

# Project - Web server publishing ESP32-CAM pictures and sensor data readings from a MQTT server:

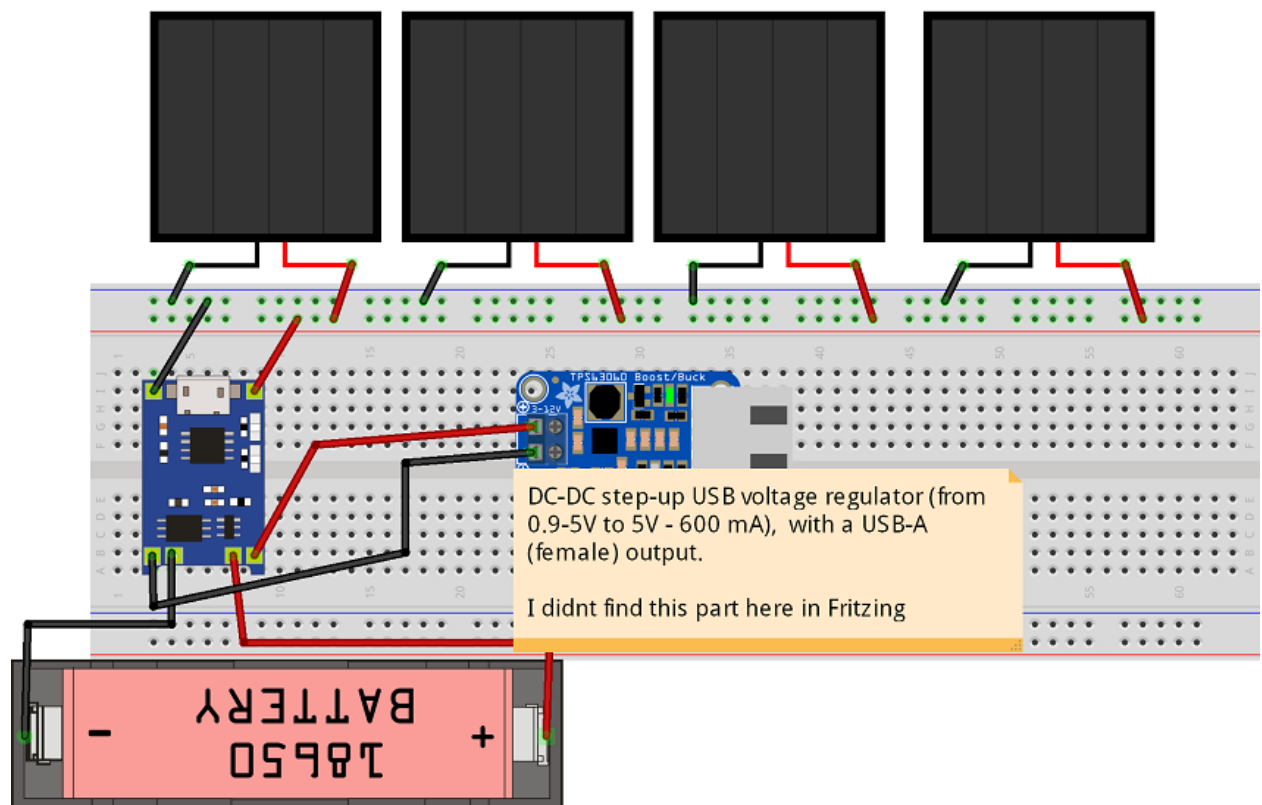
## 1 - Project Modules and Required Parts:

### 1.1 - The three modules of the project:

- 1 - Module using the 'NodeMCU V1.0' board with SoC ESP12-E (with ESP8266 microcontroller).
  - Reads the temperature (in °C) and air pressure (in hPa) from the BMP280 module and the ADC value from the Light Dependent Resistor (LDR).
  - The data read from the sensors are sent to the MQTT server.
  - The NodeMCU is power-supplied by a 18650 battery.
  - The 18650 battery is charged by a TP4056 module and four solar panels.
- 2 - ESP32-CAM Module:
  - Take a picture every five minutes. It just stores the last picture taken, in the SPIFFS memory of the ESP32-CAM.
  - This module hosts the web server HTML page which shows the last picture taken and the sensor data.
  - The HTML code of the web page is generated in a function of the sketch of this module.
  - The template of the web page is hosted in the header file 'PAG\_WEB.h'
- 3 - PC executing **Mosquitto MQTT** and Ngrok.
  - Here, I used a Raspberry Pi with the necessary software installed.
  - But its not a requisite to use a Raspberry Pi here. Any computer with windows will do the same thing. It just needs the required software.
  - **Mosquitto MQTT**: Its the MQTT server program. It receives the published data from the NodeMCU ESP8266 and sends the published data to the ESP32-CAM (the subscriber).
  - **Ngrok**: Its the internet server which redirects the ESP32-CAM web server to the internet.
    - \* It requires an account in the Ngrok site.
    - \* It offers some free services, like the one shown here. But the site has some paid services too.
    - \* I'm using here only its free services.

