2MOBR - PERVASIVE COMPUTING & INTERNET OF THINGS (IOT) APPLICATIONS

**Trabalho final da disciplina**

**Integrantes**

Gustavo Matias Cotta – 350576

Nayara Pereira Felix – 348253

Vinicius Soares Lima – 348157

Yuri Silva da Cunha – 349192

**Solução**

Simulação de uma estação meteorológica

**Vídeo demonstrativo**

https://youtu.be/8p-1t0GFwSo

**Código-fonte no GitHub**

[gmcotta/2mobr-iot-trabalho-final (github.com)](https://github.com/gmcotta/2mobr-iot-trabalho-final)

**Resumo da solução**

Uma estação meteorológica pode ajudar no fornecimento de dados para tomada de decisão informada, monitoramento das mudanças climáticas, promoção da agricultura sustentável, prevenção de desastres naturais e apoio à saúde pública.

A seguinte solução possui Edge, Fog e Cloud Computing.

O Edge Computing é composto por dois sensores, o BME280 e o DHT22, para obter valores de temperatura, umidade e pressão do ar. Pode-se adicionar também outros sensores, como sensor de luminosidade, de chuva, de velocidade e direção do vento, sensor UV, para deixar a estação mais completa. Os sensores estão ligados a um ESP8266 (NodeMCU), que obtém os valores a cada 10 segundos e envia para um Raspberry Pi via MQTT. Tanto o ESP8266 e o Raspberry Pi estão conectados na internet.

O Fog Computing é composto pelo Raspberry Pi, que recebe os valores do ESP8266 e faz três atividades que exigem um poder de processamento maior que o ESP8266:

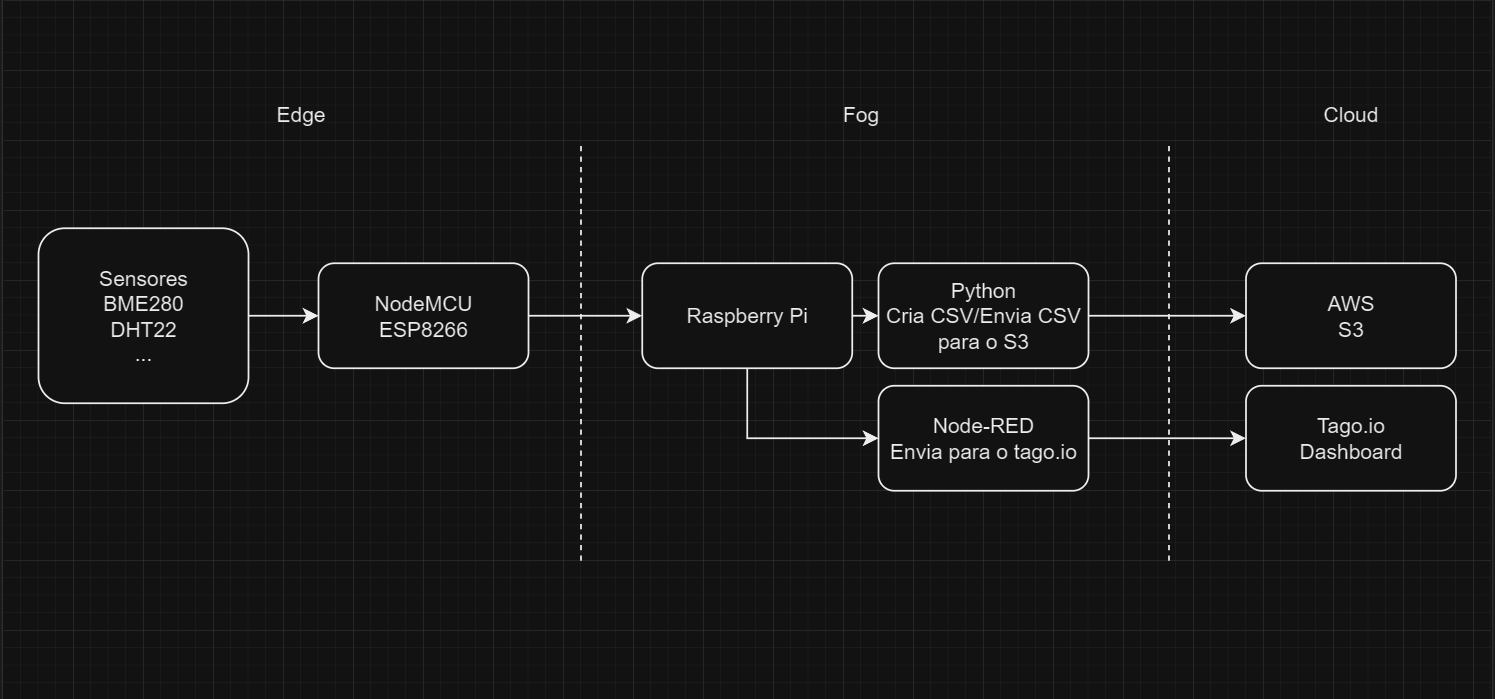
1) armazena os valores dos sensores em um csv usando Python;

