Mestrado Informática

Pós-Graduação - Data Science and Digital Transformation

Internet das Coisas e Sistemas Embebidos

Projeto Final – Sensor de Temperatura para IoT

João Alex Arruda da Silva

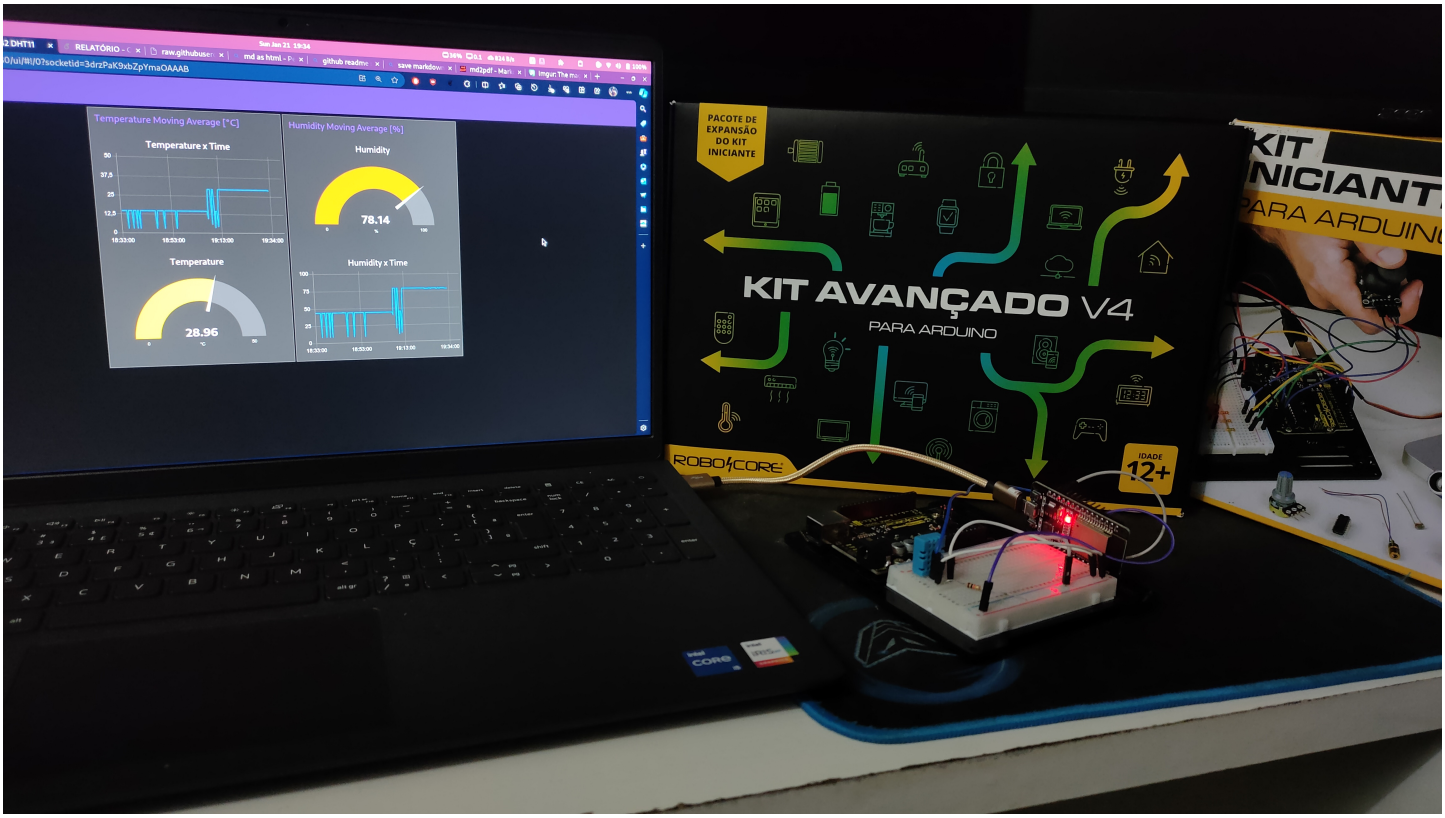
Temperature and Humidity Sensor Project

This project uses an ESP32 and DHT11 sensor to measure temperature and humidity. The data is sent to a Mosquitto broker and visualized on a Node-RED dashboard. It's an improvement over this simpler project, which only works on a local network with no dashboard nor broker. Have a look at the old one to see the differences.

