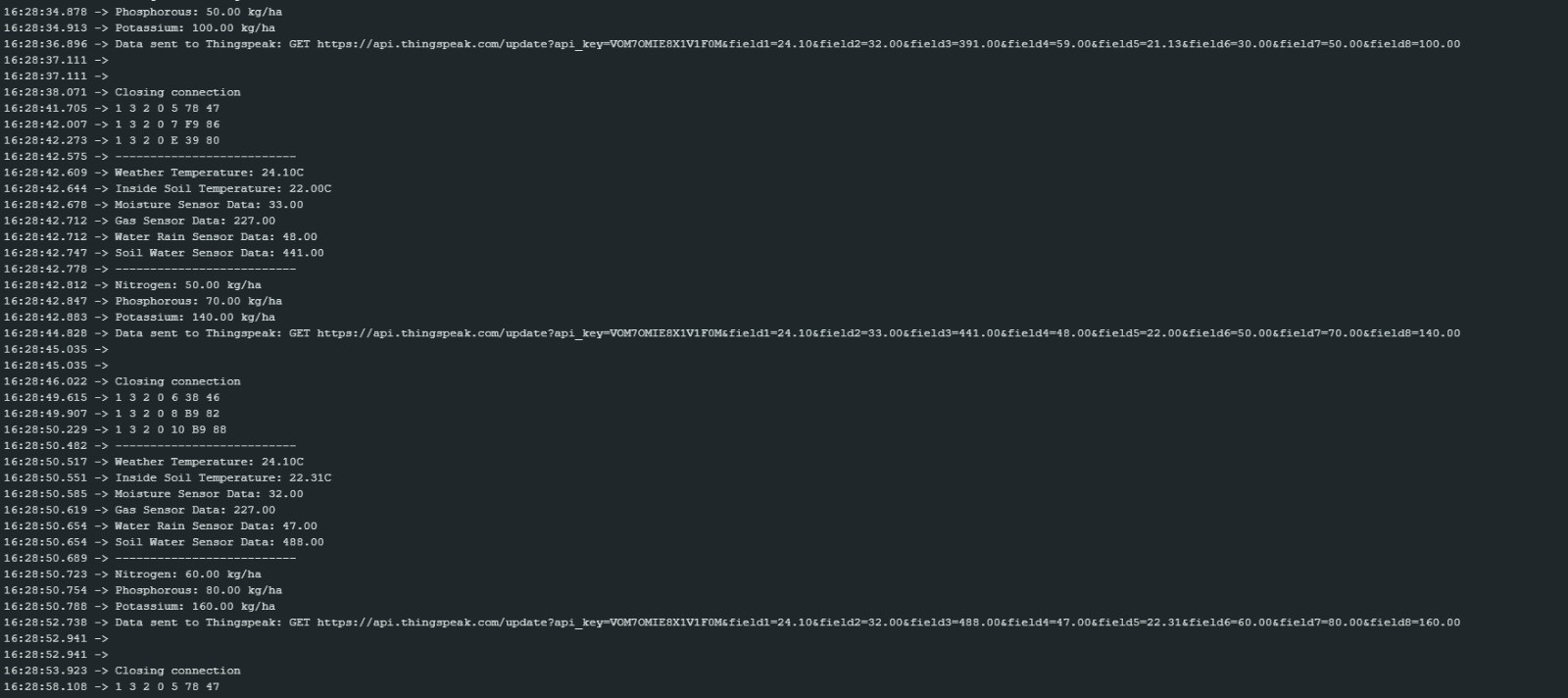
Host: Host\_server\_name )

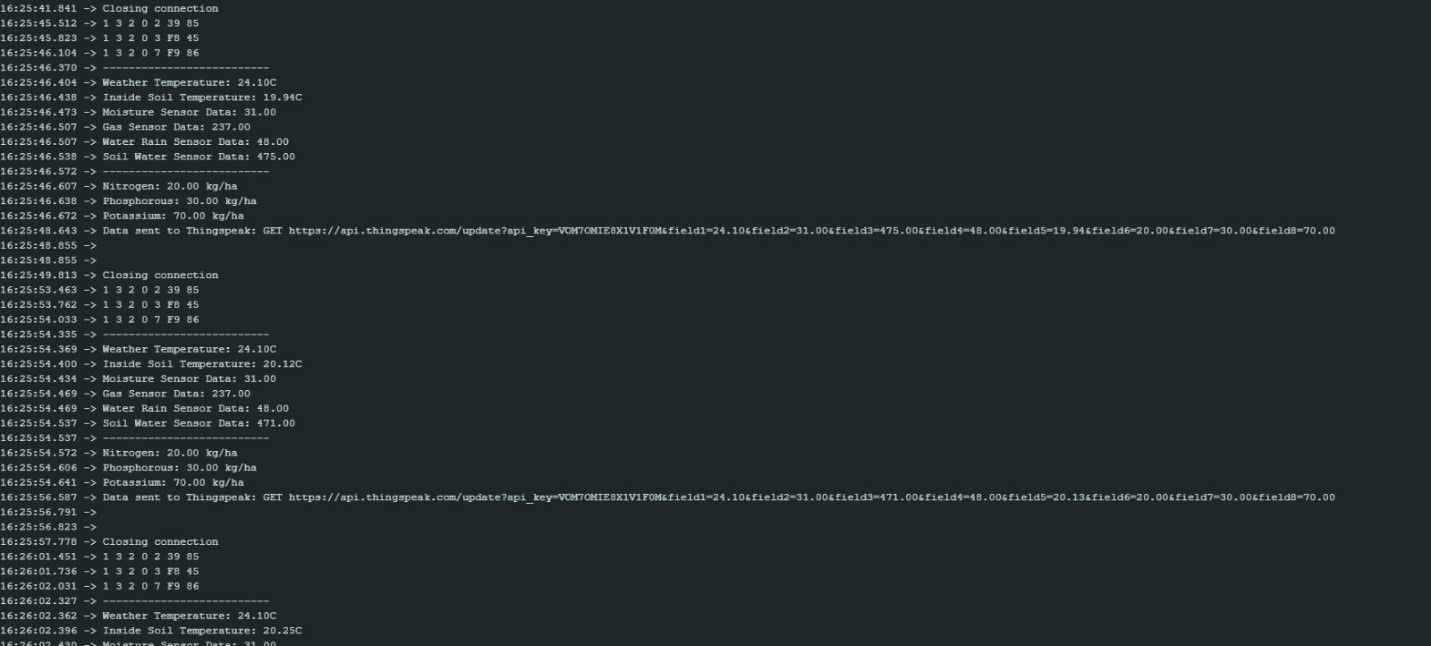
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* gelen verinin