### 1.2 - Parts used in each module of the project:

- 1 - **NodeMCU (ESP12-E) with the sensors**:
  - 1x NodeMCU 1.0 module (SoC ESP12-E, with the ESP8266 microcontroller).
  - 1x BMP280 sensor module (temperature and air pressure)
  - 1x light dependent resistor (LDR).
  - 1x 4.7K resistor.



fritzing

Figure 1: Img\_01

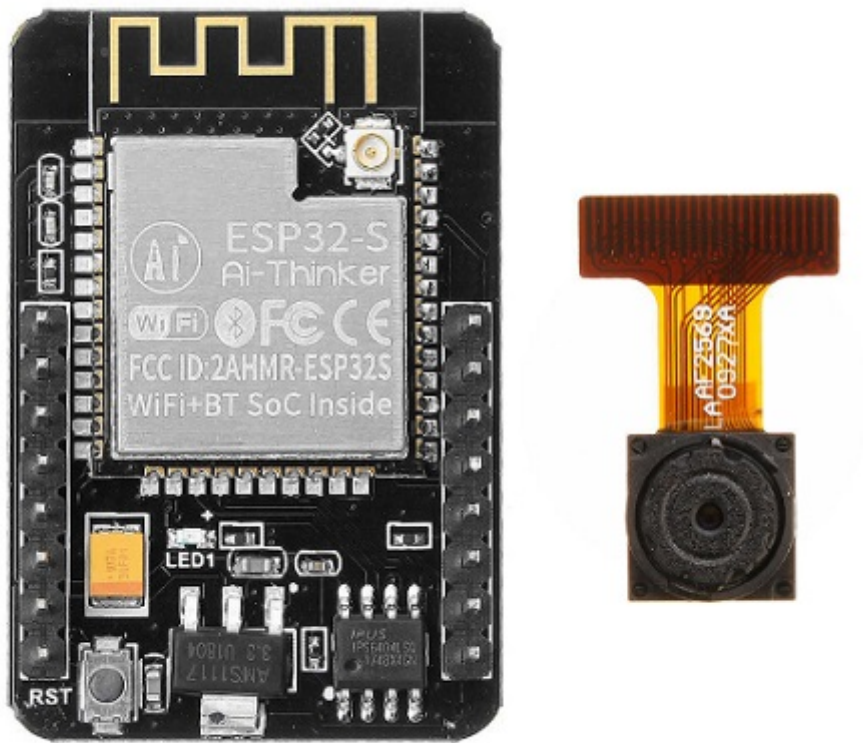


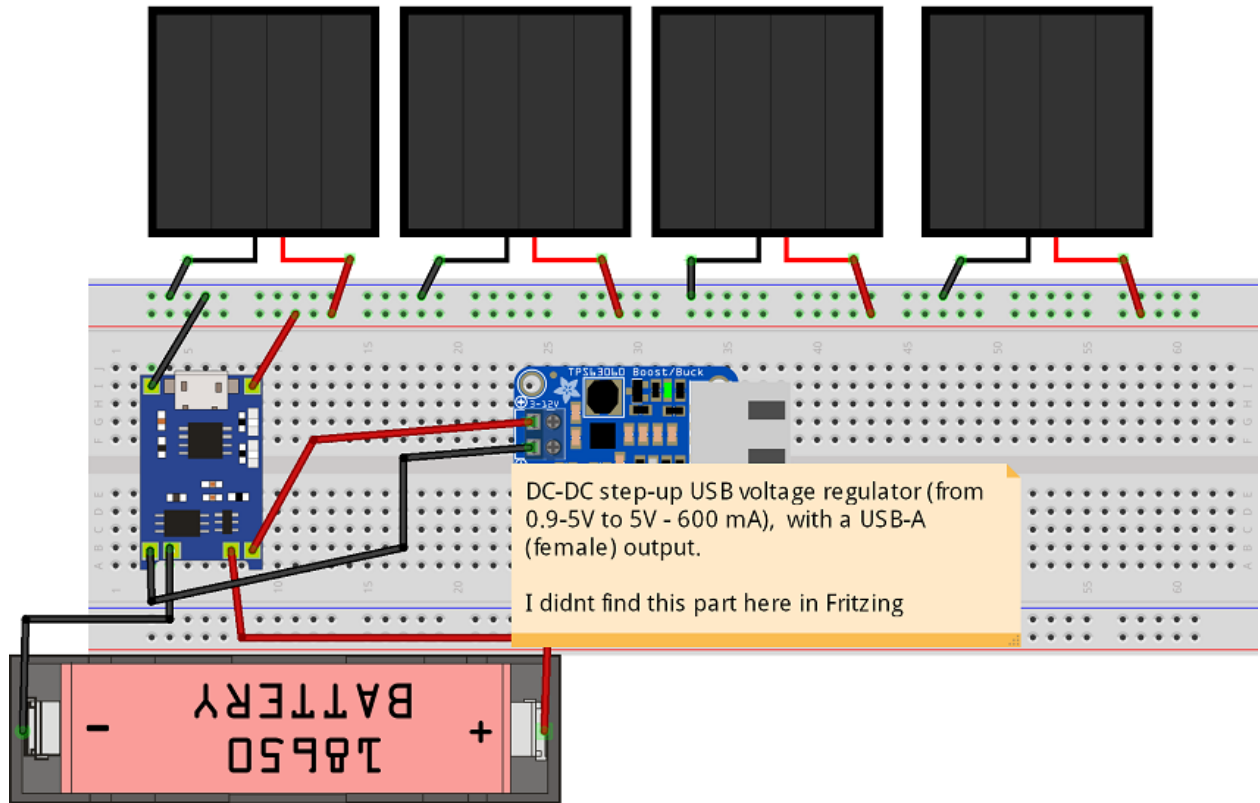
Figure 2: Img\_02

- 1x DC-DC step-up USB voltage regulator (from 0.9-5V to 5V - 600 mA - with a USB-A output).
- 1x 18650 battery (4.2V, 9800mAh).
- 1x Li-ion battery charger module **TP4056, WITH PROTECTION.**
- 4x 1W 6V solar panels (the specifications told that they were 6V panels, but my measurements showed 5V-2.15V).
- 2x USB cables, with a USB-A and a USB Micro-B terminals. **A charging-only cable does this service very well.**
- **2 - Módulo ESP32-CAM with the web server:**
  - 1x ESP32-CAM module
  - 1x **FT232R FTDI** programmer (the ESP32-CAM doesn't contain a CP2101 chip to transfer the sketches to it).
  - 1x 5V DC power supply.