2) envia os csvs gerados para um bucket do S3 da AWS usando Python;

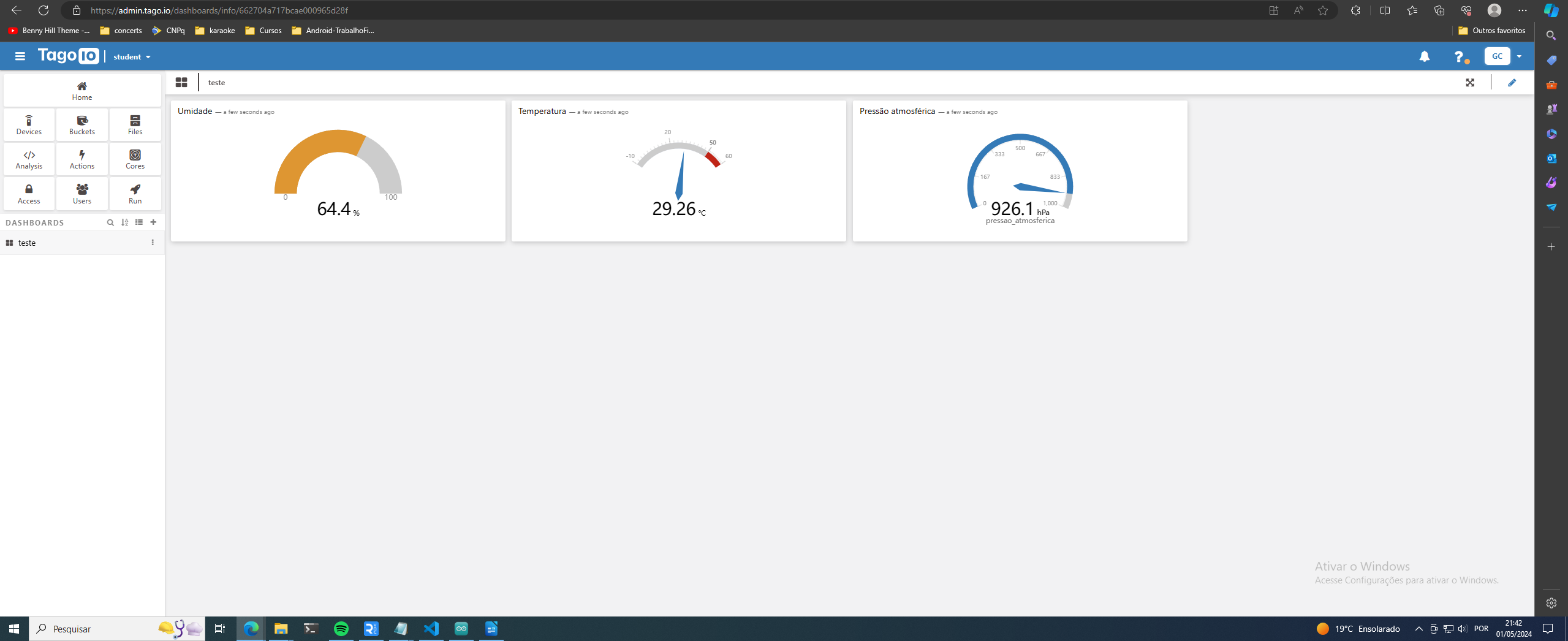
3) envia os valores dos sensores para o tago.io usando o Node-Red.

O Cloud Computing é composto basicamente pelo tago.io, que possui um dashboard onde exibe os valores recebidos no Raspberry Pi; e o S3 da AWS, que guarda os csvs gerados pelo Raspberry Pi, onde um time de ciência de dados pode baixar esses arquivos e gerar insights sobre os valores obtidos.