içinden sadece ilgili bölümü alıyoruz. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

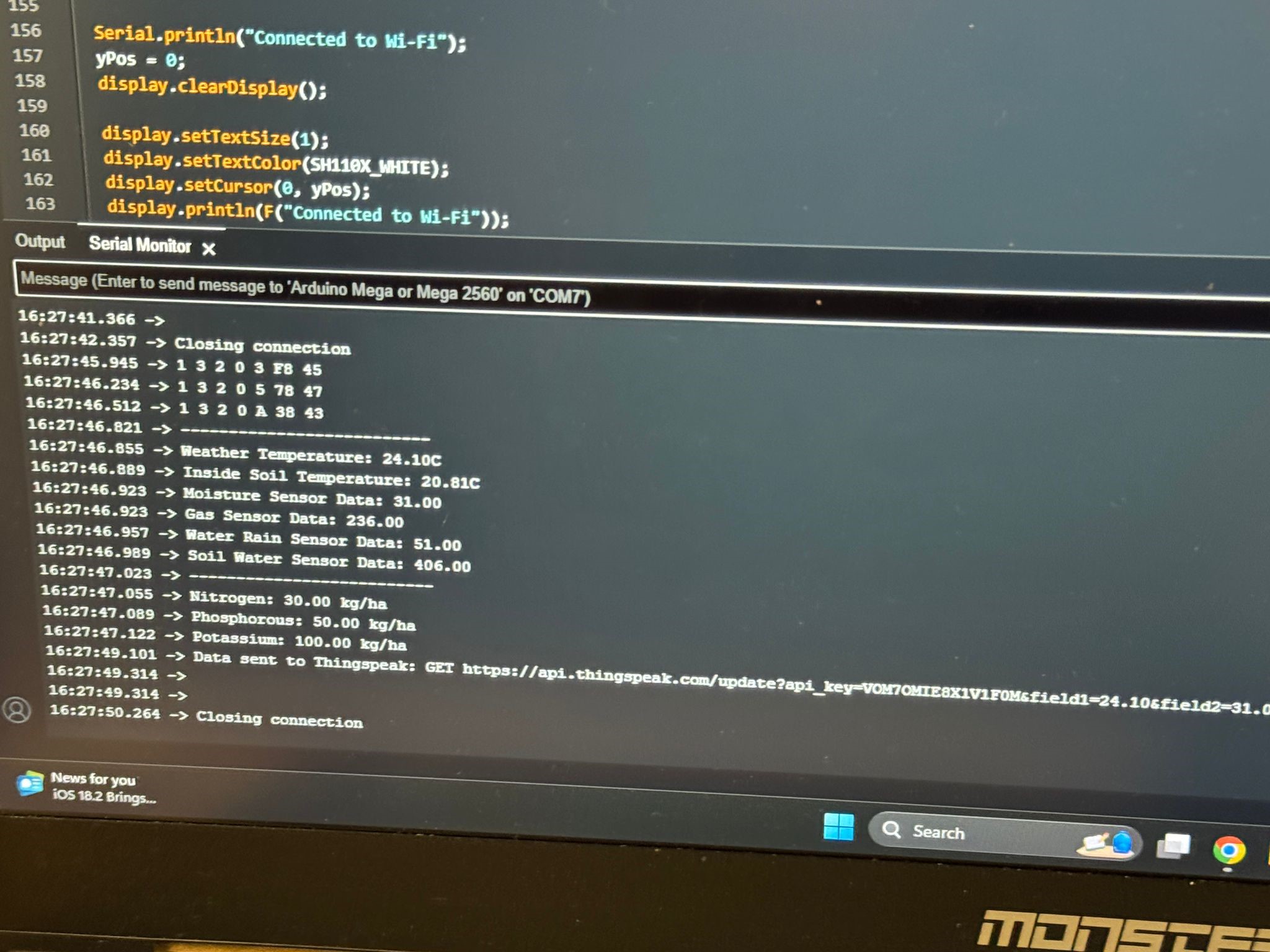
|  |
| --- |
| int baslangic=Altin.indexOf(':');  