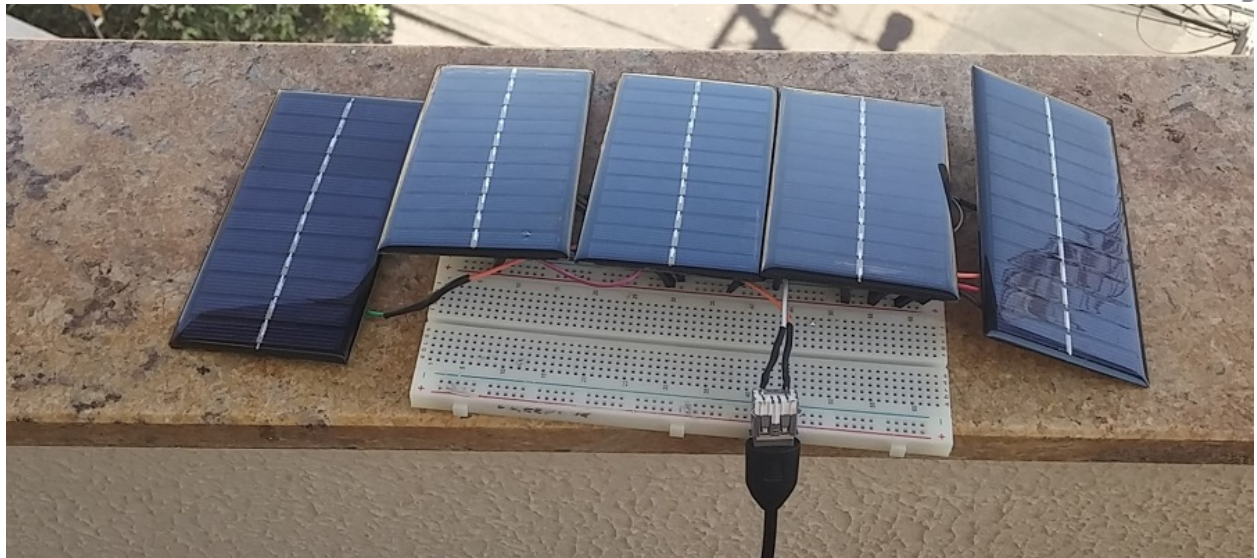
## 2 - Diagrams and photos:

### 2.1 - Part 1 - ESP8266 and sensors

#### 2.1.1 - Power Supply (solar panels and battery charger)

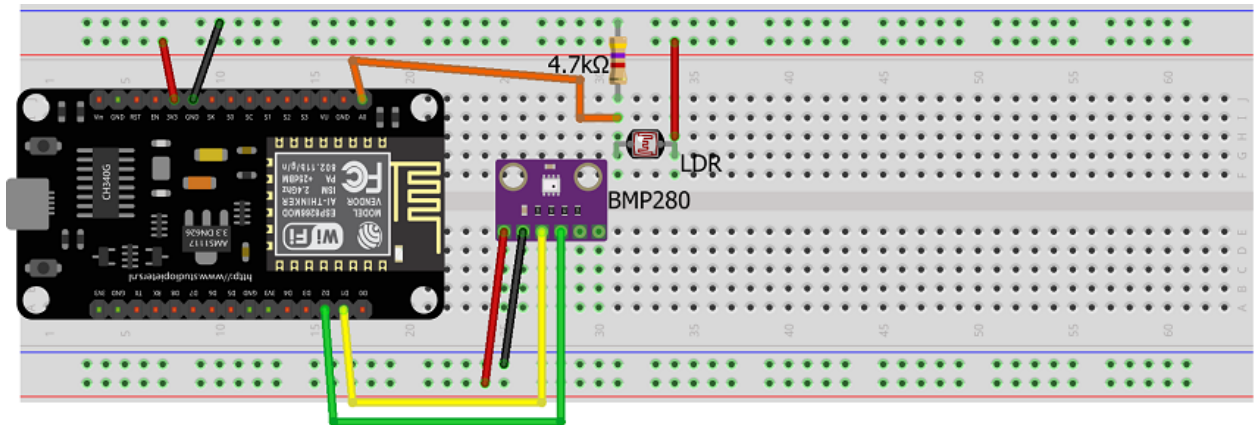


fritzing

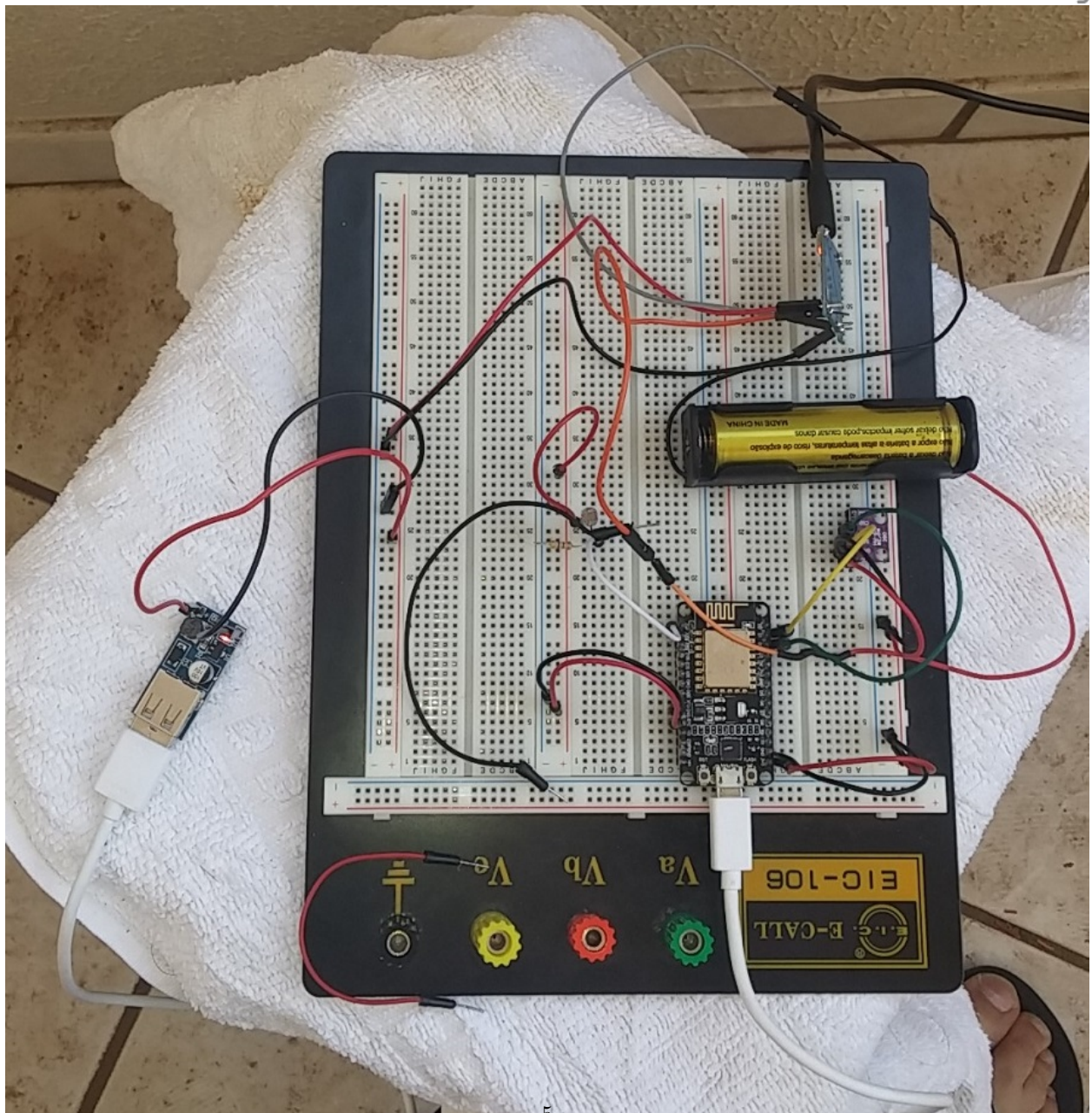




## 2.1.2 - NodeMCU and Sensores



fritzing



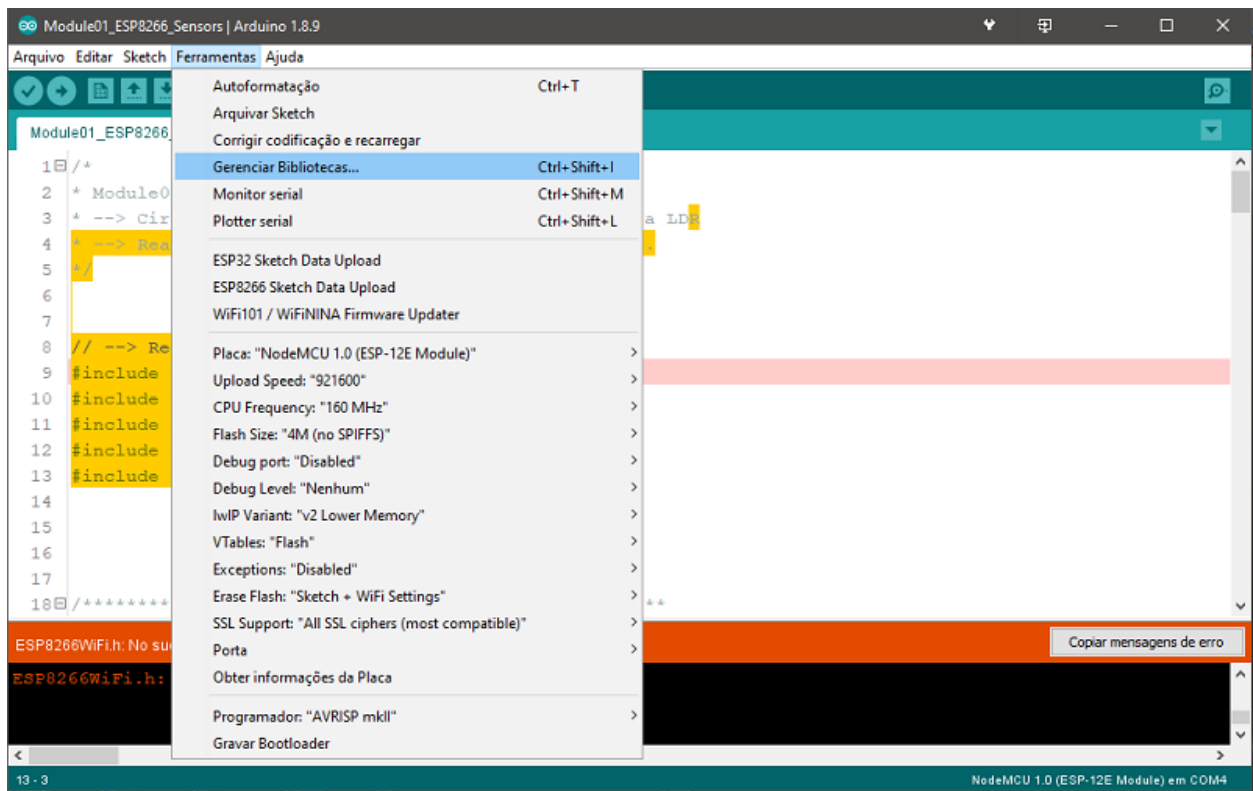


Figure 3: Selecting the NodeMCU board in Arduino IDE

### 2.1.3 - Selecting the NodeMCU board in Arduino IDE

### 2.1.4 - MQTT Server:

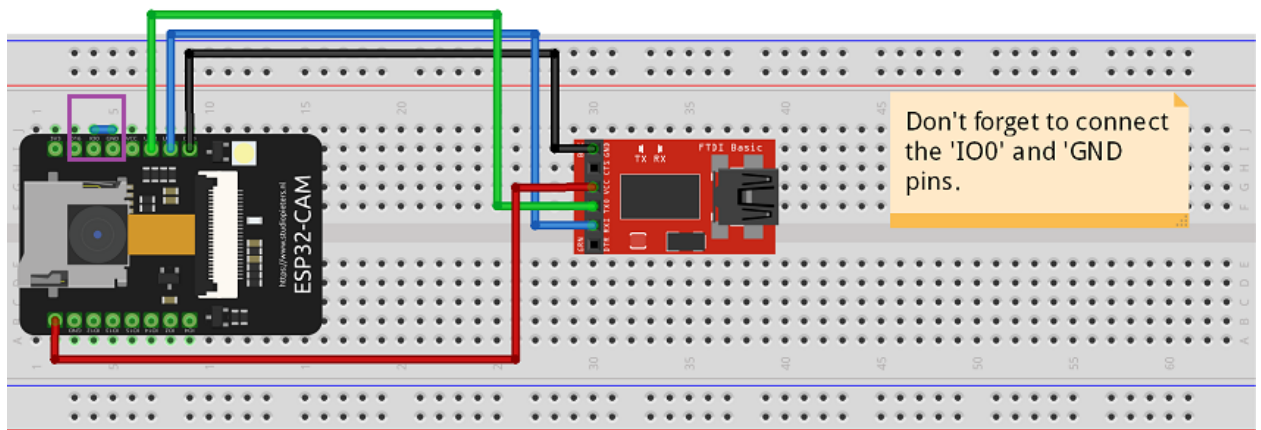
- The sensor readings are published in the MQTT Server running on Raspberry Pi.
- The MQTT server program used here is the **Moquito MQTT Broker**.
- The picture above shows the result of a subscription to the topic **sensors/temp**.
- Alternatively, you can subscribe for a MQTT with the **MyMQTT** App (Android).
- The image above shows the result on the MyMQTT App, after connecting to my MQTT server in Raspberry Pi and subscribint to the following topics:
  - **sensors/temp**.
  - **sensors/pressure**.
  - **sensors/adc\_ldr**.

```
pi@raspberrypi: ~  
Arquivo  Editar  Abas  Ajuda  
pi@raspberrypi:~ $ mosquitto_sub -h 192.168.0.7 -p 1883 -t sensors/temp  
30.59  
30.60  
30.60  
30.62  
30.62  
30.63
```