**Diagrama da solução**



**Dashboard no tago.io**



**Componentes utilizados**

- NodeMCU

- BME280

- DHT22

- Resistor 4k7 Ohms

- Raspberry Pi

- Protoboard

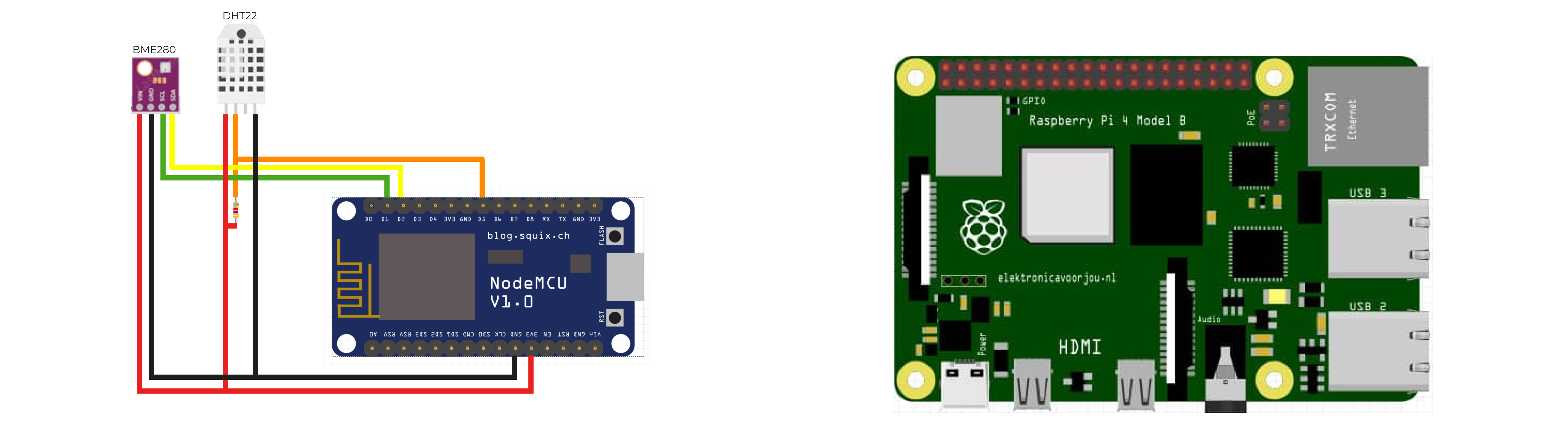
- Jumpers

- Cabos USB para o NodeMCU

- Fonte de alimentação para o Raspberry Pi

- (Opcional) Cabo HDMI, teclado e mouse para o Raspberry Pi

**Circuito eletrônico**



**Ligações**

BME280



- VCC no 3v3 do NodeMCU

- GND no GND do NodeMCU

- SCL no pino D1 do NodeMCU

- SDA no pino D2 do NodeMCU

DHT22



- VCC (primeiro pino da esquerda para direita) no 3v3 do NodeMCU

- Data (segundo pino da esquerda para direita) no pino D5 do NodeMCU e em uma perna do resistor de 4k7 Ohms. A outra perna do resistor é ligada no 3v3 do NodeMCU

- GND (quarto pino da esquerda para a direita) no GND do NodeMCU

**Configurações para o tago.io**

- Criar uma conta no tago.io

- Criar um dispositivo e pegar o nome do dispositivo e o token

- Criar um dashboard baseado nas variáveis recebidas (no caso, pressao\_atmosferica, temperatura e umidade)

**Configurações para a AWS**

- Criar uma conta na AWS

- Criar um bucket no S3

- Criar um usuário no IAM e criar uma access\_key e secret\_key ([How to Upload to an AWS S3 Bucket from a Raspberry Pi – ShillehTek](https://shillehtek.com/blogs/news/how-to-upload-to-an-aws-s3-bucket-from-a-raspberry-pi))

**Configurações para o Raspberry Pi**

- Configurar o sistema operacional do Raspberry Pi ([Raspberry Pi Headless Setup Tutorial (diyi0t.com)](https://diyi0t.com/raspberry-pi-headless-setup-tutorial/))

- (Opcional) configurar o acesso remoto do PC onde foi instalado o Arduino IDE para o Raspberry Pi ([Remote access - Raspberry Pi Documentation](https://www.raspberrypi.com/documentation/computers/remote-access.html))

- No Raspberry Pi, Instalar uma IDE ou editor de texto para criar os scripts em Python (o programa utilizado foi o Visual Studio Code)

- Verificar a versão instalada do Python (a usada aqui foi a 3.8)

