Erasmus+ University of Macedonia

Francesco Scavello

2021 - IoT Related projects

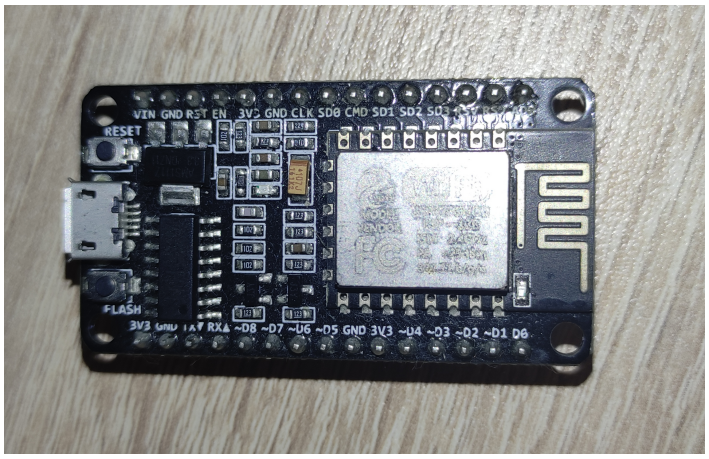
Complete Name: Francesco Scavello
Hometown institutional E-mail: francesco.scavello@studio.unibo.it
Host institutional E-mail: dai21902@uom.edu.gr
Personal E-mail: francesco.scavello99@gmail.com
Hometown institutional Student ID: 0000900942
Host institutional Student ID: dai21902

This .pdf has been written in Latex and compiled with pdfLaTeX

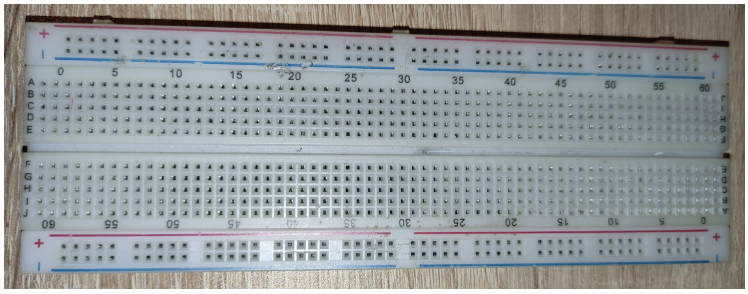
1 Communications systems Crop Farming Control System

1.1 Materials

- Arduino based microcontroller / ESP Based (8266)

