

Laboratório 13 – IoT: Comunicação MQTT

1. OBJETIVOS

- Compreender o conceito principal dos sistemas IoT com um exemplo de caso de uso real.
- Implementar e aplicar os conceitos vistos em sala de aula sobre o protocolo MQTT usando a placa de desenvolvimento ESP32.
- Identificar e avaliar os recursos da ferramenta Node-RED, como os blocos de comunicação e o plug-in para a criação de painéis.

2. INTRODUÇÃO

A Internet das Coisas (IoT) descreve a rede de objetos físicos incorporados a sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet. Um dos protocolos mais usados para realizar essa comunicação é chamado MQTT (Message Queuing Telemetry Transport).

2.1. MQTT

O MQTT é um protocolo de transporte de mensagens de formato Cliente/Servidor que usa um modelo de publicação e inscrição; um publicador publica as mensagens em um tópico e um assinante irá se inscrever nele para poder ver a mensagem. O protocolo requer a utilização de um gerenciador dessas mensagens, o Broker, não há uma conexão direta entre o publicador e o assinante.

Esse protocolo se encaixa perfeitamente nesse contexto porque requer recursos mínimos, podendo ser usado até mesmo em pequenos microcontroladores, além disso a implementação do MQTT requer uma quantidade mínima de código que consome pouquíssima energia nas operações e também tem recursos integrados que oferecem suporte à comunicação com um grande número de dispositivos IoT.

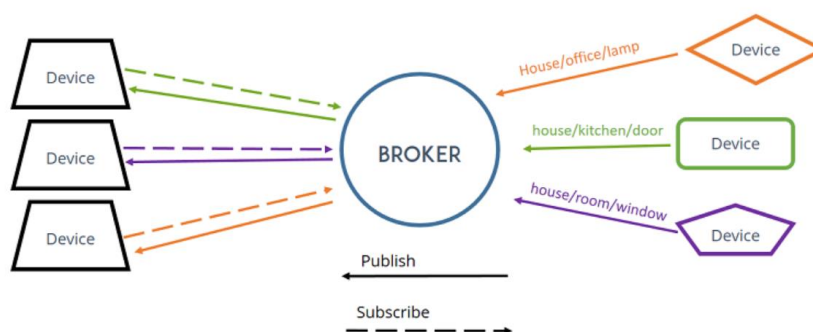


Figura 13.1. Arquitetura da rede MQTT.

Experiência No 13 - Sistemas Embarcados I

2.2. Node-RED

O Node-RED, é uma ferramenta visual de ambiente de código aberto, que inicialmente foi desenvolvida para implementar, criar e/ou conectar dispositivos de IoT. Assim sendo, por meio dos nodos ou nós é possível ler arquivos CSV, escutar eventos http, tcp, websocket, mqtt entre outros. O Node-RED oferece um editor de fluxo baseado em navegador que facilita a conexão de fluxos usando a ampla variedade de nós na paleta. Os fluxos podem ser implementados no tempo de execução com um único clique.

O tempo de execução leve foi desenvolvido com base no Node.js, aproveitando ao máximo seu modelo orientado a eventos e sem bloqueio. Os fluxos criados no Node-RED são armazenados usando JSON, que pode ser facilmente importado e exportado para ser compartilhado com outras pessoas.

3. MATERIAL UTILIZADO

Durante esta experiência, serão utilizados os seguintes equipamentos e componentes:

- Placa de desenvolvimento ESP32;
- Sensores:
 - 1 módulo Acelerômetro e Giroscópio de 3 Eixos (MPU-6050);
 - 1 sensor de temperatura e umidade (DHT11);
 - 1 LDR (*Light Dependent Resistor* - resistor dependente de luz);
 - 1 módulo sensor de distância ultrassônico (HC-SR04).
- Atuadores:
 - 3 Leds e 1 Led RGB;
 - 1 Buzzer;
 - 2 servo-motores;
 - 1 display OLED.
- Um *protoboard*;
- Resistores fixos com os seguintes valores de resistência: 330 Ω .
- Raspberry pi 4: usado para hospedar o broker.