It's an assignment for the course "Internet of Things" at the Instituto Politécnico de Portalegre, in Portugal.

You can find out more about the course here.

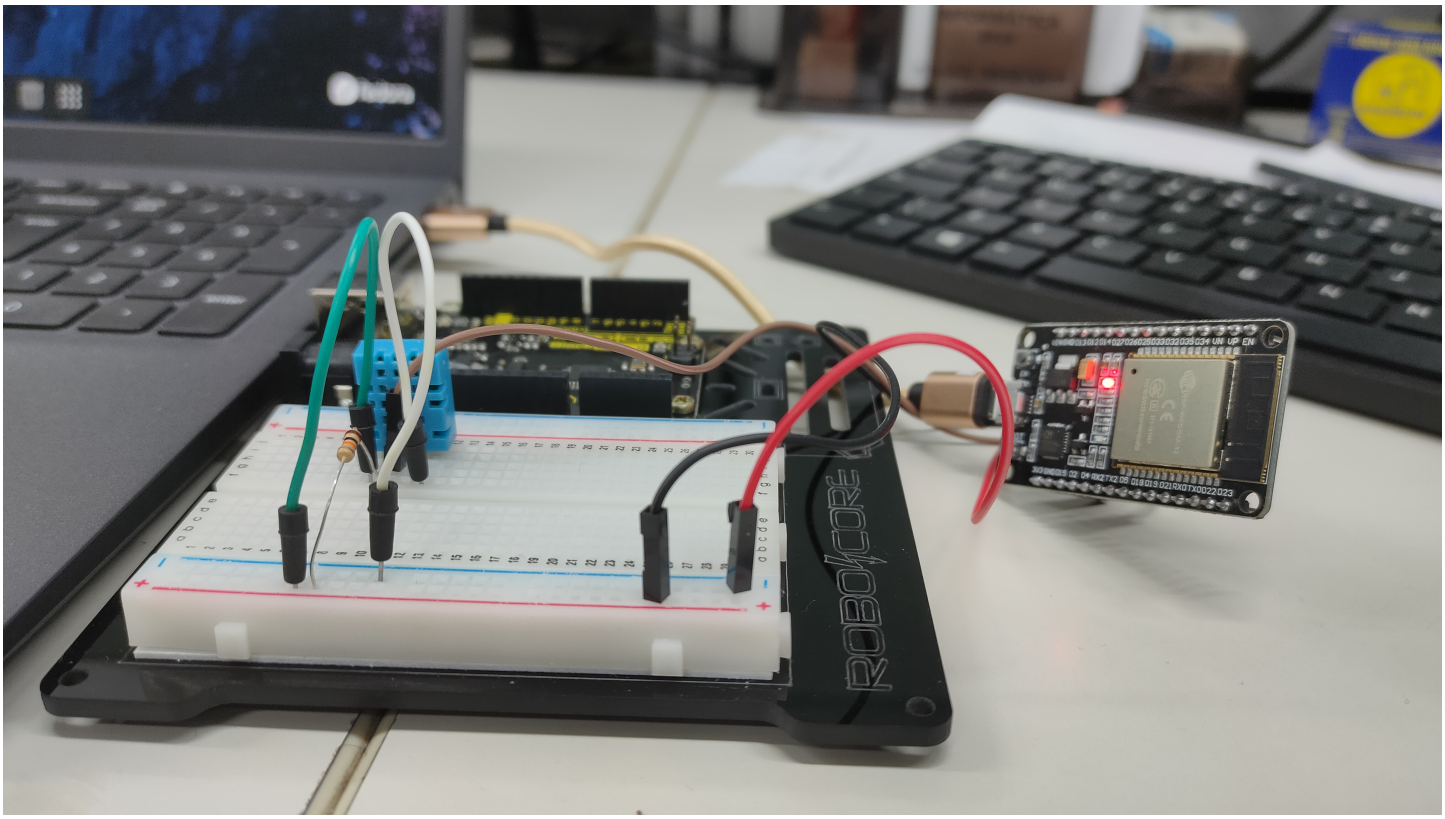
Screenshot



You can find more screenshots here.