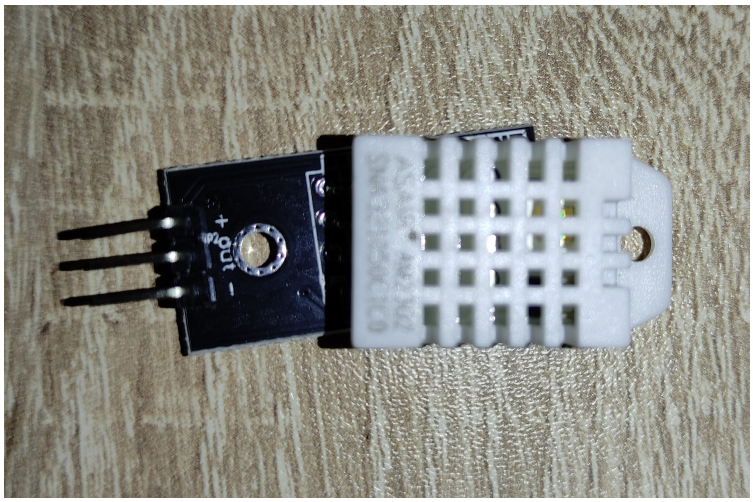


- BreadBoard



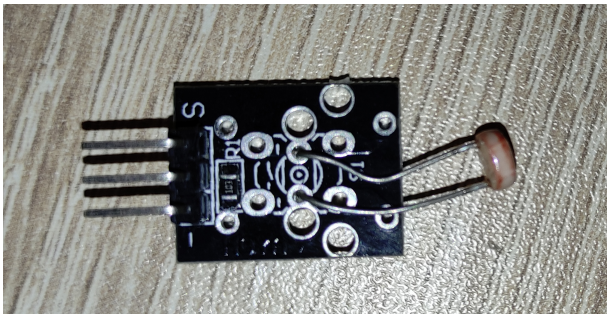
- Jumper Cables

- Dht22 humidity and temperature module

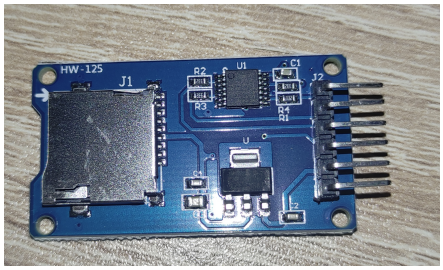


- or High quality thermometer

- Photoresistor module

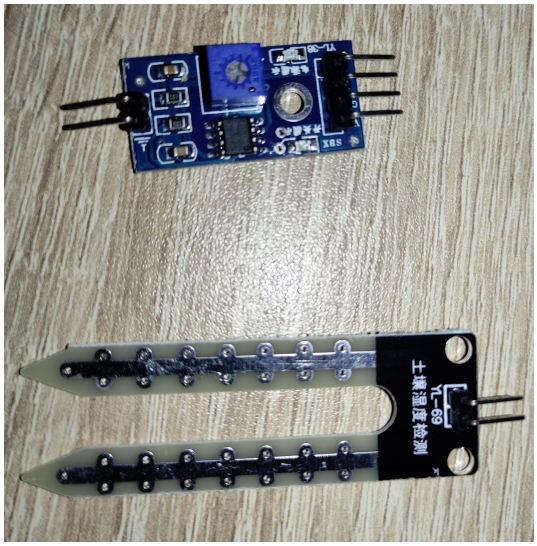


- Sd card module

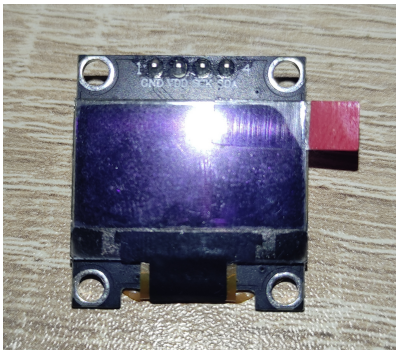


- Sd card

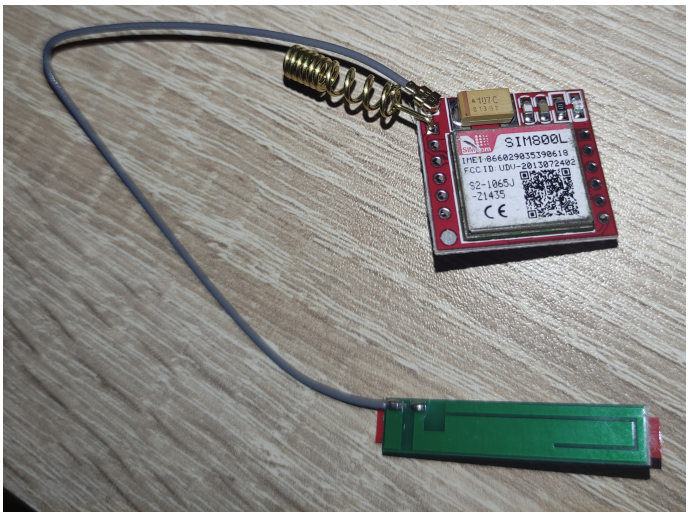
- Soil moisture modules (Low vs High quality)



- LCD (.96 OLED)



- SIM card + SIM expansion module



1.2 Description

Ideally the entire system is embedded into an electric cable pit.
The idea is a on-field crop monitor system (the power source cold be also even more green/based using amsolar panel).
The environment is controlled by an arduino based microcontroller, we keep track on a sd card of:

- Temperature
- Humidity
- Light

- Soil humidity

We have an in-loco lcd terminal and in case of cardinal temperatures "jump" or favorable conditions for the development of phytopathogenic diseases we send an sms to the farmer.
It is also possible to read data through a serial console in loco.
Ps: I want to remember, that every plant has different cardinal temperatures/values, so I suggest to choose the correct temperature for one or a specific kind of crops.