4. PROCEDIMENTO

4.1. Para simplificar a compreensão, a Figura 8.2 (a) a (f) mostra a ligação de cada sensor e atuador a placa de desenvolvimento separadamente.

OBS.: A relação de pinos utilizados da ESP32 e seus correspondentes dispositivos pode ser encontrada na Tabela 8.1.

ELE15942 – Laboratório de Sistemas Embarcados I



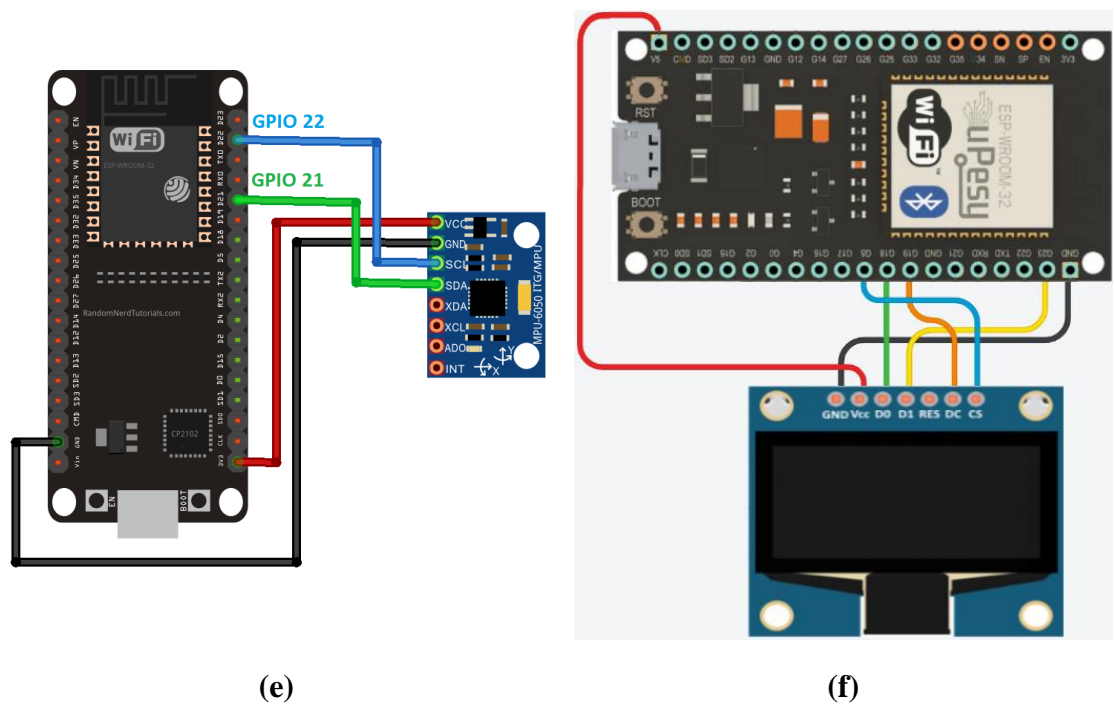


Figura 13.2 - Ligação dos dispositivos a placa de desenvolvimento. (a) Leds (b) Buzzer e Sensor de distância ultrassônico (c) Servo-motores (d) Sensor de temperatura e umidade e LDR (e) Acelerômetro/Giroscópio (f) Display OLED.

