IoT Greenhouse

03.03.2025

Caterina Gazzotti (186023)

UniMoRe, IoT Course 2023/2024

Contents

1. Introduction

Given my passion for plants I wanted to build a greenhouse that could warn me whenever there is a problem. This project's components aim to monitor temperature, humidity and light. As i only have succulents, like an Aloe and a Crassula, I decided to have certain ranges of values that are obviously different for other type of plants. The range for temperature is around 8°C minimum and 30°C maximum, humidity is a benefit for succulents so I decided that it should not be under 10%. If one of these range is not respected a sound alarm will be enabled to let the user know that something is going on, for example, the plant can be outside when its snowing. This greenhouse can be left outside the house, so two different board are needed: one to monitor values and the alarm one inside. These two will then need to communicate to each others, to do so I used the first one as an access point that can create a wifi to be used by both. The communication employs Coap protocol.

2. Hardware

Hardware components needed are:

- ESP8266 board (NodeMcu)
- ESP32 board
- buzzer (ky-006)
- LDR light sensor
- humidity and temperature sensor (DHT11)
- touch sensor (TTP223B)

I suggest you to also buy DuPont wires and two breadboards. They are used to connect sensors and actuators to the boards.

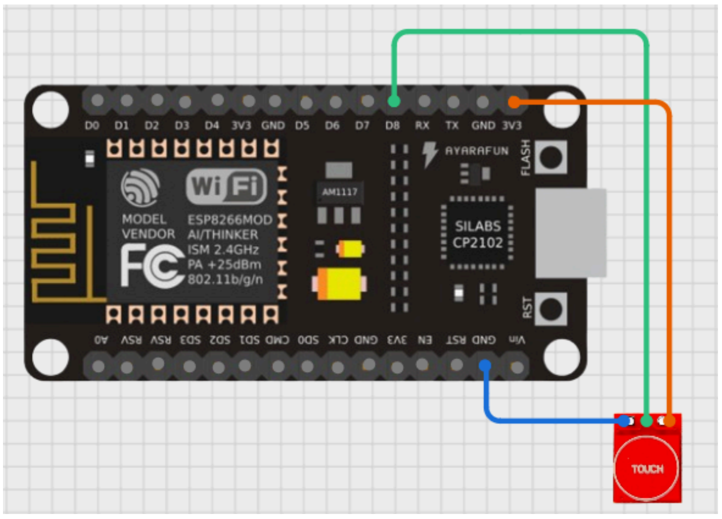
2.1. ESP8266

This board will be used for the alarm system, it will need the buzzer and a sensor to disable the sound when not needed.

2.1.1. Touch sensor

S	D8
VCC	3V
-	G

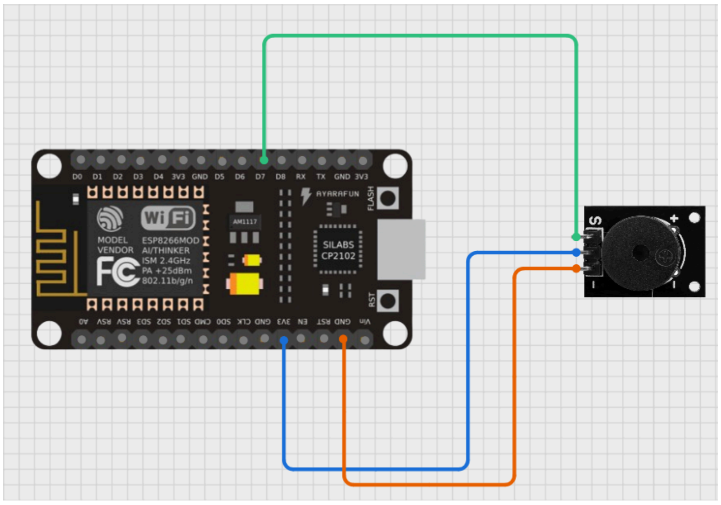
As it can be seen the esp8266 is wired with the touch sensor. The green wire allows to read signals, LOW means that nobody is touching it and HIGH means the opposite. The red wire is used to power it and the blue is connected to the ground.