1.3 Code

```
#include "DHT.h" // DTH lib
#include <SPI.h> // serial interface lib
#include <SD.h> // sd card lib
#include <SoftwareSerial.h> // serial soft lib
#include <Adafruit_GFX.h>  // Include core graphics library for the display
#include <Adafruit_SSD1306.h>  // Include Adafruit_SSD1306 library to drive the display
//#include <Fonts/FreeMono9pt7b.h>  // Add a custom font
#define DHTPIN 14 // Pin to read sensor
#define DHTTYPE DHT22 // We use a DTH11
#define SDPIN 10 // Pin to read Photoresistor
#define PHOTORESISTORPIN 9 // Pin to read Photoresistor
#define SOILMOISTUREPIN 8 // Pin to read Photoresistor
#define Tx 3 // Pin Tx for SIM800L
#define Rx 2 // Pin Rx for SIM800L
void updateSerial();

// important variables
File myFile;
DHT dht(DHTPIN, DHTTYPE);
SoftwareSerial simSerial(Tx, Rx);
Adafruit_SSD1306 display(128, 64);  // Create display

// initialing variable here improve dynamic memory management
float h, t, m, l;
int sentinel = 2;
int condition = 0;
String humidity, temperature, light, soilMoisture, toSave;

void setup() {

  // used as a max sms
  sentinel = 1;

  // start serial
```

```
  Serial.begin(9600);
  Serial.println("Initializing");
  simSerial.begin(9600);

  // start display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // Initialize display with the I2C address of

  display.clearDisplay();  // Clear the buffer

  display.setTextColor(WHITE);  // Set color of the text

  display.setRotation(0);  // Set orientation. Goes from 0, 1, 2 or 3

  display.setTextWrap(false);  // By default, long lines of text are set to automatically \w

  display.dim(0);  //Set brightness to max

  // check on sd connection
  if (!SD.begin(SDPIN)) {
    Serial.println("SD initialization failed!");
  }

  simSerial.println("AT"); //Once the handshake test is successful, it will back to OK
  updateSerial();
  simSerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  simSerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with country code and xxxxxxxxx
  updateSerial();

  // init dht
  dht.begin();
}

void loop() {
  // we read data
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float l = analogRead(PHOTORESISTORPIN);
  float m = analogRead(SOILMOISTUREPIN);

  // we check for variable missing
  if (isnan(h) || isnan(t) || isnan(l) || isnan(m)) {
    Serial.println("Failed to read from sensors!");
  }

  // we create a string containing the data to send
```

```
humidity = "Humidity: " + String(h) + "\n";
temperature = "Temperature: " + String(t) + "\n";
light = "Light: " + String(l) + "\n";
soilMoisture = "Soil Moisture: " + String(m) + "\n";
String toSave = humidity + temperature + light + soilMoisture;

// print to serial
Serial.print(toSave);

// open file
myFile = SD.open("data.txt", FILE_WRITE);


if (myFile) {

  // save on file
  Serial.print("Writing to test.txt...");
  myFile.println(toSave);

  // close the file:
  myFile.close();

  Serial.print("Success writing to SD");

} else {

  // if the file didn't open
  Serial.println("error");

}

if(t < 15){
  condition = 1;
}

// check for max sms and conditions
if(condition&&(sentinel>0)){
    simSerial.print(""); //text content
    updateSerial();sentinel--;
    simSerial.write(26); // 26 = Ctrl + z in ASCII
 }

display.clearDisplay();  // Clear the display so we can refresh

//display.setFont(&FreeMono9pt7b);  // Set a custom font
//display.setTextSize(0);  // Set text size
```

```
  // Print text:
  display.setCursor(0, 10);  // (x,y)
  display.println("Hello"); // we can choose what to print

  // print everythin
  display.display();

  // wait 10 minutes
  delay(1000 * 60 * 10);
}


void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    simSerial.write(Serial.read());
  }
  while(simSerial.available())
  {
    Serial.write(simSerial.read());
  }
}
```

1.4 Explanation - Commenting

After the definition of the necessary Macros, inclusion of libraries and correct setup in the body of the setup() function, we mainly work into the loop() function body. We then initialize the Display, the SimModule and the serial monitor Every 10 minutes we:

1. Read data from every sensor (Temperature, Humidity, Light, Soil Moisture)
2. We make a check on the data
3. We compose the string containing the informations
4. We save the data into the sd card
5. We check if the enviromental condition are critical, if so we send a messagge to the owner
6. We then print to the external display

