

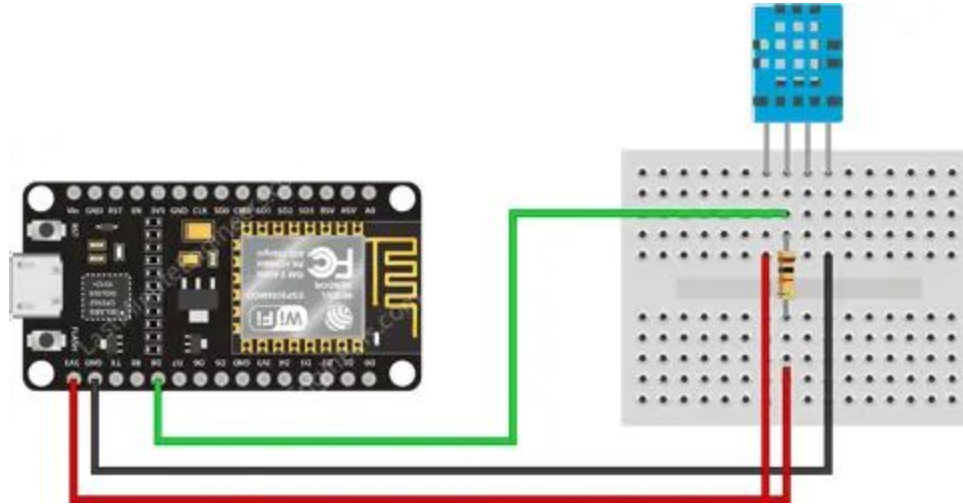
Desenvolvimento de um Dispositivo IoT para Monitoramento em Tempo Real de Umidade e Temperatura em Home Offices

Isabella de Freitas Nunes

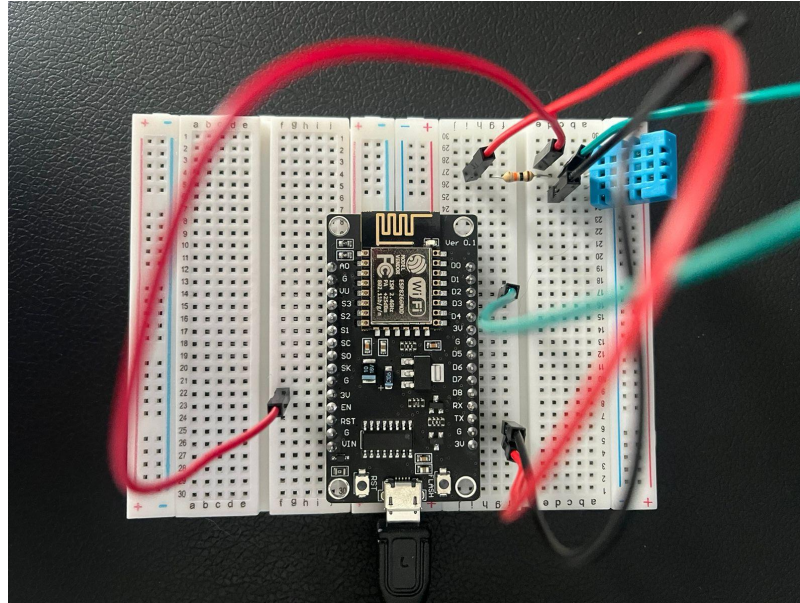
Introdução, Problema e Objetivos

- Desafios de aprendizagem teórica versus prática.
- Desenvolver um sistema IoT funcional que integra hardware e software para monitorar dados do home office do aluno, e transmiti-los utilizando o protocolo MQTT.
- Criar um cliente local que me permita usar o hardware e visualizar os dados pelo navegador na rede local.

Hardware (projeto)



Hardware (realidade)



Ambiente de desenvolvimento

Eclipse IDE for Embedded C/C++ Developers

387 MB 5,936 DOWNLOADS



An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (**Arm** and **RISC-V**) and debug plug-ins (**SEGGER J-Link**, **OpenOCD**, **pyocd**, and **QEMU**), plus a number of templates to create ready to run blinky projects.

To avoid compatibility issues with pre 6.x plug-ins, it is recommended to **create a new workspace** with the new version and **import the projects** there.