2.1.2. Buzzer

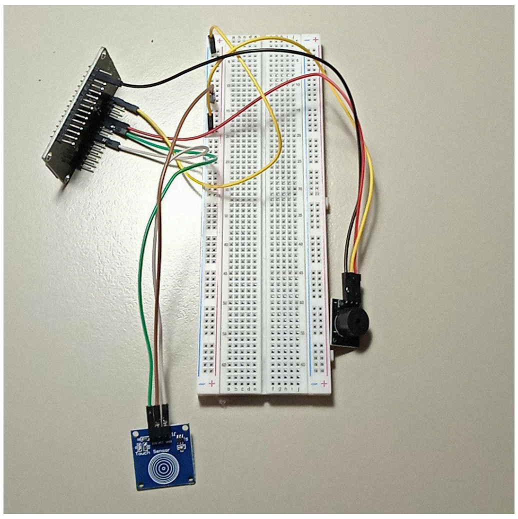
SIG	D7
VCC	3V
GND	G

As shown below the buzzer is wired on the GND with the orange line and on the power (3V) with the blue line. To pass tone/untone commands, it was connected to the board using the D7 pin.


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3V3 EN RST GND Vin
Wi Fi
MODEL ESP8266MOD
VENDOR AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
AM1117
AYARAFUN
FLASH
SILABS
CP2102
RST
S
+
-

2.1.3. Whole

The assembled system will look like the photo below:



To not use all the GND pins on the ESP8266, i decided to use a breadboard to connect all of them.

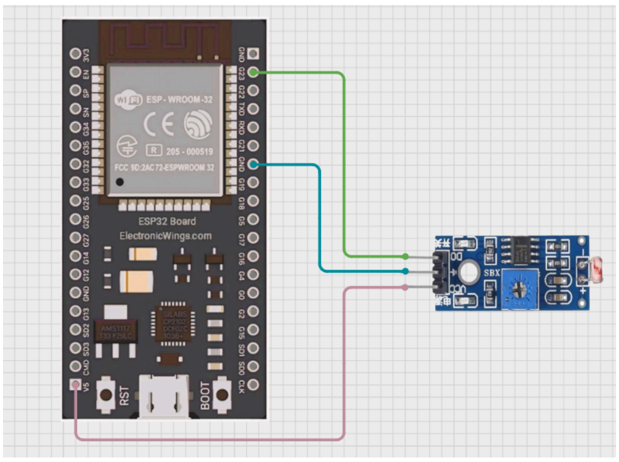
2.2. ESP32

This board will be used in the greenhouse to keep track of temperature, humidity and light values.

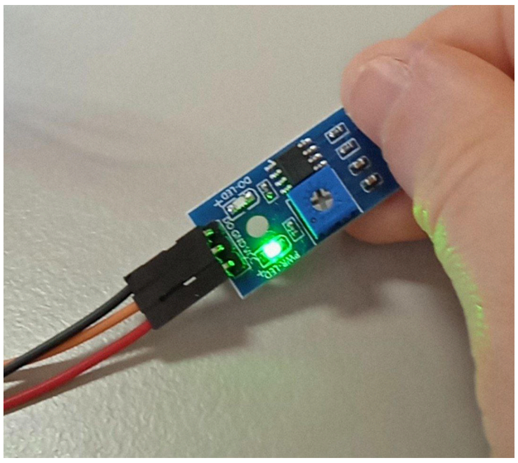
2.2.1. LDR light sensor

DO	G23
VCC	5V
GND	GND

This type of photoresistor module has analog input only. The absence of light is represented by the value 4095, lesser value instead correspond to light. I decided to use the analog pin in G23 because it was the first available both in input and output.



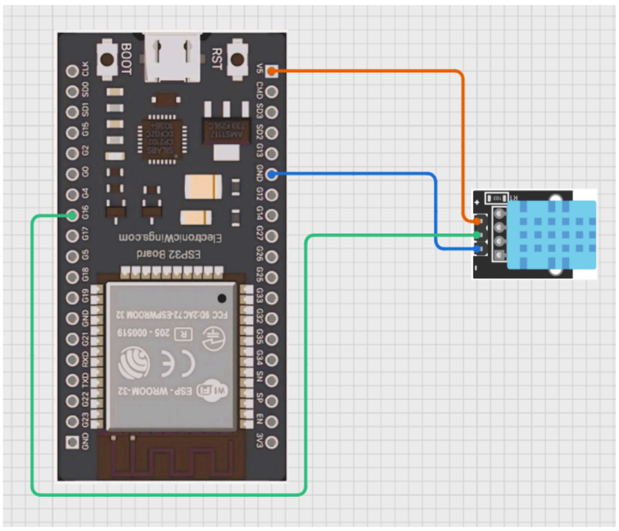
Below the photo shows me touching the photosensible sensor to turn off the second led that means light off.