Altin=Altin.substring(baslangic+1,baslangic+9);    return (Altin);  }  String sendData(String command, const int timeout, boolean debug) {  String response = "";  Serial1.print(command);    long int startTime = millis();  while ((millis() - startTime) < timeout) { while (Serial1.available()) { char c = Serial1.read(); response += c;  }  }  if (debug) {  Serial.println(response);  }  return response;  } |

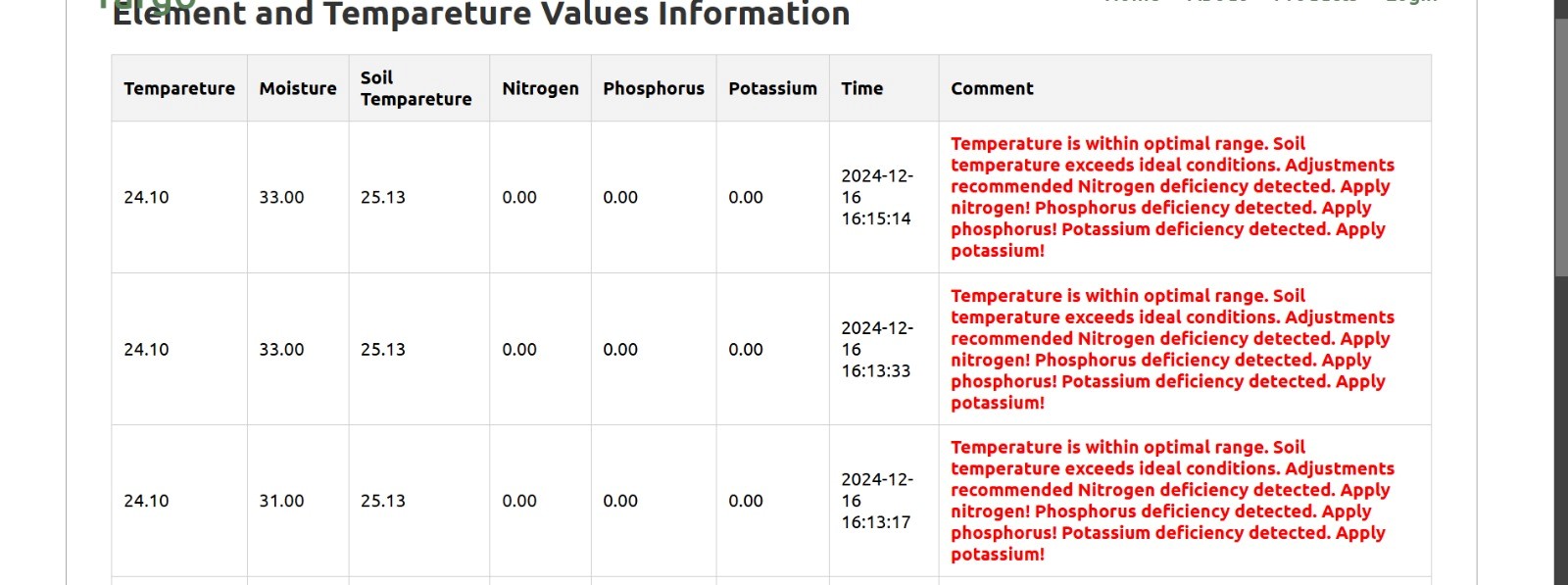


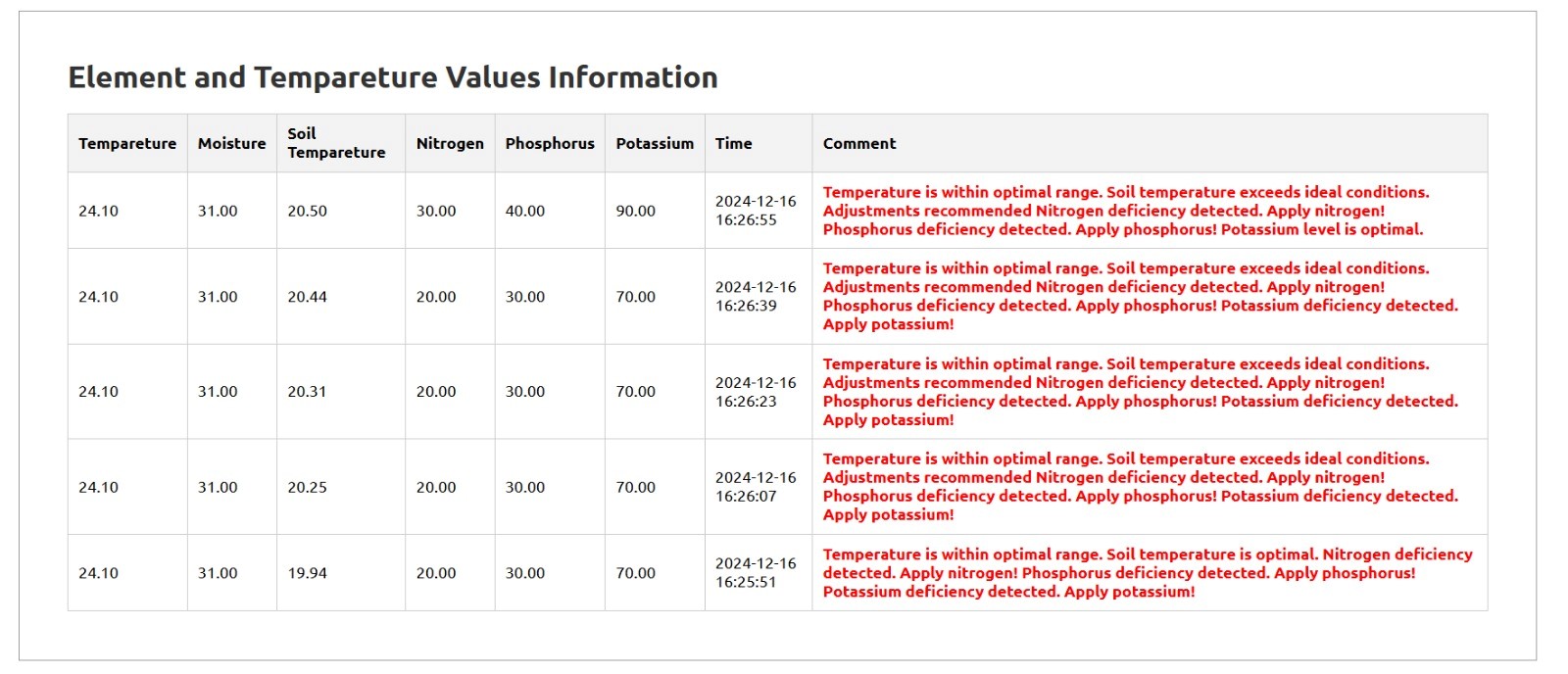




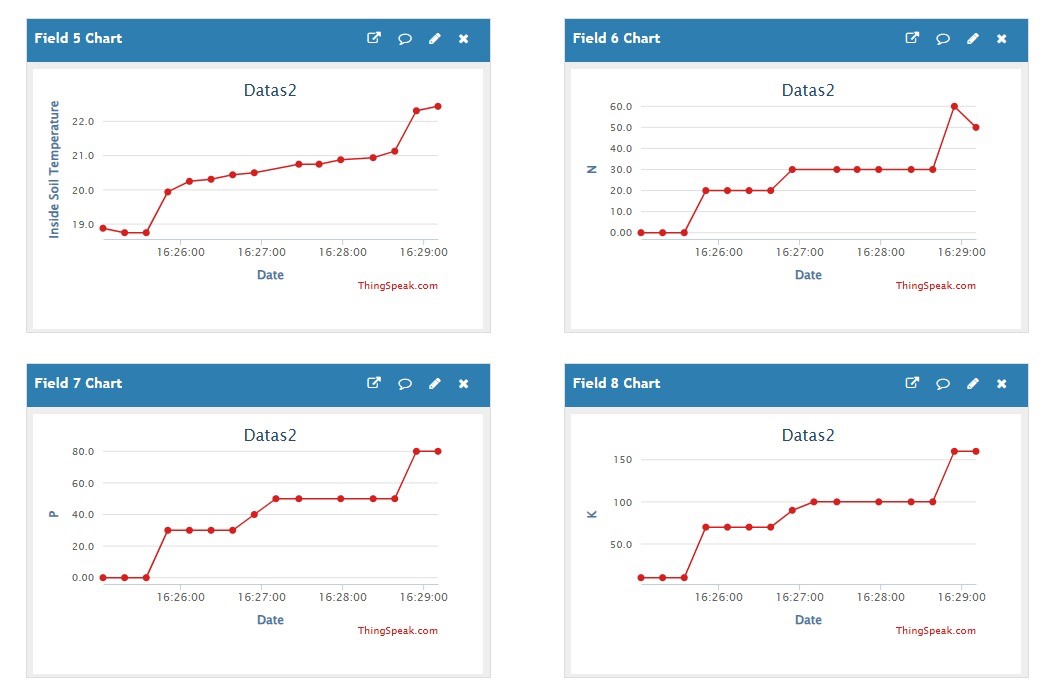