Result of the Verification process:

```
Sketch uses 319241 bytes (33%) of program storage space.
Maximum is 958448 bytes.
Global variables use 29784 bytes (36%) of dynamic memory, leaving 52136 bytes for local vari
Maximum is 81920 bytes.
```

1.5 Possible upgrades

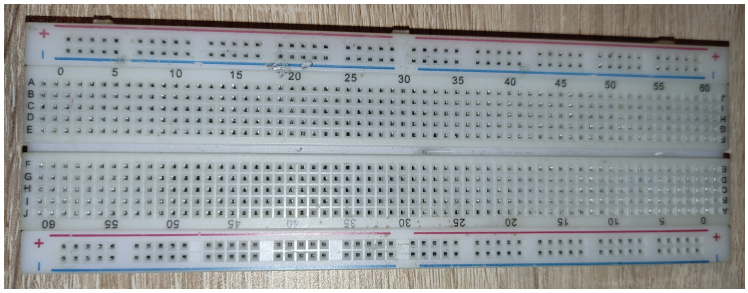
It would be possible to develop a software to read the data from the sd card and plot instantly the graphs to better lookup to the enviromental variables. Also a Solar panel module and Rain sensor could improve the utility of the project.

2 Mobile and wireless communications systems Greenhouse Control System

2.1 Materials

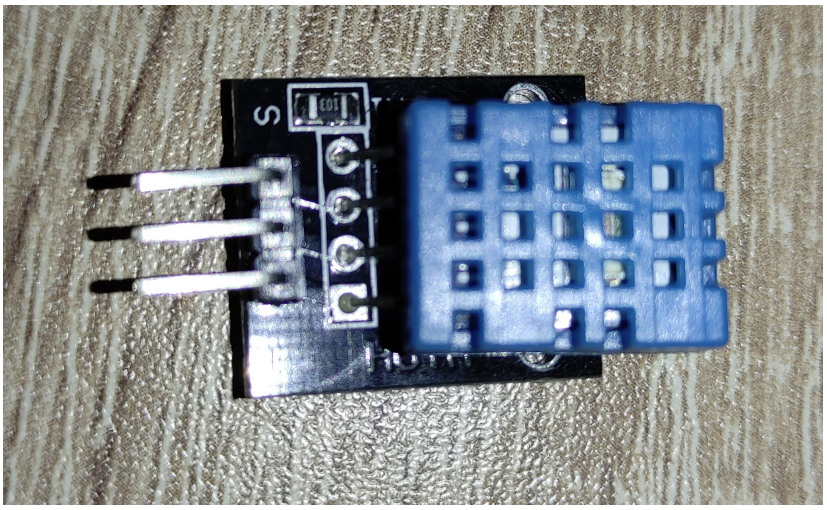
- Esp8266/32 based microchip with Wi-fi and Bluetooth