Hardware Requirements

- ESP32
- DHT11 Temperature and Humidity Sensor
- 10k Ohm resistor
- Breadboard
- Jumper wires



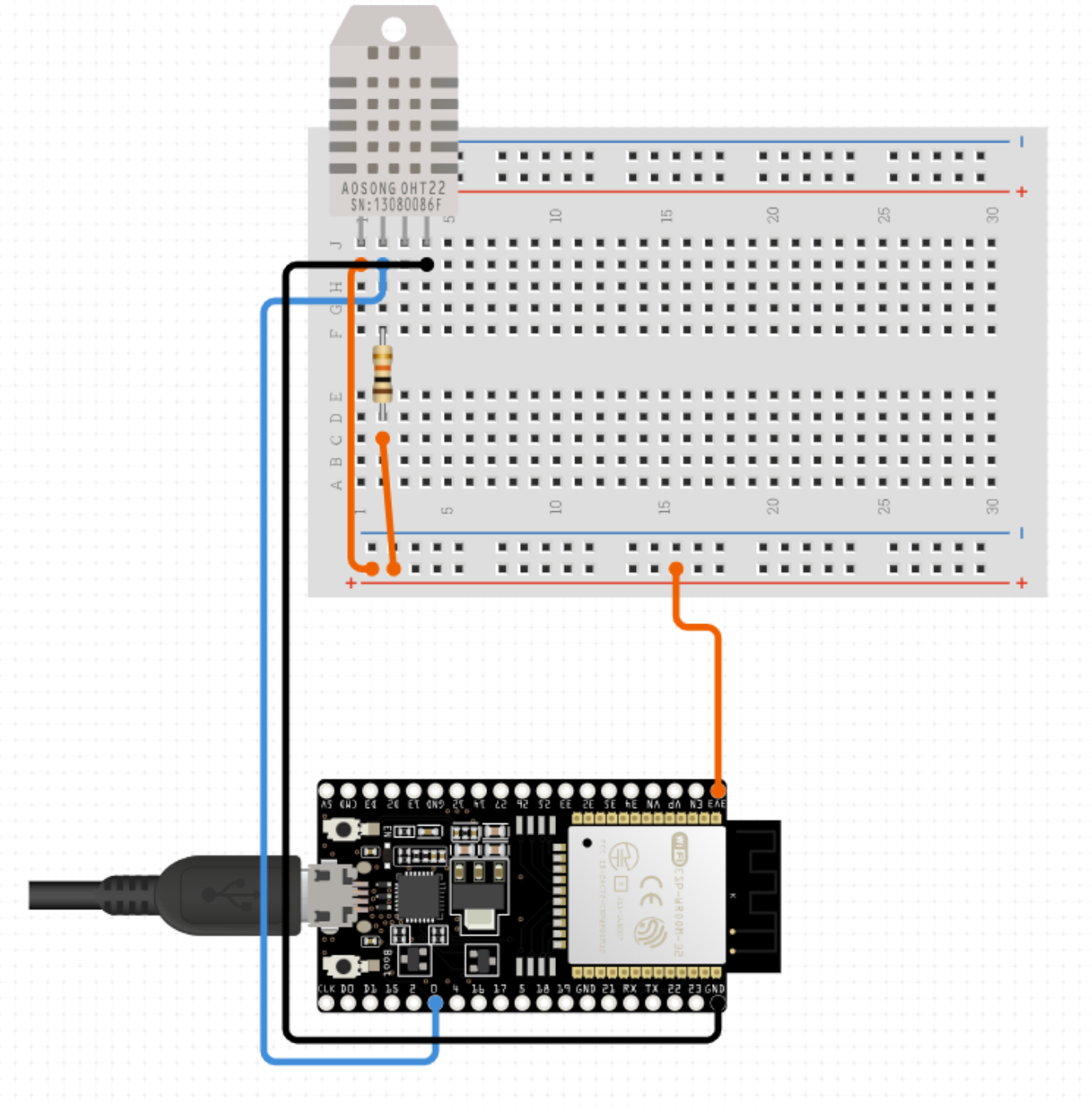
Software Requirements

- PlatformIO (preferred) or Arduino IDE
- Mosquitto MQTT Broker
- Node-RED

Setup and Installation

1. **Hardware Setup**: Connect the DHT11 sensor to your ESP32. The connections are as follows:
    - DHT11 VCC to ESP32 3V3
    - DHT11 GND to ESP32 GND
    - DHT11 DATA to ESP32 GPIO 4
    - DHT11 DATA to 10k Ohm resistor
    - 10k Ohm resistor to ESP32 3V3

It should look something like this:



2. **Software Setup**:
    - Install and setup VSCode + PlatformIO (preferred) or Arduino IDE.
    - Install Mosquitto MQTT Broker.
    - Install Node-RED.
    - Install Node-RED Dashboard.
3. **Code Deployment**: Upload the provided code to your ESP32.

Running the Project

1. Start the Mosquitto broker.

2. Start Node-RED and open the provided dashboard.
3. Power on the ESP32.

Code Explanation

The code is all contained in the `src/main.cpp` file. It's divided into 3 main sections:

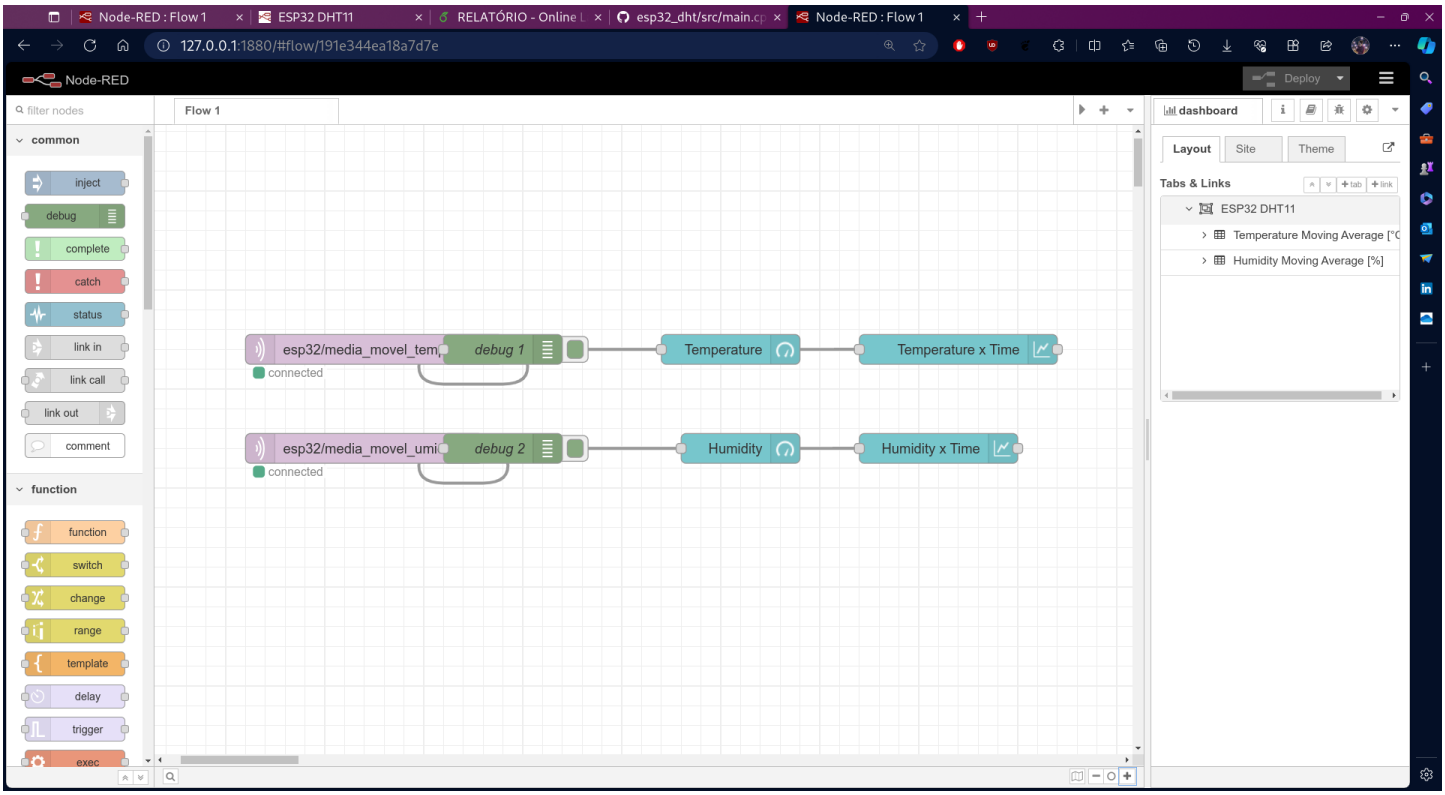
1. **Libraries**: The libraries used in this project are:
    - `WiFi.h` : Used to connect to a WiFi network.
    - `PubSubClient.h` : Used to connect to a MQTT broker and publish messages.
    - `DHT.h` : Used to read data from the DHT11 sensor.
    - `Adafruit_Sensor.h` : Used along with the DHT library.
2. **Constants**: The constants used in this project are:
    - `ssid` : The SSID of the WiFi network to connect to.
    - `password` : The password of the WiFi network to connect to.
    - `mqtt_server` : The IP address of the MQTT broker.
    - `mqtt_port` : The port of the MQTT broker.
    - `DHT_PIN` : The GPIO pin of the ESP32 that the DHT11 sensor is connected to.
    - `DHT_TYPE` : The type of the DHT sensor (DHT11, DHT21 or DHT22).
    - `numReadings` : The number of readings on the circular buffer before calculating the average.
3. **Functions**: The functions used in this project are:
    - `setup_wifi()` : The setup_wifi function is used to connect to the WiFi network.
    - `reconnect()` : The reconnect function is used to connect to the MQTT broker.
    - `updateReadings()` : The updateReadings function is used to update the circular buffer with new readings from the DHT sensor.
    - `calculateMovingAverage()` : The calculateMovingAverage function is used to calculate the moving average of the circular buffer.
    - `setup()` : The setup function is called once when the ESP32 boots up. It's used to initialize the serial port, connect to the WiFi network and the MQTT broker, and initialize the DHT sensor.
    - `loop()` : The loop function is called repeatedly after the setup function. It's used to read the temperature and humidity from the DHT sensor and publish them to the MQTT broker. It also calls the updateReadings and calculateMovingAverage functions to publish the moving average of the temperature and humidity. It also handles the reconnection to the MQTT broker. It has a nice formatted output to the serial port for debugging purposes.

MQTT Broker

Download and install the Mosquitto MQTT Broker from here. You can start the broker with the default configuration by running `mosquitto` in the terminal.