2.2.2. Humidity and Temperature sensor (DHT11)

VCC	5V
DATA	G16
GND	GND

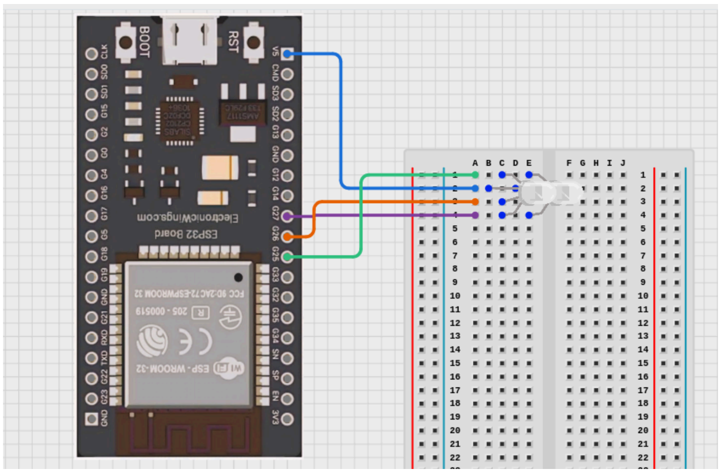
I decided to buy this type of sensor because it has both humidity and temperature data. As other sensor i simply attached its data pin to a digital pin on the board.



2.2.3. RGB led

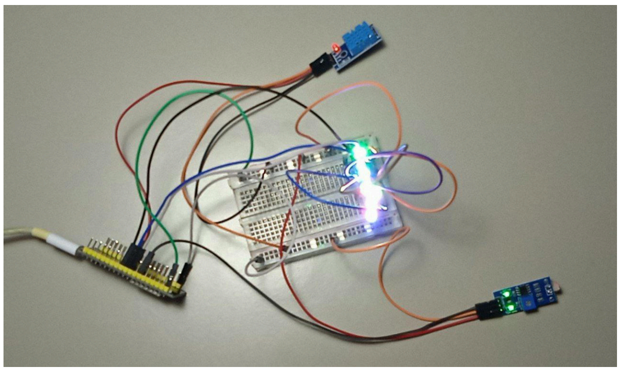
RED	G25
BLUE	G26
GREEN	G27
VCC	5V

I could not find UV leds so i inserted 7 rgb leds controlled by the same pins. These are all cathode, so they are powered up by the connection to VCC.



2.2.4. Whole

The assembled system will look like the photo below:



Because of the limited number of GND and V5 pins on ESP32 i used a breadboard as for the ESP8266.

3. Software implementation

I used some libraries downloaded with Arduino IDE, they can also be found on PlatformIO:

- coap-simple; implements Coap protocol to be used by both boards.
- DHT; library needed to read easily both temperature and humidity from the DHT11.
- WiFi; used for the ESP32 to create a wifi. WiFiUdp is already in this module.
- ESP8266WiFi; wifi support for ESP8266.
- WiFiUdp

To create and to connect to the wifi the SSID and the password will be needed. I decided to create a header file (WiFiUtils) with those credentials. Used with GitHub can also be a way to hide important information, a best practice is to make a file with passwords that will never be pushed on the repository.

3.1. Board 1 (ESP32)

This board collects data from two sensors and it sends a coap request every time values are off the range.

```
int LightPin    = 32; //GPIO32
int HumidityPin = 16; //GPIO16
// same configuration for every led
int Green = 25, Blue = 26, Red = 27;
int temp; // used to store the temperature
WiFiUDP udp;
Coap coap(udp);

// initialization for humidity and temperature sensor
DHT dht(HumidityPin, DHT11);
```

I implemented a wifi initializer after defining global variables. This function will set the board in access point mode and it will create a wifi using the passed settings (SSID and PASS).

```
void initWifi()
{
  WiFi.mode(WIFI_AP); // initialization of access point wifi
  WiFi.softAP(SSID, PASS); // using SSID and PASS written in
WiFiUtils

  Serial.print("Access Point with IP: ");
  Serial.println(WiFi.softAPIP());
}
```