- Instalar as seguintes bibliotecas do Python:

- paho-mqtt ([paho-mqtt · PyPI](https://pypi.org/project/paho-mqtt/))

- python-dotenv ([python-dotenv · PyPI](https://pypi.org/project/python-dotenv/))

- boto3 ([python-dotenv · PyPI](https://pypi.org/project/python-dotenv/))

- Configurar o Broker MQTT (foi utilizado o mosquitto - [Send data from ESP8266 or ESP32 to Raspberry Pi via MQTT (diyi0t.com)](https://diyi0t.com/microcontroller-to-raspberry-pi-wifi-mqtt-communication/))

- Passar os arquivos .env.example, get\_mqt\_data.py, send\_csv\_s3.py e node-red.json para o Raspberry Pi

- Criar o arquivo .env com base no .env.example

- Abrir o Node-RED, importar o node-red.json e configurar os nodes de MQTT do NodeMCU e do tago.io

**Configurações para o NodeMCU**

- Instalar o Arduino IDE para a programação no NodeMCU e configurar a placa do ESP8266 no Arduino IDE ([Installing ESP8266 in Arduino IDE (Windows, Mac OS X, Linux) | Random Nerd Tutorials](https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/))

- Instalar as seguintes bibliotecas no Arduino IDE ([Installing Libraries | Arduino Documentation](https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries/))

- DHT.h ([adafruit/DHT-sensor-library: Arduino library for DHT11, DHT22, etc Temperature & Humidity Sensors (github.com)](https://github.com/adafruit/DHT-sensor-library))

- PubSubClient ([knolleary/pubsubclient: A client library for the Arduino Ethernet Shield that provides support for MQTT. (github.com)](https://github.com/knolleary/pubsubclient))

- Adafruit BMP280 ([adafruit/Adafruit\_BMP280\_Library: Arduino Library for BMP280 sensors (github.com)](https://github.com/adafruit/Adafruit_BMP280_Library))

- ArduinoJson ([bblanchon/ArduinoJson: 📟 JSON library for Arduino and embedded C++. Simple and efficient. (github.com)](https://github.com/bblanchon/ArduinoJson))

- Montar o circuito do NodeMCU e conectar a porta USB do NodeMCU em alguma porta USB do PC onde instalou o Arduino IDE

- Criar o arquivo credentials.h usando como base o credentials\_example.h e preencher os valores

- Abrir o arquivo codigo-mqtt.ino no Arduino IDE, selecionar a placa do NodeMCU e a porta serial, e fazer o upload do .ino na placa.

**Referências**

- Apostilas das aulas

- [How to Upload to an AWS S3 Bucket from a Raspberry Pi – ShillehTek](https://shillehtek.com/blogs/news/how-to-upload-to-an-aws-s3-bucket-from-a-raspberry-pi)

- [Raspberry Pi Headless Setup Tutorial (diyi0t.com)](https://diyi0t.com/raspberry-pi-headless-setup-tutorial/)

- [Remote access - Raspberry Pi Documentation](https://www.raspberrypi.com/documentation/computers/remote-access.html)

- [paho-mqtt · PyPI](https://pypi.org/project/paho-mqtt/)

- [python-dotenv · PyPI](https://pypi.org/project/python-dotenv/)

- [python-dotenv · PyPI](https://pypi.org/project/python-dotenv/)

- [Send data from ESP8266 or ESP32 to Raspberry Pi via MQTT (diyi0t.com)](https://diyi0t.com/microcontroller-to-raspberry-pi-wifi-mqtt-communication/)

- [Installing ESP8266 in Arduino IDE (Windows, Mac OS X, Linux) | Random Nerd Tutorials](https://randomnerdtutorials.com/how-to-install-esp8266-board-arduino-ide/)

- [adafruit/DHT-sensor-library: Arduino library for DHT11, DHT22, etc Temperature & Humidity Sensors (github.com)](https://github.com/adafruit/DHT-sensor-library)

- [adafruit/Adafruit\_BMP280\_Library: Arduino Library for BMP280 sensors (github.com)](https://github.com/adafruit/Adafruit_BMP280_Library)

- [knolleary/pubsubclient: A client library for the Arduino Ethernet Shield that provides support for MQTT. (github.com)](https://github.com/knolleary/pubsubclient)

- [bblanchon/ArduinoJson: 📟 JSON library for Arduino and embedded C++. Simple and efficient. (github.com)](https://github.com/bblanchon/ArduinoJson)