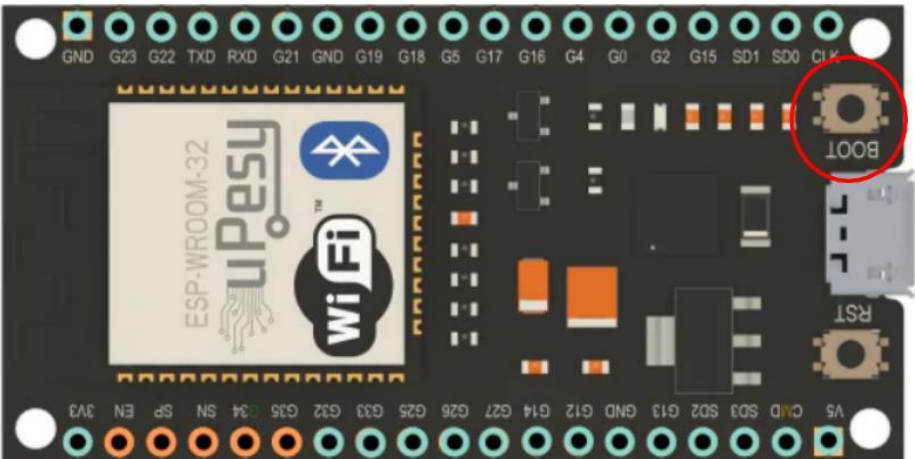
Tabela 8.1 - Relação de Pinos da ESP32 com os dispositivos usados

	Dispositivo	Pino ESP32
OUTPUTS	Led RGB - R	G2
	Led RGB - G	G15
	Led RGB - B	G0
	Led Branco	G17
	Led Vermelho	G16
	Led Verde	G4
	Buzzer	G12
	Servomotor 1	G25
	Servomotor 2	G26
	Display OLED - MOSI	G23
	Display OLED - CLK	G18
	Display OLED - DC	G19

Experiência No 13 - Sistemas Embarcados I

	Display OLED - CSI	G5
INPUTS	DHT	G14
	LDR	G34
	Acelerômetro - SDA	G21
	Acelerômetro - SCL	G22
	Ultrassom - Trigger	G33
	Ultrassom - Echo	G27

OBS.: Para programar a ESP32, em alguns casos, ao carregar o código é preciso manter pressionado o botão BOOT (Figura 6.3(a)) assim que a mensagem “Connecting...” aparecer no terminal da Arduino IDE (Figura 6.3(b)), até que seja iniciada a escrita na memória do ESP:



(a)

```
esptool.py v4.5.1
Serial port COM3
Connecting.....
```

(b)

Figura 8.3 – Programação da ESP32. (a) Localização do botão BOOT. (b) Mensagens no terminal da Arduino IDE.

4.2. Node-red

1. Instalação do node-red:

- a. Acesse: <https://nodered.org/docs/getting-started/windows> e siga as instruções passo a passo.
- b. Descarregue e instale o Node.js

Quick Start

1. Install Node.js

Download the latest LTS version of Node.js from the official [Node.js home page](#). It will offer you the best version for your system.

Run the downloaded MSI file. Installing Node.js requires local administrator rights; if you are not a local administrator, you will be prompted for an administrator password on install. Accept the defaults when installing. After installation completes, close any open command prompts and re-open to ensure new environment variables are picked up.

Once installed, open a command prompt and run the following command to ensure Node.js and npm are installed correctly.

Using Powershell: `node --version; npm --version`

Using cmd: `node --version && npm --version`

You should receive back output that looks similar to:

```
v18.15.0
9.5.0
```

- c. Instale Node-red

2. Install Node-RED

Installing Node-RED as a global module adds the command `node-red` to your system path. Execute the following at the command prompt:

```
npm install -g --unsafe-perm node-red
```