- BreadBoard

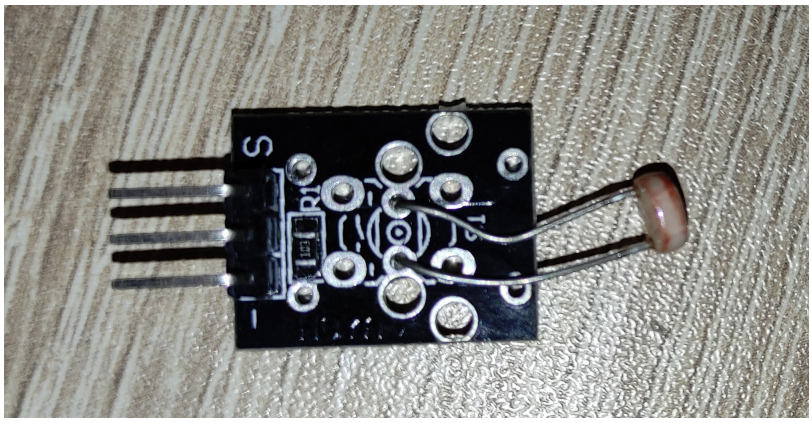


- Jumper Cables

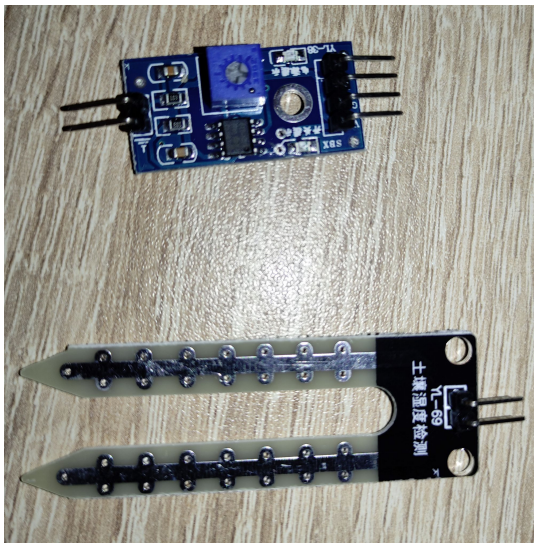
- Dht11 humidity and temperature module



- Photoresistor module



- Soil moisture module



2.2 Description

This project is a "more indoor" version of the previous one, in point of fact we use a DTH11 module (with a smaller temperature range) and an esp8266 with Wi-fi/Bluetooth to access data!
The idea for this is a home based monitor for plants.
The environment is controlled by an esp8266 based microchip, we read (and not keep record) data of:

- Temperature
- Humidity

- Light
- Soil humidity

Commands are directly programmed into the Telegram bot and every data request is in real time!
It is also possible to read data through the serial console in loco.

2.3 Code

```
#include "DHT.h" // DTH lib
#include "CTBot.h" // Telegram bot library
#include "BluetoothSerial.h" // Bluetooth library
#define DHTPIN 12 // Pin to read sensor
#define DHTTYPE DHT11 // We use a DTH11
#define PHOTORESISTORPIN 14 // Pin to read Photoresistor
#define SOILMOISTUREPIN 13 // Pin to read Photoresistor

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

CTBot myBot;
DHT dht(DHTPIN, DHTTYPE);
BluetoothSerial SerialBT;

String ssid  = "******"; // Wifi name
String pass  = "******"; // WIfi pass
String token = "******"; // Telegran token

float h, t, l, m;
String humidity, temperature, light, soilMoisture, toSend;

void setup() {

  // start serial console and bluetooth
  Serial.begin(115200);
  Serial.println("Starting TelegramBot...");
  SerialBT.begin("ESP32");

  // connect the ESP8266 to the desired access point
  myBot.wifiConnect(ssid, pass);

  // set the telegram bot token
  myBot.setTelegramToken(token);

  // check if all things are ok
```

```
  if (myBot.testConnection())
    Serial.println("\ntestConnection OK");
  else
    Serial.println("\ntestConnection NOK");

  // init dht
  dht.begin();
}

void loop() {
  // we read data
  h = dht.readHumidity();
  t = dht.readTemperature();
  l = analogRead(PHOTORESISTORPIN);
  m = analogRead(SOILMOISTUREPIN);

  // we check for variable missing
  if (isnan(h) || isnan(t) || isnan(l) || isnan(m)) {
    Serial.println("Failed to read from sensors!");
  }

  // we create a string containing the data to send
  humidity = "Humidity: " + String(h) + "\n";
  temperature = "Temperature: " + String(t) + "\n";
  light = "Light: " + String(l) + "\n";
  soilMoisture = "Soil Moisture: " + String(m) + "\n";
  toSend = humidity + temperature + light + soilMoisture;

  // print to serial
  Serial.print(toSend);

  // print to bluetooth
  SerialBT.print(toSend);

  // a variable to store telegram message data
  TBMessage msg;

  // if there is an incoming message...
  if (myBot.getNewMessage(msg)){
    // ...forward it to the sender
    if(msg.text.equalsIgnoreCase("/data")){
      myBot.sendMessage(msg.sender.id, toSend);
    }
    else{
      myBot.sendMessage(msg.sender.id, msg.text);
    }
```

```
  }

  // wait 2000 milliseconds - 2 secs
  delay(2000);
}
```