Node-RED

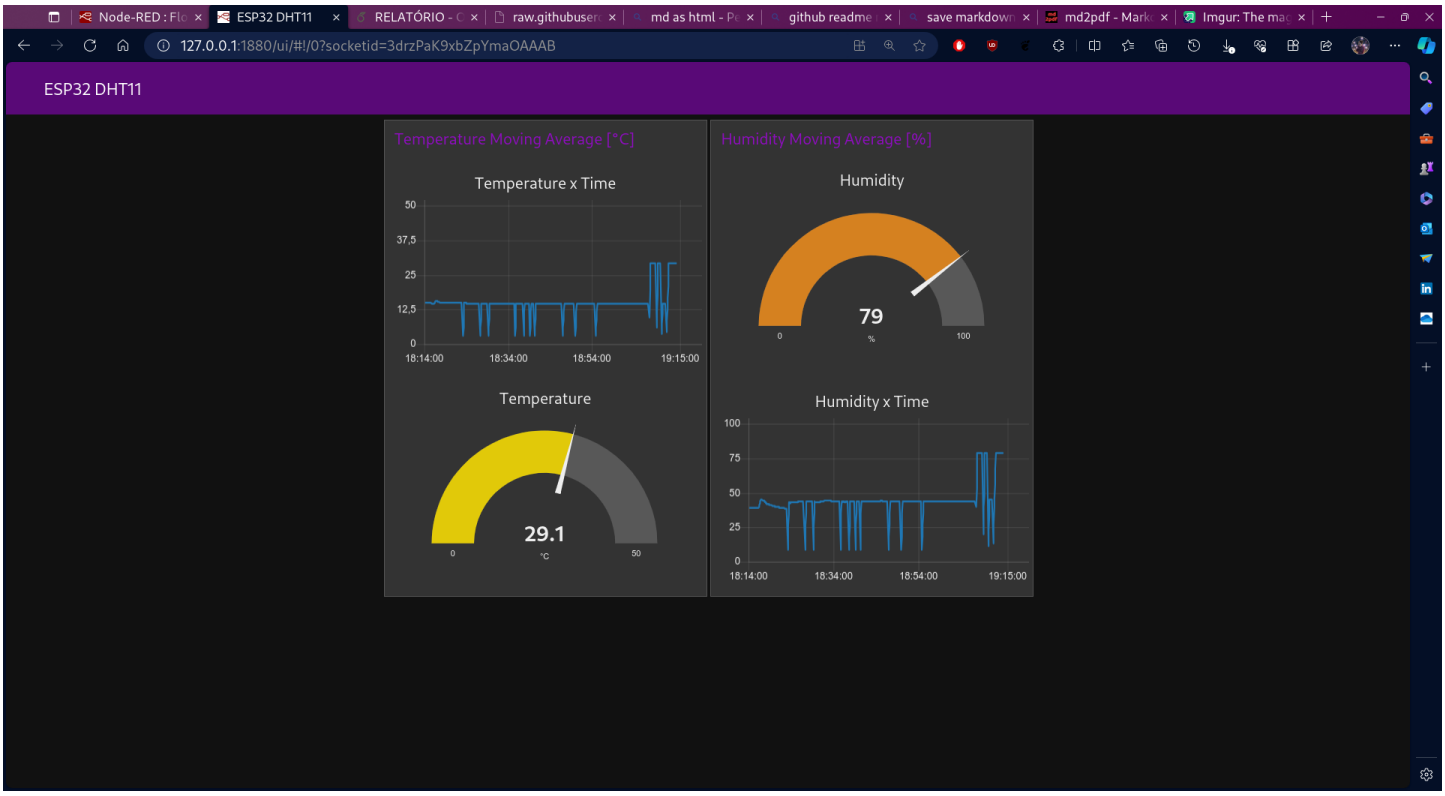
Download and install Node-RED from here. You can start Node-RED by running `node-red` in the terminal.



It's very easy to create a flow in Node-RED. You can find more information about it here.