2. Abrir Node-red:

Experiência No 13 - Sistemas Embarcados I

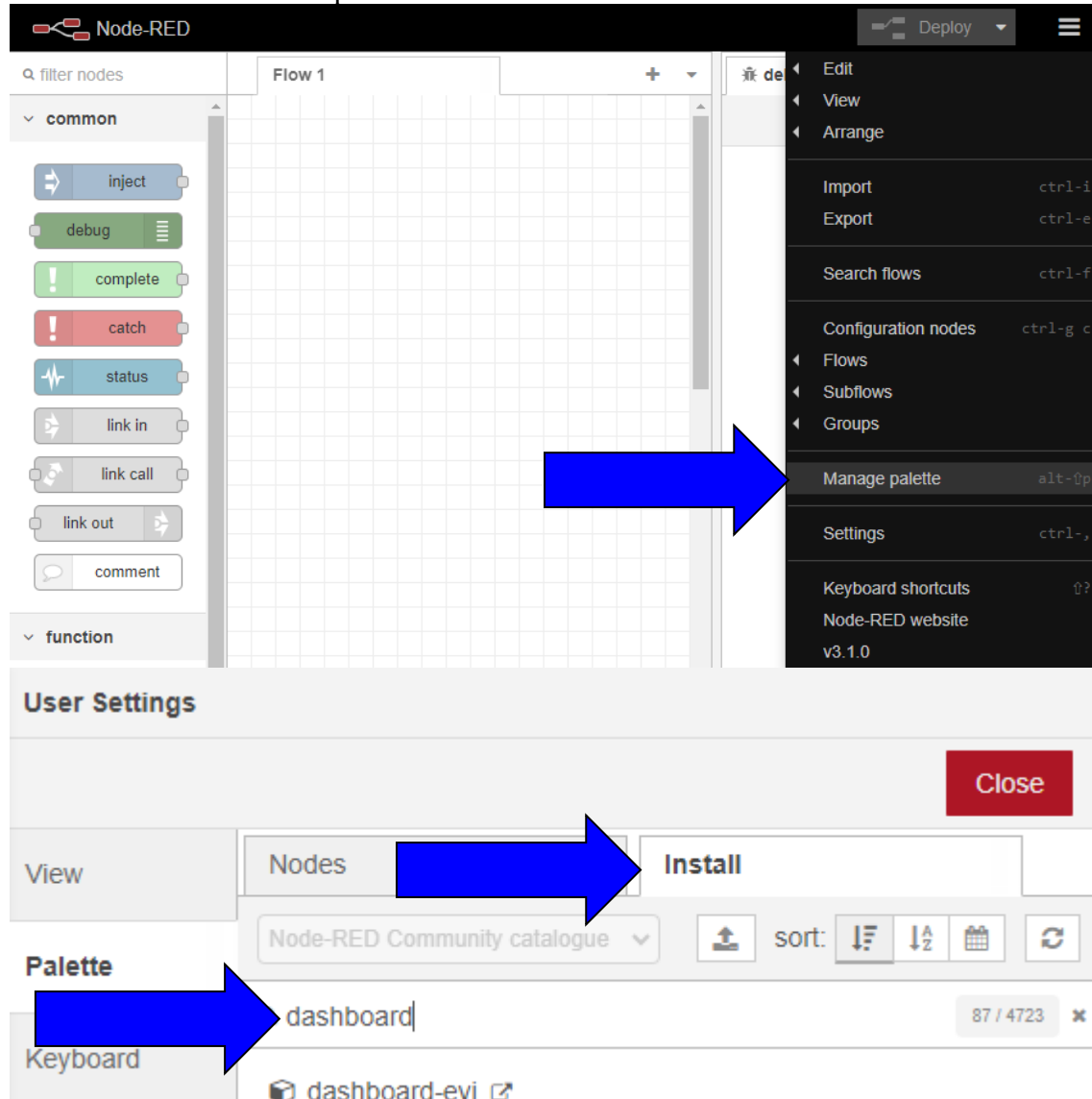
- a. No CMD escrever: **node-red**

```
C:\Users\...>node-red
9 Nov 17:27:21 - [info]

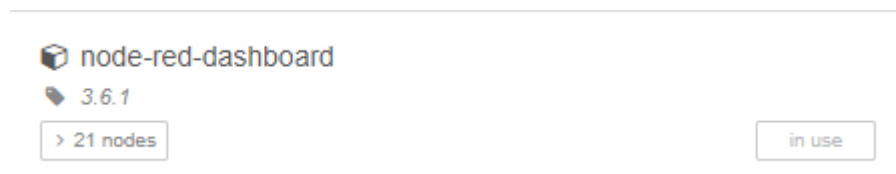
Welcome to Node-RED
=====
```