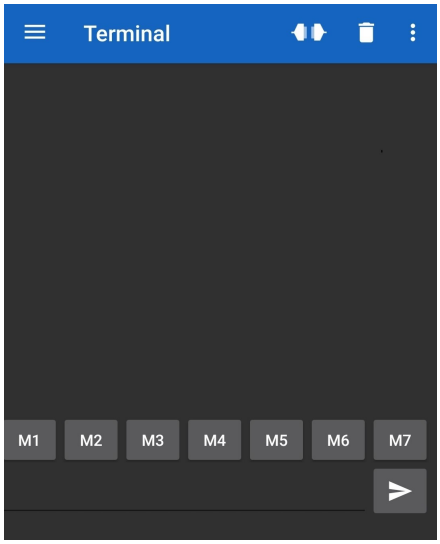
2.4 Explanation - Commenting

After the definition of the necessary Macros, inclusion of libraries and correct setup in the body of the setup() function, we mainly work into the loop() function body.
Every 2 seconds we:

1. Read data from every sensor (Temperature, Humidity, Light, Soil Moisture)
2. We make a check on the data
3. We compose the string containing the informations
4. We send the string through Blutetooth and Serial display
5. We then check if there is a request from the telegram bot, if so we send the string

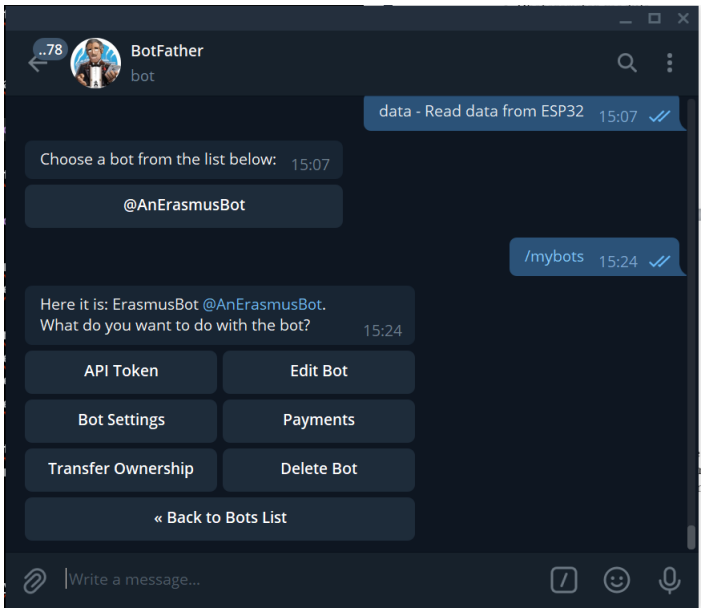
The two connectivity access interfaces are the Bluetooth one and the Telegram one:

1. The bluetooth interface can be managed easily, in this case is a simple serial bluetooth terminal app

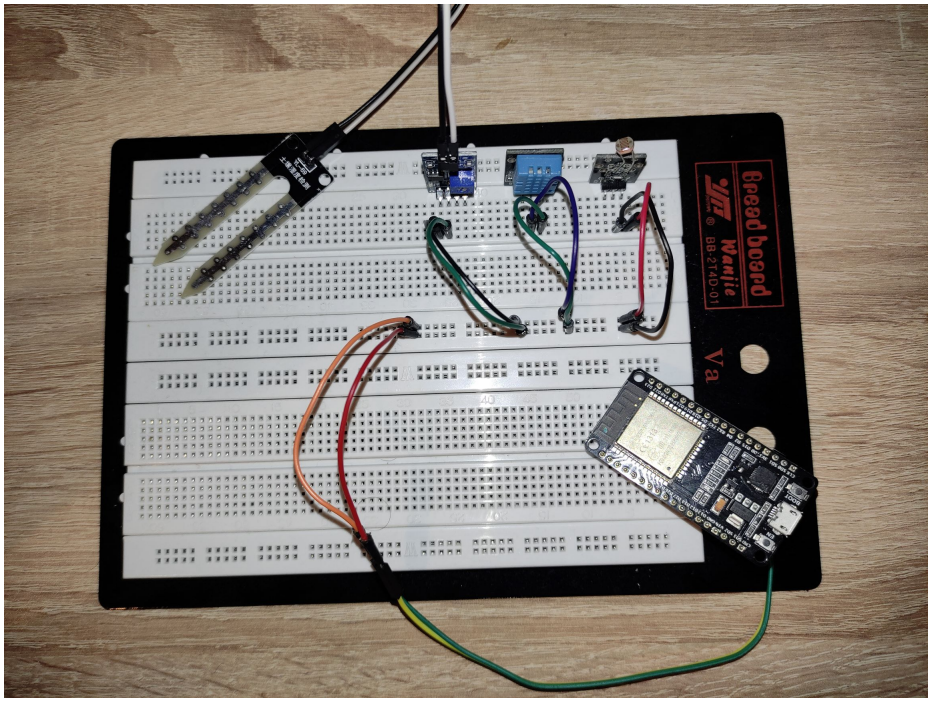


The Bluetooth configuration is handled in the setup() body, and the data is sent at every cicle.

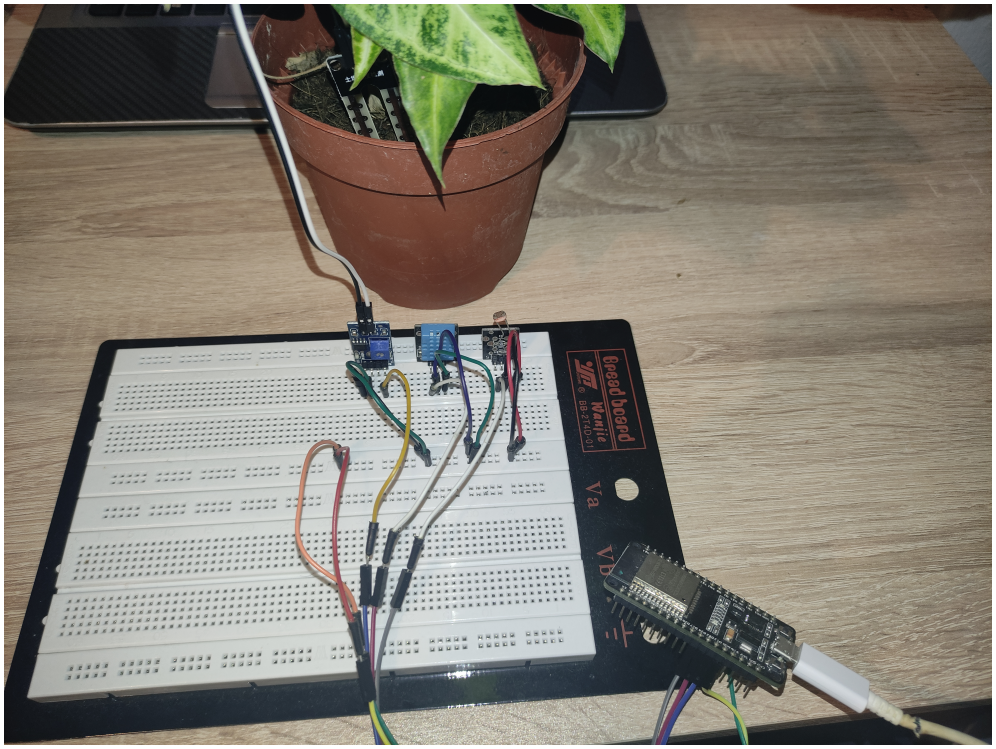
2. The Telegram interface is managed through FatherBot (The Bot API is an HTTP-based interface). We create a bot and receive a token key that we use in the code to make requests.
   In the Telegram Bot interface we create a command '/data' that allow us to contact the ESP32 and request the data.



Power supply wiring



Data Wiring



Result of the Verification process:

```
Sketch uses 1511914 bytes (48%) of program storage space.
Maximum is 3145728 bytes.
Global variables use 52224 bytes (15%) of dynamic memory,
leaving 275456 bytes for local variables.
Maximum is 327680 bytes.
```

P.S. It may be necessary (depending on the board) to swap the partition scheme ("Arduino IDE: Tools − > Partition Scheme:change "Default" to 'Huge APP (3MB No OTA)' ") because of the dimension of the code.
(https://github.com/espressif/arduino-esp32/issues/1075)

2.5 Possible upgrades

It is possible to implement the use of heating pad or a small fan to improve the ecosystem and manage the temperature/humidity.
Also an LCD screen could be added for faster lookup and a watering pump attached to a relay to manage the soil moisture.