

**ESCOLA SENAI “ROBERTO MANGE”**  
**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**YURI HENRIQUE REZENDE**

**PROJETO**  
**INTERNET DAS COISAS (IoT)**

**CAMPINAS**

**2023**

**YURI HENRIQUE REZENDE**

**PROJETO**  
**INTERNET DAS COISAS (IoT)**

Relatório apresentado à Escola Senai “Roberto  
Mange” como um dos requisitos avaliativos para obtenção da  
Graduação de Tecnólogo em Análise e Desenvolvimento de Sistemas

**Orientador:** Professor Michel De Moura Chaparro

**CAMPINAS**  
**2023**

**YURI HENRIQUE REZENDE**

**PROJETO**  
**INTERNET DAS COISAS (IoT)**

Relatório apresentado à Escola Senai “Roberto  
Mange” como um dos requisitos avaliativos para obtenção da  
Graduação de Tecnólogo em Análise e Desenvolvimento de Sistemas

Data da aprovação:

\_\_\_\_/\_\_\_\_/\_\_\_\_

Examinador:

---

Nome:

Cargo:

Instituição:

## RESUMO

A Internet of Things, ou Internet das Coisas (IoT), é uma revolução tecnológica que está transformando a maneira como interagimos com o mundo digital e físico. Ela representa a interconexão de dispositivos, objetos e sistemas através da internet, permitindo a coleta, troca e análise de dados em tempo real. A IoT está redefinindo a forma como vivemos, trabalhamos e interagimos com nosso ambiente, abrindo um vasto leque de possibilidades em diversos setores, desde a automação residencial e industrial até a saúde, agricultura, transporte e muito mais. Nesta introdução, exploraremos os princípios, aplicações e implicações da Internet das Coisas, destacando seu impacto significativo na sociedade e na economia global.

Por isso propus a montar um projeto simples e que se aplica no mundo social que vivemos.

O sistema monitora a temperatura ambiente por meio de um sensor conectado ao ESP32. Quando a temperatura atinge um limite predefinido, o ESP32 envia uma mensagem MQTT para controlar o ar-condicionado. O servidor MQTT (Mosquitto) atua como intermediário para permitir a comunicação entre os dispositivos. Esse sistema oferece uma solução eficiente e escalável para o controle de temperatura em ambientes IoT.

## LISTA DE ABREVIATURAS E SIGLAS

ESP32	“Espressif Systems ESP32 (microcontrolador)”
MQTT	“Message Queuing Telemetry Transport”
IOT	“internet of Things”

## SUMARIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>7</b>
<b>2</b>	<b>SITUAÇÃO ATUAL.....</b>	<b>7</b>
<b>3</b>	<b>SITUAÇÃO PROPOSTA.....</b>	<b>7</b>
<b>4</b>	<b>DESENVOLVIMENTO.....</b>	<b>8</b>
<b>5</b>	<b>CONCLUSÃO.....</b>	<b>11</b>

Você foi contratado para desenvolver um sistema de controle de ambiente residencial utilizando o ESP32 e a nuvem MQTT. O objetivo é monitorar a temperatura em diferentes cômodos da casa e acionar atuadores, como um ventilador, de acordo com as condições climáticas detectadas.

## 2 SITUAÇÃO PROPOSTA

Para isso, você precisa implementar as seguintes funcionalidades adicionais ao código anterior: Adicionar um tópico MQTT: "casa/ventilador" para controlar o acionamento dos atuadores. Configurar os pinos do ESP32 para conectar os atuadores (por exemplo, ventilador no pino 5. Assim que o valor da temperatura for lido do sensor DHT, o sistema deve compará-los com valores de referência predefinidos para determinar se é necessário acionar o ventilador.

Se a temperatura ambiente ultrapassar um limite superior (por exemplo, 30°C), o sistema deve publicar uma mensagem no tópico "casa/ventilador" com o payload "ligar" para acionar o ventilador. O sistema deve monitorar constantemente o tópico "casa/ventilador" para receber mensagens de controle enviadas externamente. Se uma mensagem com o payload "desligar" for recebida neste tópico, o sistema deve desligar o respectivo atuador. Com essa solução implementada, será possível controlar o ambiente residencial de forma automatizada, mantendo as condições climáticas confortáveis nos diferentes cômodos da casa.

## 1. Materiais usados no projeto:

LISTA DE MATERIAL	PREÇO
1 - Protoboard	R\$ 10,00
1- Led	R\$ 0,22
1- Resistor 10k	R\$ 1,14
1 - Resistor 220K	R\$ 0,09
Conjunto de Cabo Jumpers	R\$ 10,00
1 - ESP32	R\$ 43,00
1 - DHT22	R\$ 25,00
1 - Cabo micro usb	R\$ 11,00
<b>TOTAL:</b>	<b>R\$ 100,45</b>

## 2. Descrição do Hardware Utilizado:

Em outras palavras, a **protoboard** é uma placa de ensaio que serve como um protótipo de um aparelho eletrônico, com uma matriz de contatos que possibilita construir circuitos de teste sem que haja necessidade de solda e, assim, garantindo segurança e agilidade em diferentes atividades.