Figure 4: Subscribing to a Topic in Mosquitto

## 2.2 - Part 2 - ESP32-CAM

### 2.2.1 - Programming the ESP32-CAM with the FT232R FTDI



fritzing



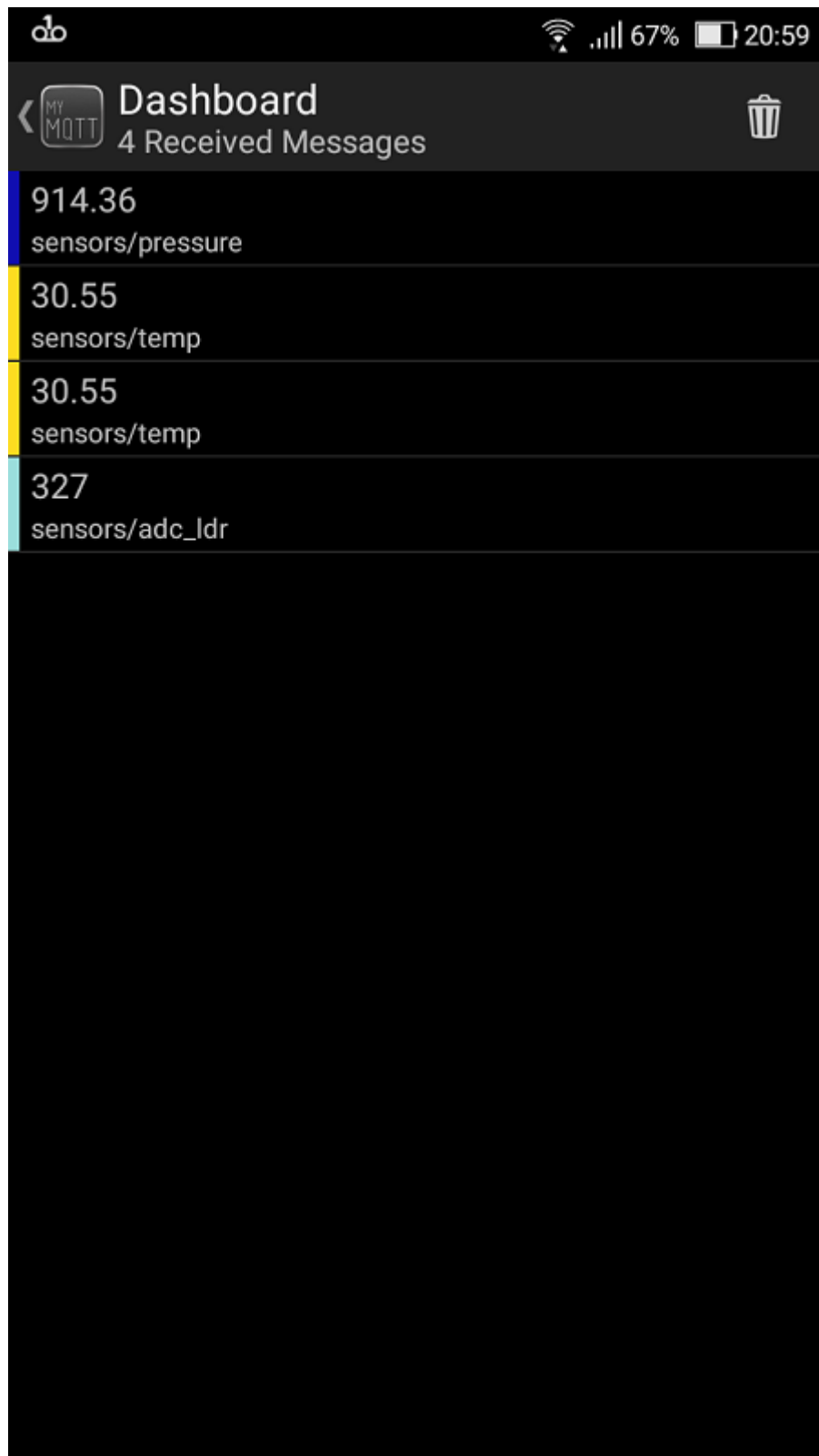


Figure 5: MyMQTT (Android) Screenshot



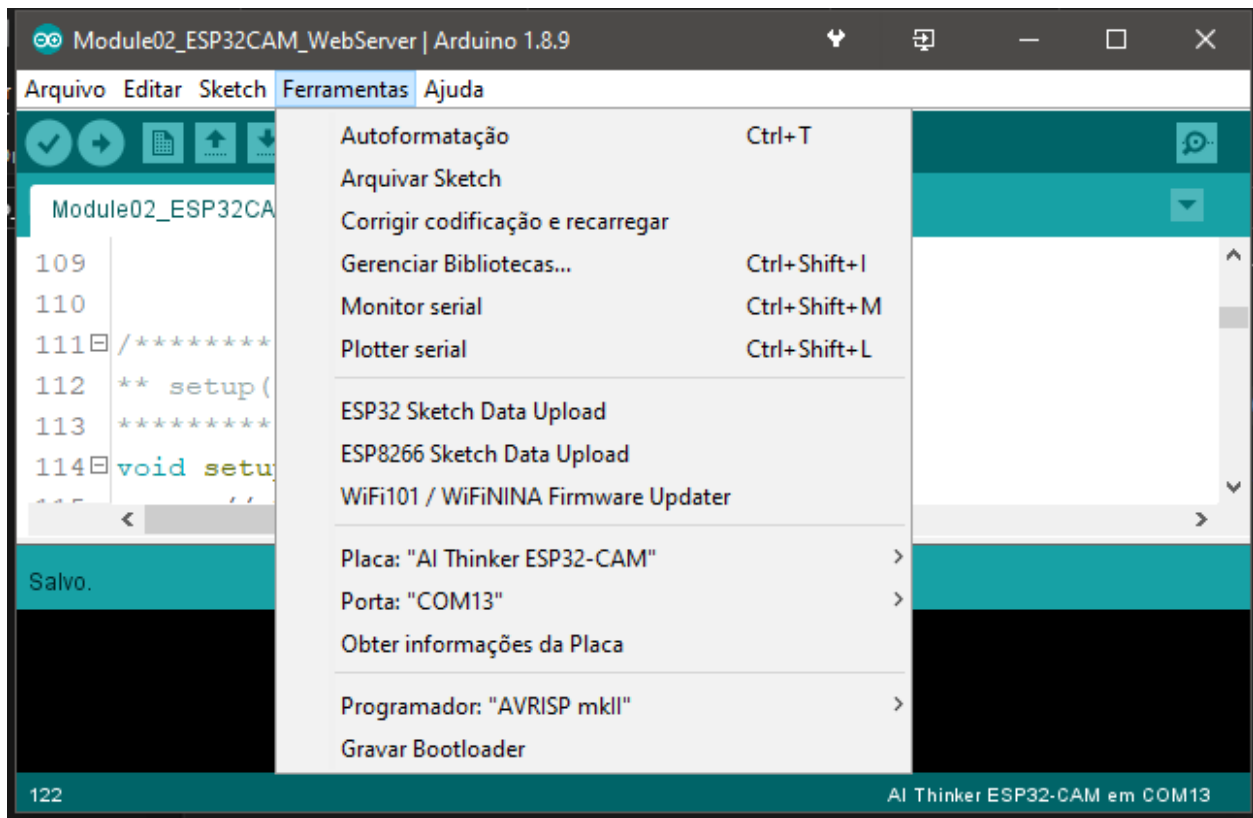
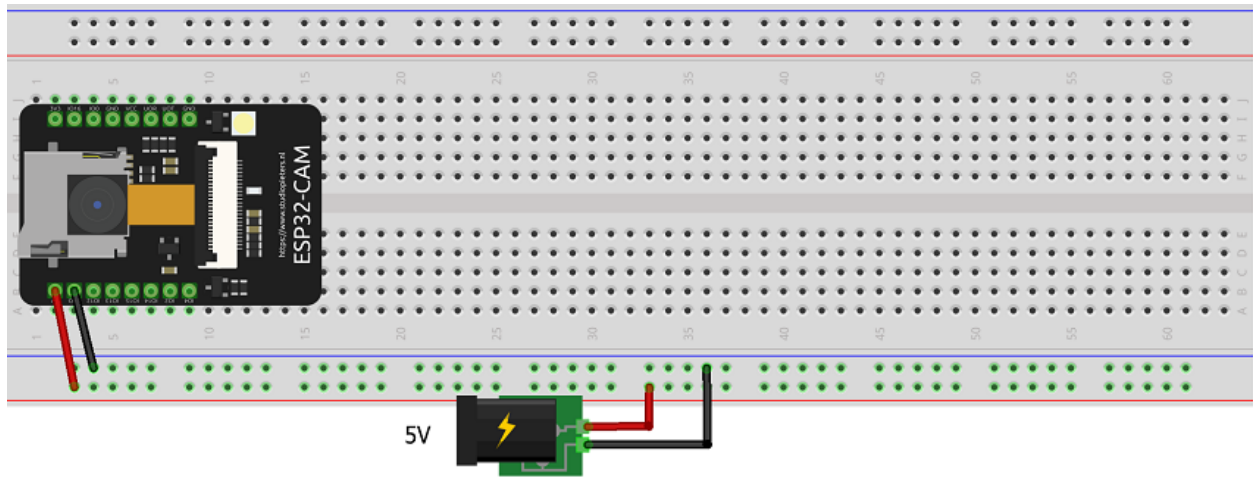