Node-RED Dashboard

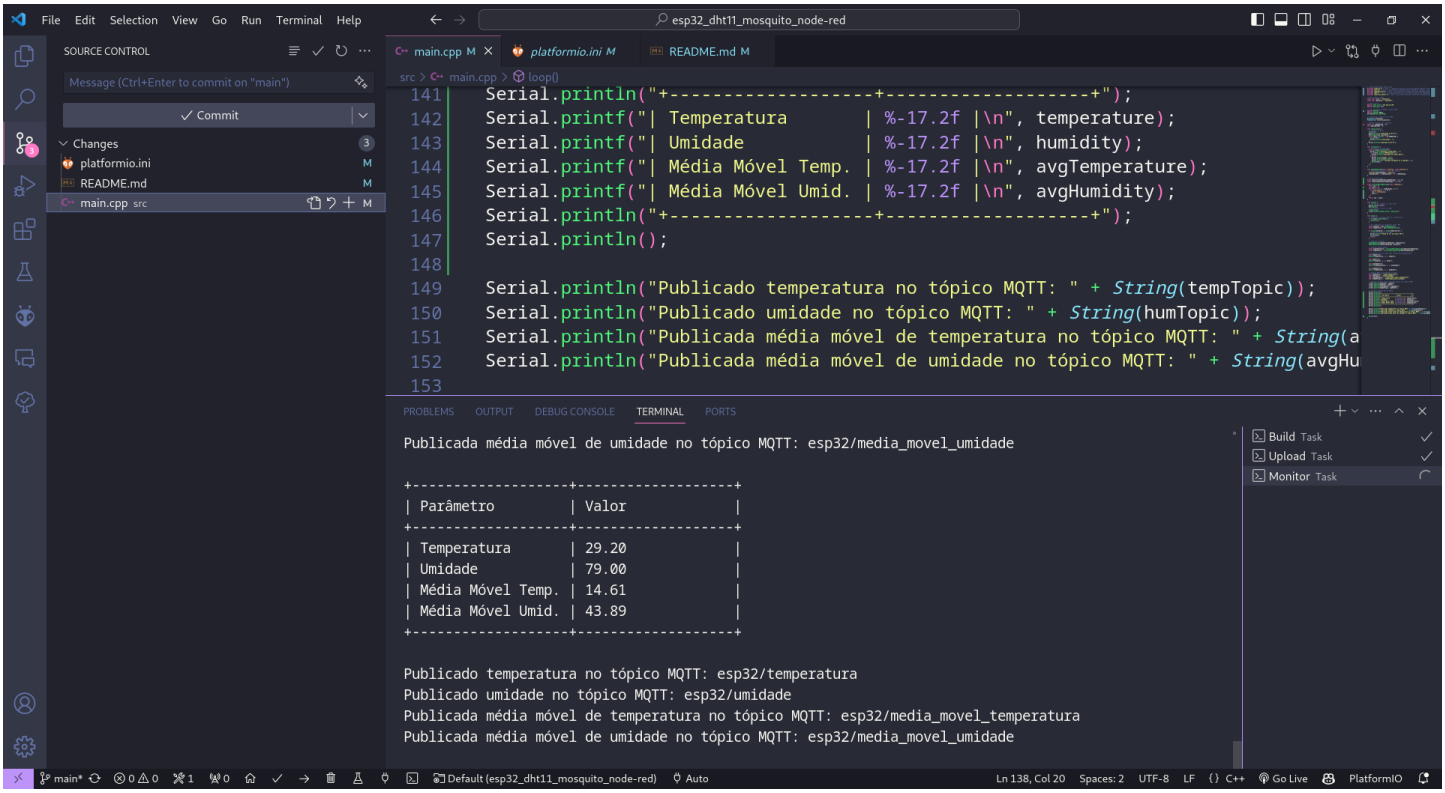
Install the Node-RED Dashboard from here. You can access the dashboard by going to `http://localhost:1880/ui` in your browser.