This board is the client part for coap, it only makes requests and it don't expose any actuators to be used.

```
void callback_response(CoapPacket &packet, IPAddress ip, int port)
{
  Serial.print("[Coap Response got] ");

  char p[packet.payloadlen + 1];
  memcpy(p, packet.payload, packet.payloadlen);
  p[packet.payloadlen] = NULL;

  Serial.println(p);
}
```

The starting point for the board is the setup function. It is used to initialize wifi, the DHT sensor, coap client and all the RGB leds. I decided to set those leds on a whitish light because all the professional lights to grow plants suggest it to be the color for an healthy plant. A more blueish or reddish color can be used to make enlarge themselves or bloom.

```
void setup()
{
  Serial.begin(115200);
  dht.begin();
  initWifi();

  pinMode(Red, OUTPUT);
  pinMode(Green, OUTPUT);
  pinMode(Blue, OUTPUT);

  // the color is setted to white
  analogWrite(Red, 0);
  analogWrite(Green, 0);
  analogWrite(Blue, 0);

  coap.response(callback_response);
  coap.start();
}
```

Finally the loop function controls temperature and humidity values. If there is something off it will make a coap request with the put function.

```
void loop()
{
  // some test showed that the temperature was inaccurate for ~2°C
  temp = dht.readTemperature() - 2.30;
  // call for the buzzer if temp and humidity are not in the right range
  if(temp >= 30.0 || temp <= 8.0 || dht.readHumidity() <= 10.0)
    coap.put(IPAddress(192, 168, 4, 2), 5683, "buzzer", "1");

  delay(1000);
  coap.loop();
}
```

To be sure the request is sent, instead of blowing on the DHT i added an `analogRead(LightPin) == 4095`. With this part i can trigger a false allarm with my hand obscurating the light sensor.

3.2. Board 2 (ESP8266)

This part will explore how to make the buzzer stop beeping with the help of the touch sensor. Moreover, I will show how to make it start when a certain Coap message arrives from the other board. Here we will use: ESP8266WiFi, WiFiUdp and coap-simple.

As a first thing I defined global variables:

```
int TouchPin = 13; // D8 has assigned number 13
int BuzzPin  = 15; // D7 has assigned number 15
```

```
// Defines the ip address for this device
IPAddress ip(192, 168, 4, 2);
// Defines the gateway (ESP32 device)
IPAddress gateway(192, 168, 4, 1);
IPAddress subnet(255, 255, 255, 0);
WiFiUdp udp;

// Initializes coap variable with the transmission channel
Coap coap(udp);
bool bip = false; // Used to know when the buzzer should beep or not
```

I defined a function to initialize wifi. This simple method will try to connect with the given IP and credentials. It differs from board1 initWiFi function because with this one we are trying to connect to an already created wifi.

```
void initWifi()
{
  // Configuring which ip address the device should have
  WiFi.config(ip, gateway, subnet);
  // Begins to search the wifi and tries to connect with the password
  WiFi.begin(SSID, PASS);
  while(WiFi.status() != WL_CONNECTED)
  {
    delay(2000);
    Serial.println("Trying to connect...");
  }
}
```

To make the buzzer work when the client requests it, i implemented a function to be used by coap as the behavior of the server. If board1 sends a message equals to 1 then the bip variable will be setted to true and the buzzer will beep.

```
// Server coap to beep the buzzer
void callback_buzzer(CoapPacket &packet, IPAddress ip, int port)
{
  Serial.println("[Buzzer] ON");

  char p[packet.payloadlen + 1];
  memcpy(p, packet.payload, packet.payloadlen);
  p[packet.payloadlen] = NULL;

  String message(p);
  if (message.equals("1"))
  {
    bip = true;
    coap.sendResponse(ip, port, packet.messageid, "1");
  }
}
```

The setup function initializes wifi, touch pin and coap server.

```
void setup()
{
  Serial.begin(9600);

  WiFi.mode(WIFI_STA); // the mode is different from board1
  WiFi.disconnect(); // disconnect before trying to connect again
  initWifi();

  pinMode(TouchPin, INPUT);

  // done to prevent the pin to be always HIGH
  digitalWrite(TouchPin, LOW);

  coap.server(callback_buzzer, "buzzer");
  coap.start();
}
```