Figure 6: Img\_09

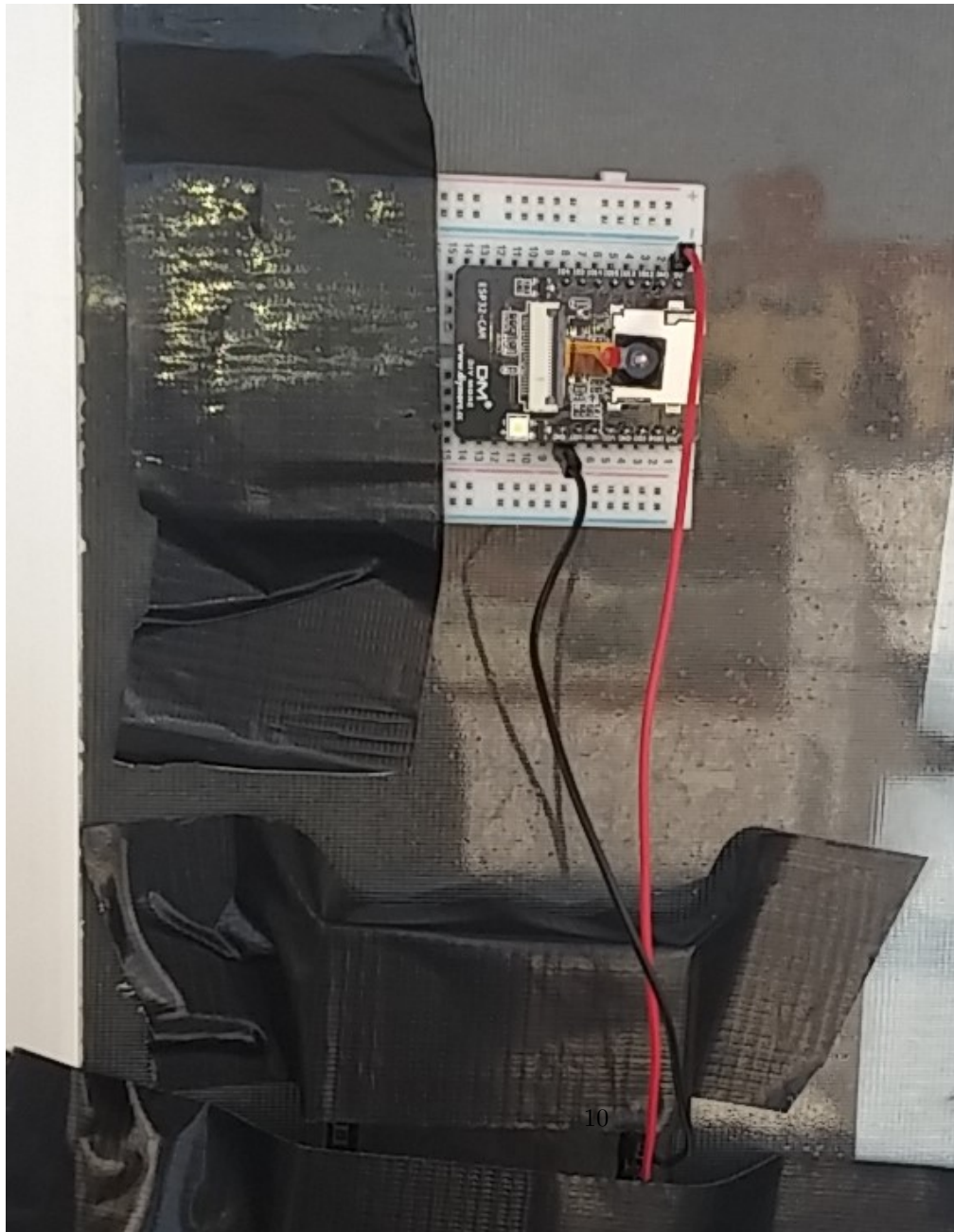
- The image above shows how to connect the ESP32-CAM to the FT232R FTDI programmer.
- The **I00** pin of the ESP32-CAM must be connected to the **GND** pin of the ESP32-CAM.

2.2.2 - Selecting the ESP32-CAM in Arduino IDE:

2.2.3 - Running the sketch



fritzing



- I Used here an broken Arduino UNO, whose power-supply pins were the only pins working.
- But any 5V DC power-supply will do the same.
- Don't forget to remove the wire connecting the pins IO0 and GND.

#### 2.2.4 - The ESP32-CAM Web server:

COM13

Enviar

```

Initializing...
Initializing the SPIFFS file system...Ok!
Inicializing the OV2640 camera module...Ok!
Connecting to ALGAR 207
.....
Conected!
IP Address: 192.168.0.9
MAC Address: 80:7D:3A:B7:01:E8

Initializing the NTP (Network Time Protocol) client...Ok!

Connecting to the MQTT server...
Connected to the MQTT server in 192.168.0.7:1883
Ok!

Taking a picture...
Picture file name: /Photo.jpg
The picture has been saved in /Photo.jpg - Size: 0 bytes
Taking a picture...
Picture file name: /Photo.jpg
The picture has been saved in /Photo.jpg - Size: 73472 bytes

```

☒ Auto-rolagem
☐ Show timestamp

Ambos, NL e CR

115200 velocidade

Deleta a saida

Web Server ESP32-CAM

Não seguro | 192.168.0.9:8088

Apps

Dissertação

Importado do Firefox

Cursos\_RNT

RaspberryPi&Ardui...

SistemasEmbarcados

PIC

widgetsnextensio...