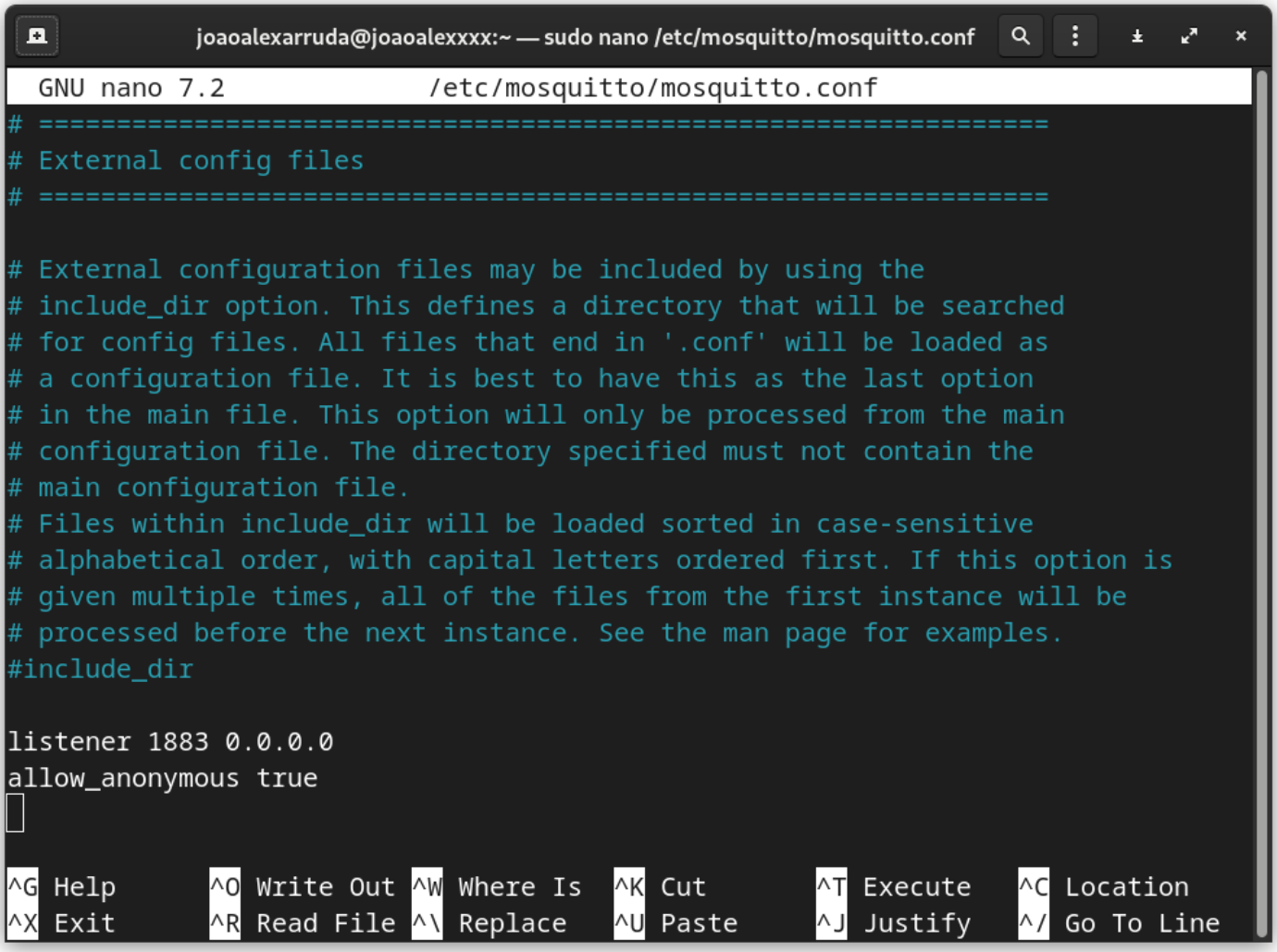
Troubleshooting

Be sure to follow the hardware setup correctly. If you're using a different ESP32 board, the GPIO pins may be different.

Pay attention to the serial output of the ESP32. It's very useful for debugging.



In the mosquitto.conf file, be sure to set the `allow_anonymous` option to `true`. Also set listener to `1883 0.0.0.0` as seen below.



Considerations

- The code has comments explaining what each line does.
- The DHT11 sensor is a very cheap sensor. It's not very accurate and it's not very reliable. It's a good sensor to start with, but it's not recommended for real-world applications.
- It's very easy to install everything if you are running Linux. If you are running Windows, you may have some problems. I recommend using Fedora.
- VSCode + PlatformIO is preferred over the Arduino IDE. It's much easier to use and it has a lot of features, such as code completion, code linting, git integration, etc.
- The ESP32 is a very powerful microcontroller. It's very easy to use and it has a lot of features. It's a good choice for this project.
- Note that the arduino uno showed in the picture is not used in this project. The only microcontroller on is the ESP32.

Future Improvements

- Connect another ESP32 with a DHT11 sensor to the same MQTT broker and display the data on the same dashboard.

- Subscribe multiple devices to the same MQTT broker so they can do actions based on the data received.
- Get a Raspberry Pi and install the Mosquitto MQTT Broker and Node-RED on it. This way you can have a dedicated device for this project.
- Use a good sensor instead of a cheap DHT11 sensor.
- Implement a better way to handle errors.
- Utilize these sensors in a real-world scenario, like a smart home. For example, turn on the air conditioner when the temperature is too high, or turn on the humidifier when the humidity is too low.