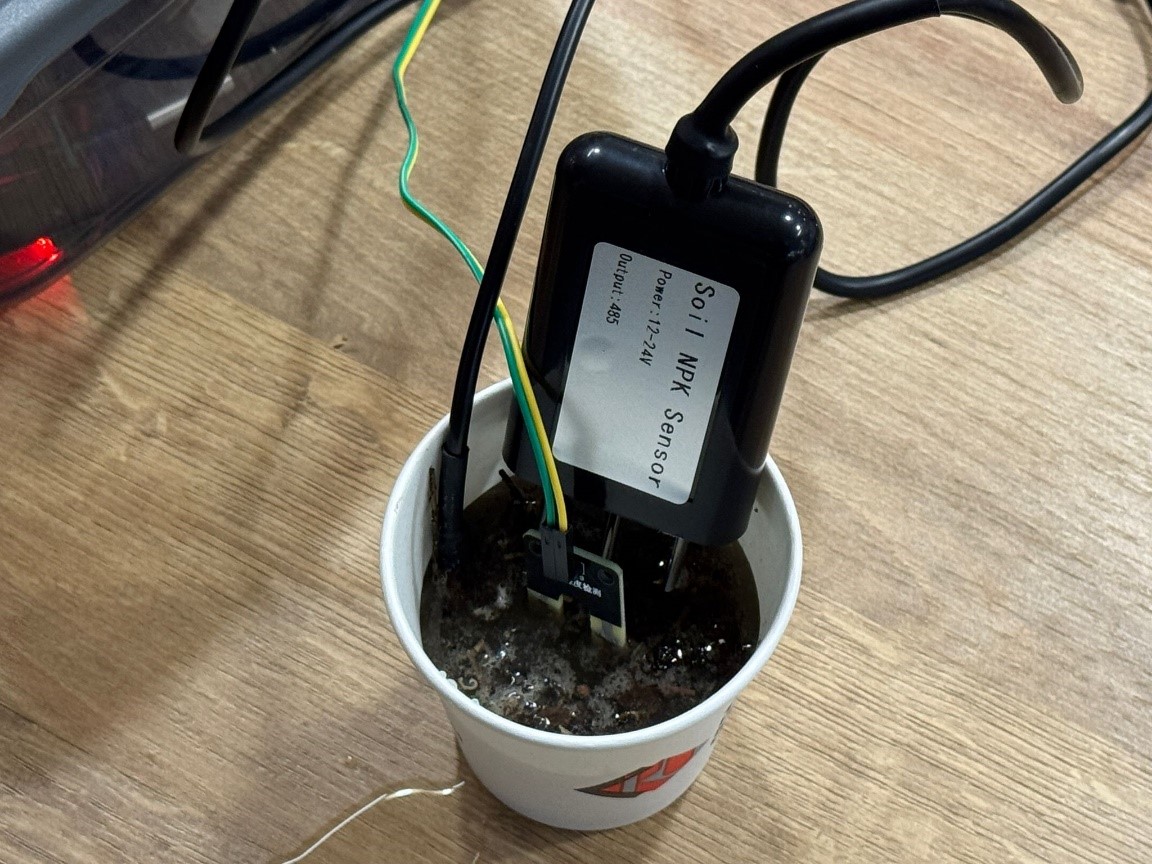




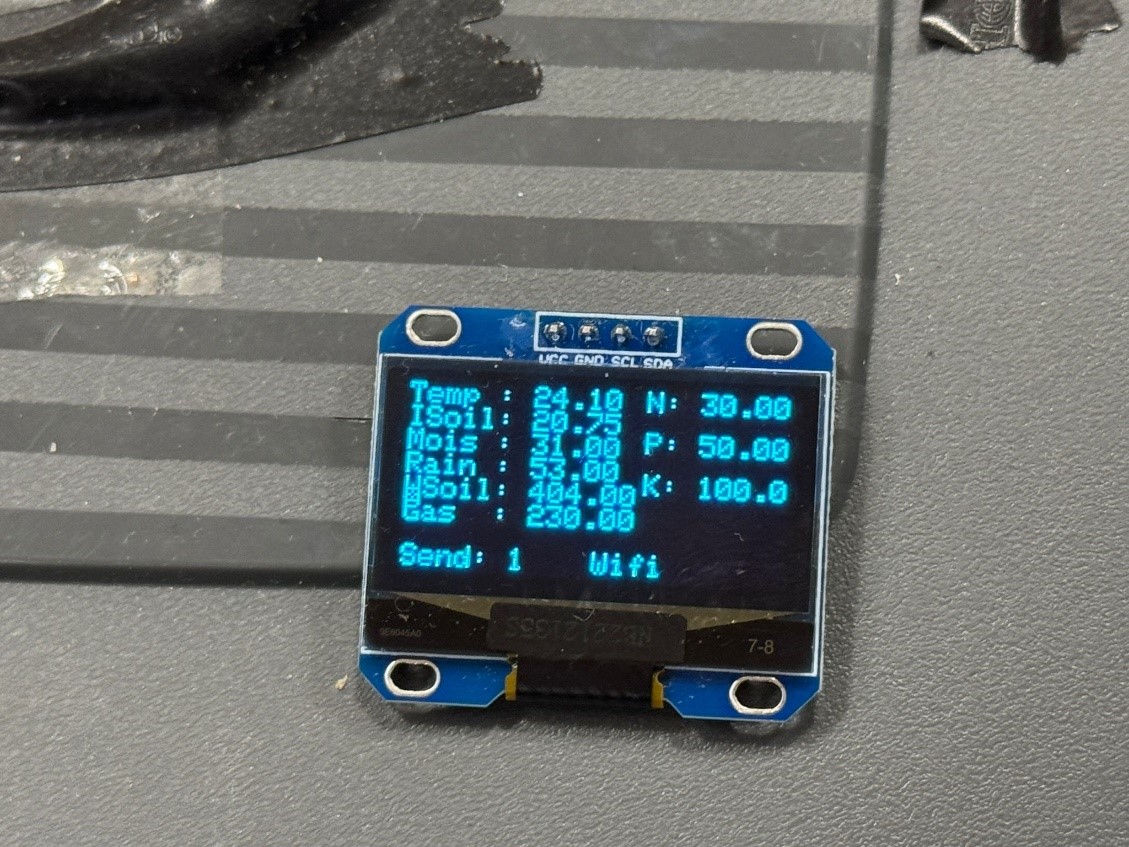




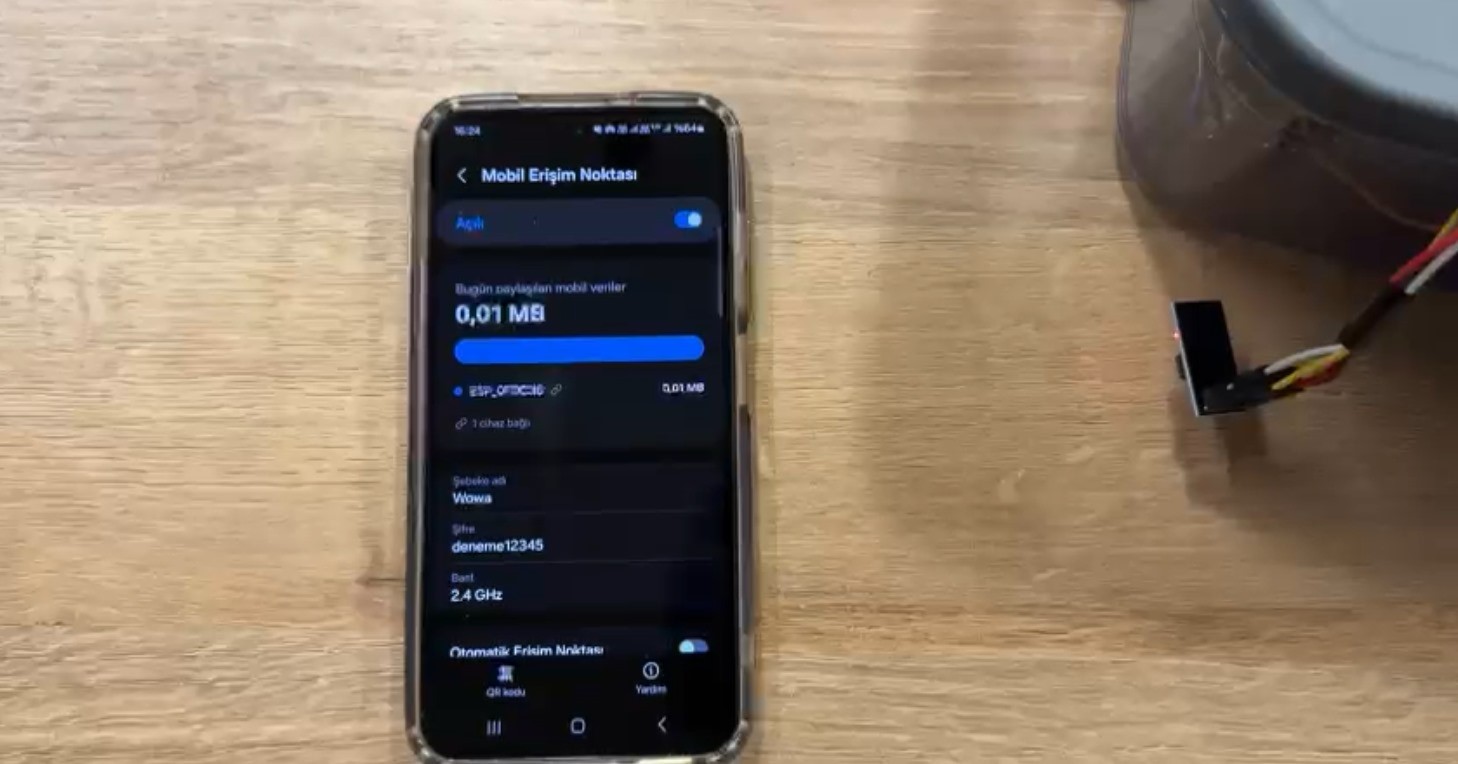




NPK sensor in a cup full of soil measures data



Results on OLED Display



The data obtained from the soil is sent to the remote server via Wi-Fi connection.



Box with Arduino mega, breadboard and sensor connections.