Blog Porta 23

Python

Outros favoritos

### Web Server Using the ESP32-CAM:

Date of the last photo: 2019-09-01T23:32:39Z


Time of the last photo:: 23:32:39

Temperature: 30.13°C

Pressure: 917.60hPa

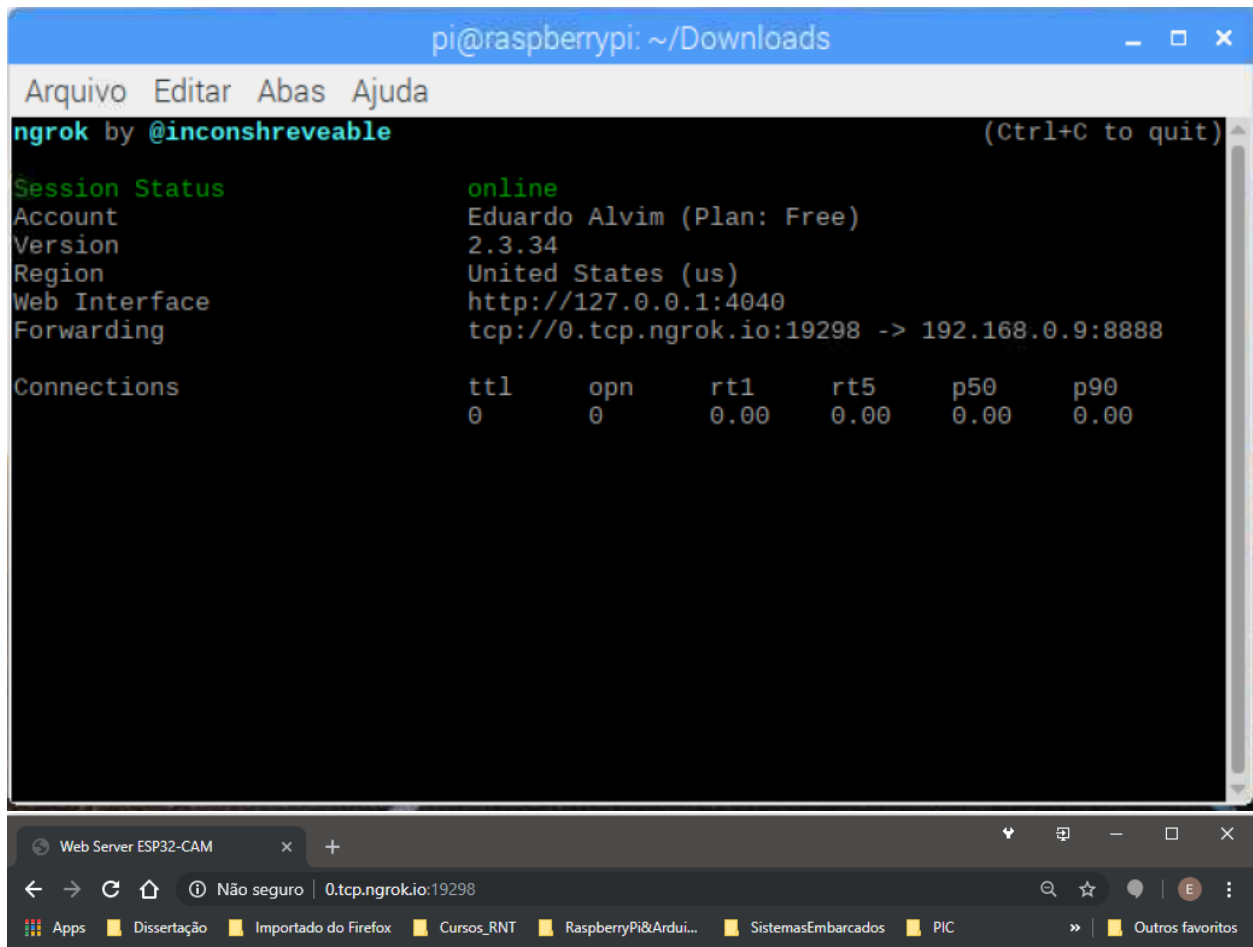
Level of Illumination from the LDR ([0, 1023]): 220

Picture saved in the SPIFFS:



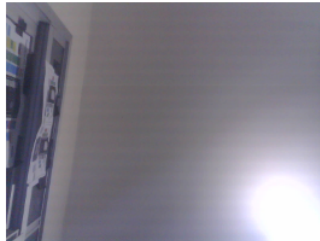
- The first image above shows the output in Monitor Serial of the Arduino IDE.
- The ESP32-CAM IP address in my Wi-Fi network is **192.168.0.9**.
- To access the web server (second image above), just type in your internet browser the IP address of the ESP32-CAM and the port number (in my case, **192.168.0.9:8888**).

### 2.2.5 - The Ngrok redirection service:



#### Web Server Using the ESP32-CAM:

Date of the last photo: 2019-09-01T23:37:22Z  
Time of the last photo:: 23:37:22  
Temperature: 30.15°C  
Pressure: 917.46hPa  
Level of Illumination from the LDR ([0, 1023]): 224  
Picture saved in the SPIFFS:



- After install and configure the Ngrok program in your Raspberry Pi or PC, and with the web server of the ESP32-CAM running, you can execute the Ngrok to make the web server available in the internet.
- In Raspberry Pi:
  - Go to the folder where the Ngrok program is installed: `cd /home/pi/Downloads` (Or the folder



which you installed it).

- Execute Ngrok:
  - \* **Raspberry Pi:** `./ngrok tcp 192.168.0.9:8888`
  - \* **Windows Power Shell:** `ngrok tcp 192.168.0.9:8888`
- The output shown in the first image above shows that my web server is available in the address `http://0.tcp.ngrok.io:19298/`
- To access your web server outside your Wi-Fi network, just type this address (`http://0.tcp.ngrok.io:19298/`) in any internet browser.