- b. No seu navegador escrever: **localhost:1880/**

3. Instale os “nodes” para usar el dashboard:



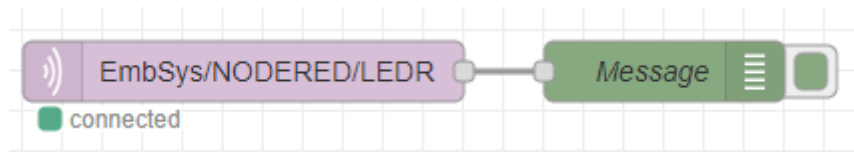
Instalar: node-red-dashboard



4. Criar o “Flow”

Experiência No 13 - Sistemas Embarcados I

- a. Como primeira experiência se recomenda usar o bloco “debug”:



- b. Os tópicos a usar são:

- i. EmbSys/NODERED/LEDR
- ii. EmbSys/NODERED/LEDG
- iii. EmbSys/NODERED/LEDB
- iv. EmbSys/NODERED/LEDW
- v. EmbSys/NODERED/LEDV
- vi. EmbSys/NODERED/LEDv
- vii. EmbSys/NODERED/BUZZ
- viii. EmbSys/NODERED/MOT1
- ix. EmbSys/NODERED/MOT2
- x. EmbSys/NODERED/OLED
- xi. EmbSys/ESP32/TEMP
- xii. EmbSys/ESP32/LDR
- xiii. EmbSys/ESP32/GIRX
- xiv. EmbSys/ESP32/GIRY
- xv. EmbSys/ESP32/GIRZ
- xvi. EmbSys/ESP32/ACLX
- xvii. EmbSys/ESP32/ACLY
- xviii. EmbSys/ESP32/ACLZ
- xix. EmbSys/ESP32/UDIS

Onde os sensores usam o “Subtopic” EPS32 e os atuadores usam o “Subtopic” NODERED

- c. A configuração do “Node” MQTT vai ser:

Experiência No 13 - Sistemas Embarcados I

Edit mqtt in node

Delete Cancel Done

Properties

Server: Raspi

Action: Subscribe to single topic

Topic: EmbSys/NODERED/LEDR

QoS: 0

Output: auto-detect (parsed JSON object, string or buf)

Name: Name

Properties

Name: Raspi

Connection Security Messages

Server: 192.168.67.75 Port: 1883

☒ Connect automatically

☐ Use TLS

Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☒ Use clean session

Endereço do broker

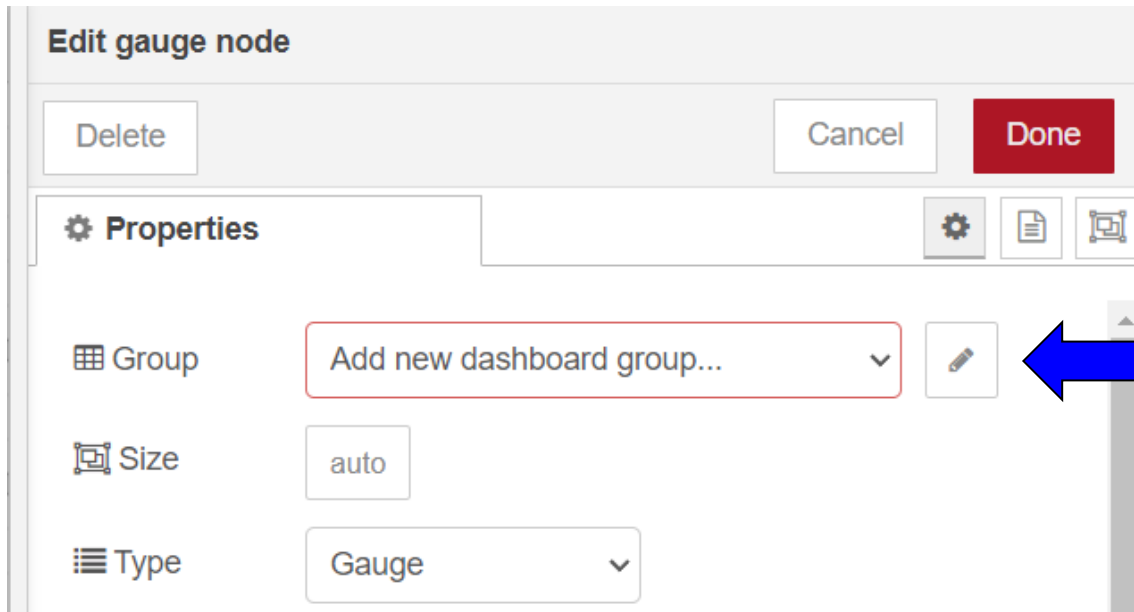
- i. Na parte de “Security” usar:
 1. Username: EMBSYS#N