Windows | x86_64
macOS x86_64 | AArch64
Linux x86_64 | AArch64









Ambiente de desenvolvimento





ESP-IDF

Espressif Systems  espressif.com |  773,320 installs |     (114) | Free

Develop and debug applications for Espressif chips with ESP-IDF

Installation

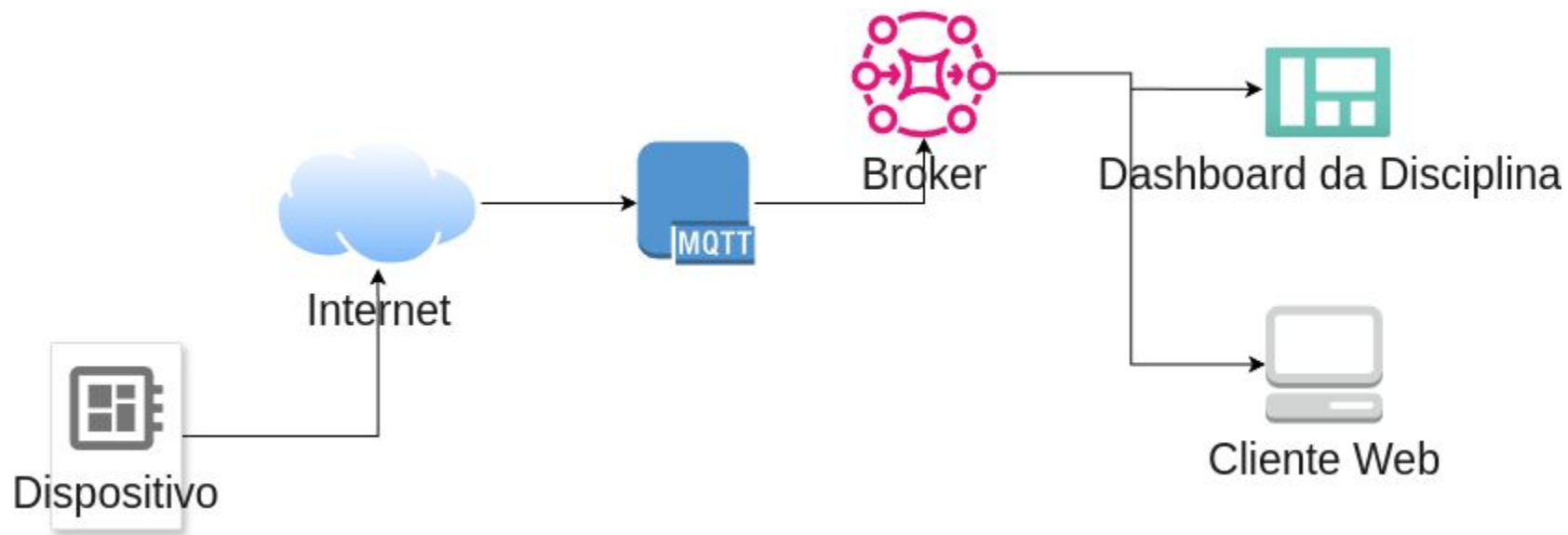
Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

`ext install espressif.esp-idf-extension`

Copy

[More Info](#)

Arquitetura



Código

```
#include <stdio.h>
```

```
#include "freertos/FreeRTOS.h"
```

```
#include "freertos/task.h"
```

```
#include "freertos/event_groups.h"
```

```
#include "freertos/queue.h"
```

```
#include "esp_system.h"
```

```
#include "esp_spi_flash.h"
```

```
#include "esp_wifi.h"
```

```
#include "esp_event_loop.h"
```

```
#include "nvs_flash.h"
```

```
#include "driver/gpio.h"
```

```
#include "mqtt_client.h"
```


Código

```
#define WIFI_SSID ""  
#define WIFI_PASS ""
```

```
#define MQTT_ENABLED 1  
#define MQTT_BROKER_URI ""  
#define INTERNAL_MQTT_BROKER_URI ""  
#define MQTT_TEMPERATURE_TOPIC ""  
#define MQTT_HUMIDITY_TOPIC ""
```

Código

```
#!/bin/zsh
```

```
make clean && make all && make  
flash && make monitor
```

Código

```
1: Inicializar NVS (Non-Volatile Storage)
2: Inicializar Wi-Fi com SSID e senha especificados
3: Inicializar sensor DHT11 no pino definido
4: Criar fila dht_queue para armazenar dados do sensor DHT
5: Criar tarefa dht_task:
6: while verdadeiro do
7:   Ler temperatura e umidade do DHT11
8:   Imprimir os valores lidos
9:   Enviar dados para a fila dht_queue
10:  Aguardar 10 segundos
11: end while
12: Criar tarefa mqtt_task:
13: Aguardar conexão Wi-Fi estabelecida
14: Inicializar cliente MQTT com configurações especificadas
15: while verdadeiro do
16:   Receber dados da fila dht_queue
17:   Publicar temperatura no tópico MQTT definido
18:   Publicar umidade no tópico MQTT definido
19: end while
```

Cliente local

ESCRITÓRIO



25 °C



40%

[illegible]

```
▼ isabella
  temperatura = 23
  umidade = 50
```

Conclusão

- Desafios de gerenciamento de tarefas, erros e leituras.
- O SDK carece de material disponível para iniciantes.
- Excelente desafio para aprendizagem.