Finally loop controls if bip is true, if it is the beep sound will be heard and only touching the touch sensor will stop it.

```
void loop()
{
  if(bip)
  {
    tone(BuzzPin, 100, 100); // Sound up when bip is true
    delay(100);

    // If the touch sensor is touched it means that buzzer
    // should not beep
    if(digitalRead(TouchPin) == HIGH)
      bip = false;
  }

  coap.loop();
}
```

3.3. Testing

I first tested every components independetly from one another. It was trivial and i will not show the codes for that because i just tested if every light or sensor was working and sometimes i didn’t even produced code. After i decided to implement wifi using my hotspot. Then i changed the access point to ESP32 to be sure the system was using only its resources to work.

4. Conclusion

This system is working as intended but some problems can be found.
The alarm will beep even if we stopped it one second before. This is because the ESP32 will continue to send request without knowing if we touched the sensor on the ESP8266. One solution can be using a counter that will consume the request without triggering the buzzer, unless it is 0. Another, more fancy, solution can be to use time and when n minutes pass the buzzer can be triggered again.

4.1. Future works

4.1.1. Moisture sensor

Adding a capacitive moisture sensor to monitor the terrain can be useful for plants who needs a lot of water. I will opt for capacitive because resistive ones will corrode in less time. Although there are solutions for this problem i think its faster to buy something already equipped to prevent corrosion.

4.1.2. UV leds

UV leds can benefit plants making them grow better, faster or bloom. Already assembled lamps can be found but they aren't cheap, i think that using UV led strips can lead to a similar but less expensive result.

4.1.3. Customization

Customize values for temperature and humidity can be helpful because every type of plant has its own ranges. For example, if i had a Hibiscus i probably couldn't use a temperature range from 8 to 30°C. To make this system more customable i thought of two possibilities: add a display and a selector to select wich type of plant its in the greenhouse or a more complex android application. In both cases the type of plants will define the range of values to be used.

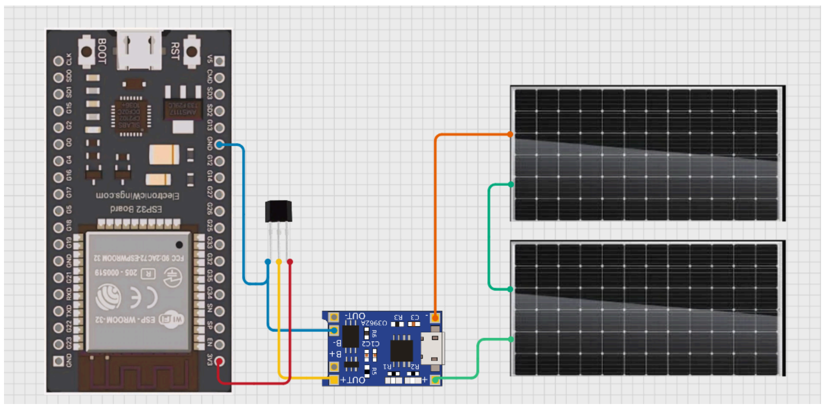
4.1.4. Solar panels

The board used for the greenhouse may be placed where there aren't electrical sockets. I am trying to build a solar panel system using:
- charger (TP4056)
- RGB led no. 7
- transformer (HT7133a)
- solar panel (3v) no. 2
- Li-On battery 18650

The procedure should be:
1. The two panels should be soldered in series to have an output voltage of ~4V.

2. TP4056 input should be soldered with the solar panels. The -/+ on the panels corresponds to the same polarity on the charger.
3. The HT7133a regulates the voltage and its GND should be soldered with TP4056 out- as well as its VIN with the TP4056 out+.
   The same soldering has to be done for the Li-On battery that will store power on the daytime but it is also used to power on the board.
4. Finally the voltage regulator should be soldered with the board (GND→GND and VOUT→3V3).

The final result without the battery is simplified in the picture below.



This part is still in progress because i am having some issues. I thought that a power bank could be used instead of the batteries but it seems that the usb-C is in input and not output. I then thought that buying an already assembled solar panel power bank can be a solution.