Experiência No 13 - Sistemas Embarcados I

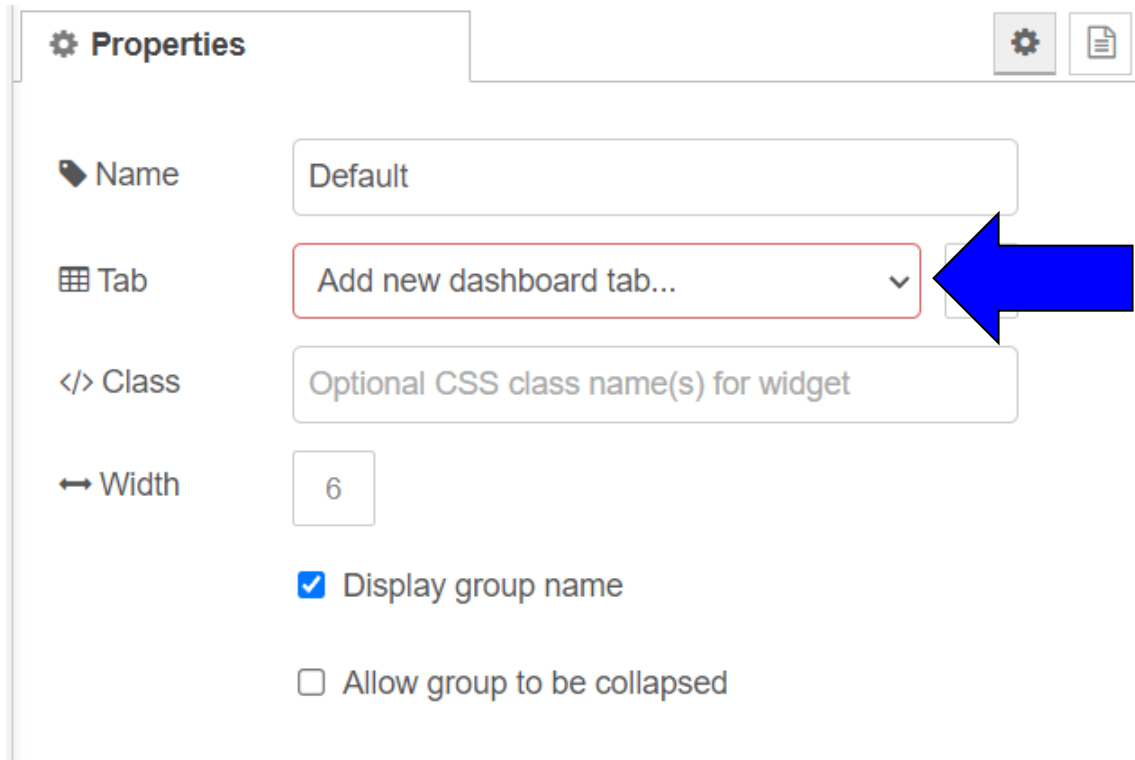
2. Password: EmSy###

Onde tem que trocar o “#” por seu número de grupo, se não ter solicite.

- d. Especificações para os “Nodes” de visualização:
 - i. Exemplo com um “Gauge”:



Experiência No 13 - Sistemas Embarcados I



Properties

Name: Default

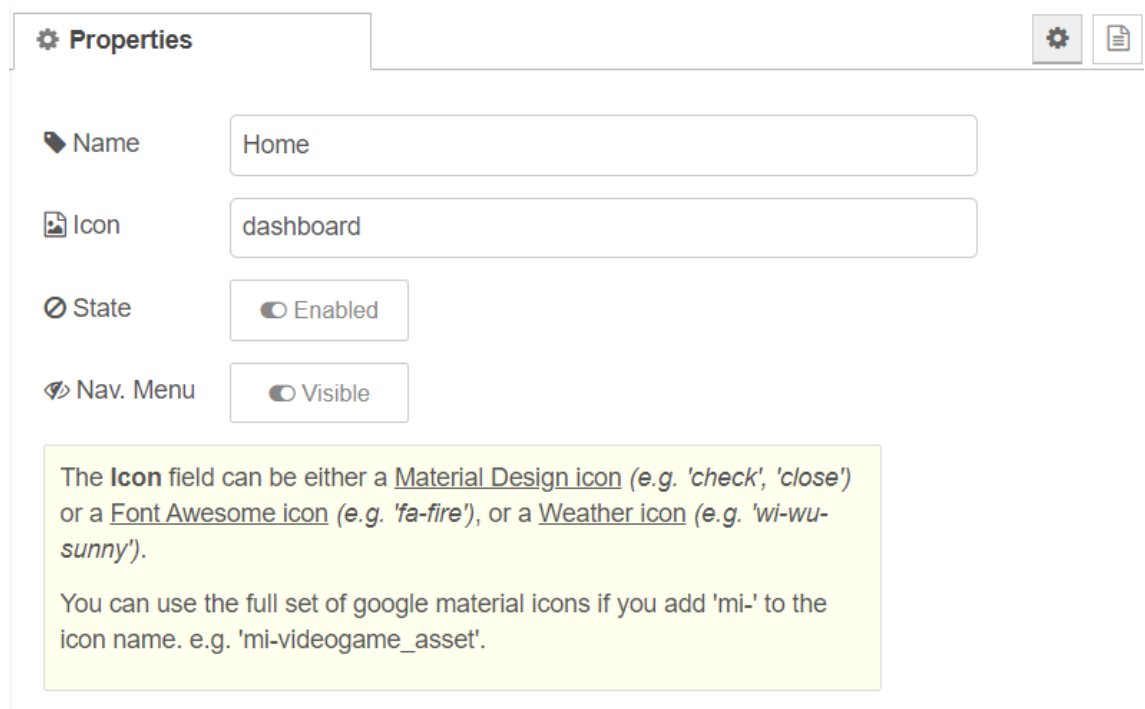
Tab: Add new dashboard tab...

Class: Optional CSS class name(s) for widget

Width: 6

☒ Display group name

☐ Allow group to be collapsed



Properties

Name: Home

Icon: dashboard

State: Enabled

Nav. Menu: Visible

The **Icon** field can be either a [Material Design icon](#) (e.g. 'check', 'close') or a [Font Awesome icon](#) (e.g. 'fa-fire'), or a [Weather icon](#) (e.g. 'wi-wu-sunny').

You can use the full set of google material icons if you add 'mi-' to the icon name. e.g. 'mi-videogame_asset'.

- ii. Lembre-se de trocar os parâmetros para que seu “Dashboard” seja mais entendível

4.3 Código de arduino.

Esta seção explicará as partes do código encarregadas da comunicação com o broker MQTT por meio dos tópicos.

As variáveis “ssid” e “password” são necessárias para a conexão do ESP32 à rede WIFI.

Experiência No 13 - Sistemas Embarcados I

```
// WiFi
const char *ssid = "PipeWiFi"; // Enter your WiFi name
const char *password = "pipe2000"; // Enter WiFi password
```

Na imagem a seguir estão todos os tópicos usados para postagem e assinatura.

```
// DEFINE DE TOPICS
const char *tLEDR = "EmbSys/NODERED/LEDR";
const char *tLEDG = "EmbSys/NODERED/LEDG";
const char *tLEDB = "EmbSys/NODERED/LEDB";
const char *tLEDW = "EmbSys/NODERED/LEDW";
const char *tLEDV = "EmbSys/NODERED/LEDV";
const char *tLEDv = "EmbSys/NODERED/LEDv";
const char *tBUZZ = "EmbSys/NODERED/BUZZ";
const char *tMOT1 = "EmbSys/NODERED/MOT1";
const char *tMOT2 = "EmbSys/NODERED/MOT2";
const char *tOLED = "EmbSys/NODERED/OLED";

const char *tTEMP = "EmbSys/ESP32/TEMP";
const char *tHUM = "EmbSys/ESP32/HUM";
const char *tLDR = "EmbSys/ESP32/LDR";
const char *tGIRX = "EmbSys/ESP32/GIRX";
const char *tGIRY = "EmbSys/ESP32/GIRY";
const char *tGIRZ = "EmbSys/ESP32/GIRZ";
const char *tACLX = "EmbSys/ESP32/ACLX";
const char *tACLY = "EmbSys/ESP32/ACLY";
const char *tACLZ = "EmbSys/ESP32/ACLZ";
const char *tUDIS = "EmbSys/ESP32/UDIS";
```

Nesse caso, é usada uma autenticação de nome de usuário/senha, conforme mostrado na imagem a seguir.

```
// MQTT AUTHENTICATION
const char *mqtt_username = "esp-32-0"; // username for authentication
const char *mqtt_password = "public"; // password for authentication
```

É implementada uma função de retorno de chamada, que é chamada sempre que uma mensagem é recebida e inclui a lógica de ativação dos diferentes atuadores.

Experiência No 13 - Sistemas Embarcados I

```
void callback(char* topic, byte* payload, unsigned int length) {
    String payloadR = "";
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
        payloadR+=(char)payload[i];
    }
    if (strcmp(topic,tLEDR)==0){
        if(payloadR == "on") ledOnOff('R', 1);
        if(payloadR == "off") ledOnOff('R', 0);
    }
}
```

Pode acontecer de a conexão entre o microcontrolador e o corretor ser perdida em alguns momentos, para os quais uma função de reconexão é implementada. Com a linha “client.subscribe(tópico)”, é possível assinar tópicos criados anteriormente.

```
void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("ESP32Client",mqtt_username, mqtt_password)) {
            Serial.println("connected");
            client.subscribe(tLEDR);
        }
    }
}
```

Na seção de configuração, é iniciada a conexão WiFi e selecionados o servidor e a função de retorno de chamada a serem usados.

```
setup_wifi();
client.setServer(mqtt_broker, 1883);
client.setCallback(callback);
```

Para publicar em um tópico, é usada a seguinte sintaxe: “client.publish(tópico, valor)” (os valores são enviados como caracteres), conforme mostrado na figura a seguir, no caso do sensor LDR. É recomendável usar um atraso após cada publicação.

```
client.publish(tLDR, String(ldrValue).c_str());
delay(100);
```

5. EXPERIENCIA NO LABORATORIO

Experiência No 13 - Sistemas Embarcados I

- a. Completar o código de Arduino.
- b. Usar o seu código final para inferir que mensagens de controle tem que ser enviadas.
- c. Usar a ferramenta de desenvolvimento Node-red, para gerar uma interface de usuário amigável na qual:
 - i. Os dados dos sensores serão plotados.
 - ii. Mensagens de controle serão enviadas.

OBS.: Cada grupo deve usar pelo menos um atuador dependendo do tamanho da turma, e o maior número possível de sensores.