O termo **LED** significa “Light Emitting Diode” (Diodo Emissor de Luz, em português). Esse componente converte eletricidade em luz, e consome menos energia do que fontes de iluminação tradicionais, como lâmpadas incandescentes e fluorescentes.

**Resistor** ou uma resistência é um dispositivo elétrico muito utilizado em eletrônica, ora com a finalidade de transformar energia elétrica em energia térmica por meio do efeito joule, ora com a finalidade de limitar a corrente elétrica em um circuito.



Os **jumpers** são cabos ou fios elétricos com pontas devidamente preparadas para fazer as conexões elétricas entre os componentes de um circuito possibilitando a condução eletricidade ao longo dele.

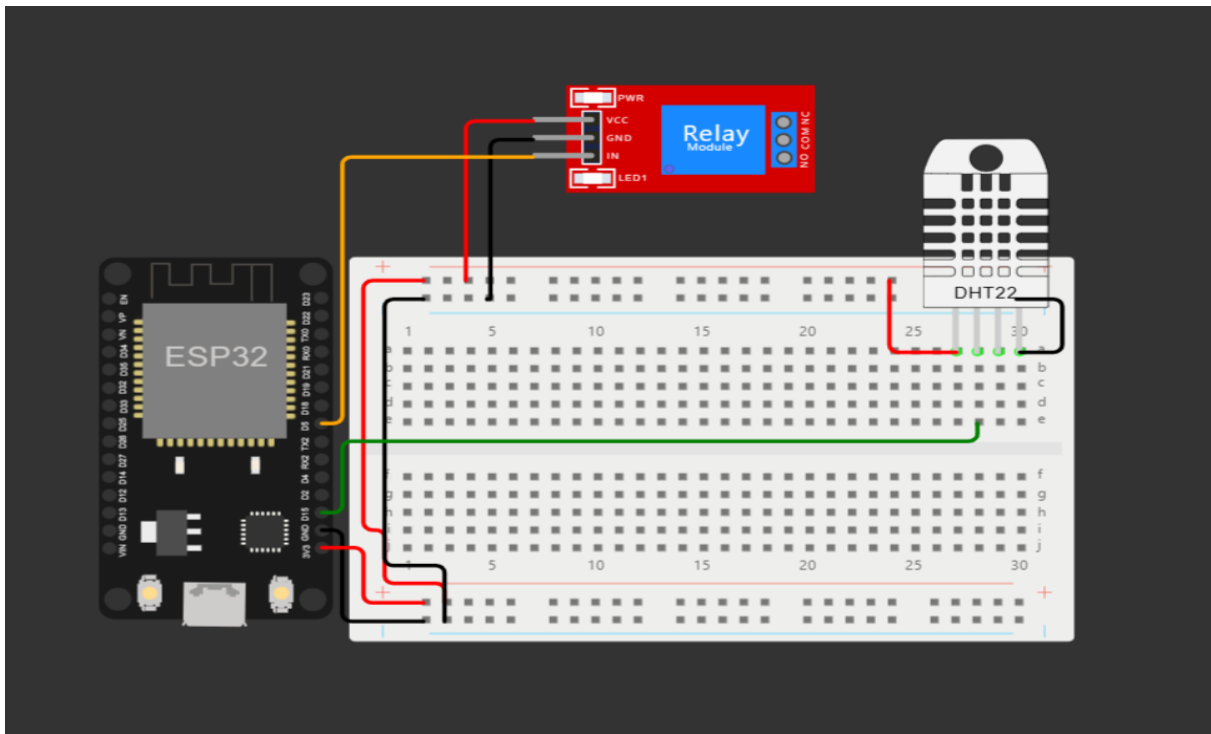
O **ESP32** pode ser alimentado por uma fonte de alimentação de 5V usando um regulador de tensão linear interno ou externo, ou por uma fonte de alimentação de 3,3V diretamente nos pinos de entrada de alimentação. A corrente máxima que pode ser fornecida pelos pinos de saída do ESP32 é de cerca de 40 mA.

O **DHT22** funciona através de um sensor capacitivo de umidade e um termistor para medir o ar circundante, todos enviando informações para um microcontrolador de 8 bits que responde com um sinal digital para outro microcontrolador.

O **Micro-USB** costumava ser a porta USB mais comum e ainda é encontrada em muitos modelos de Smartphone. Este tipo de conexão permite que os dados sejam lidos sem a necessidade de um computador. Observação: Os cabos Micro-USB só entrarão em uma porta na posição correta.

Um **relé**, ou, menos frequentemente, relê, é um interruptor eletromecânico projetado por Michael Faraday na década de 1830, com inúmeras aplicações possíveis em comutação de contatos elétricos, servindo para ligar ou desligar dispositivos.

### 3. Circuito eletrônico:



### 4. entradas e saídas:

ESP32:

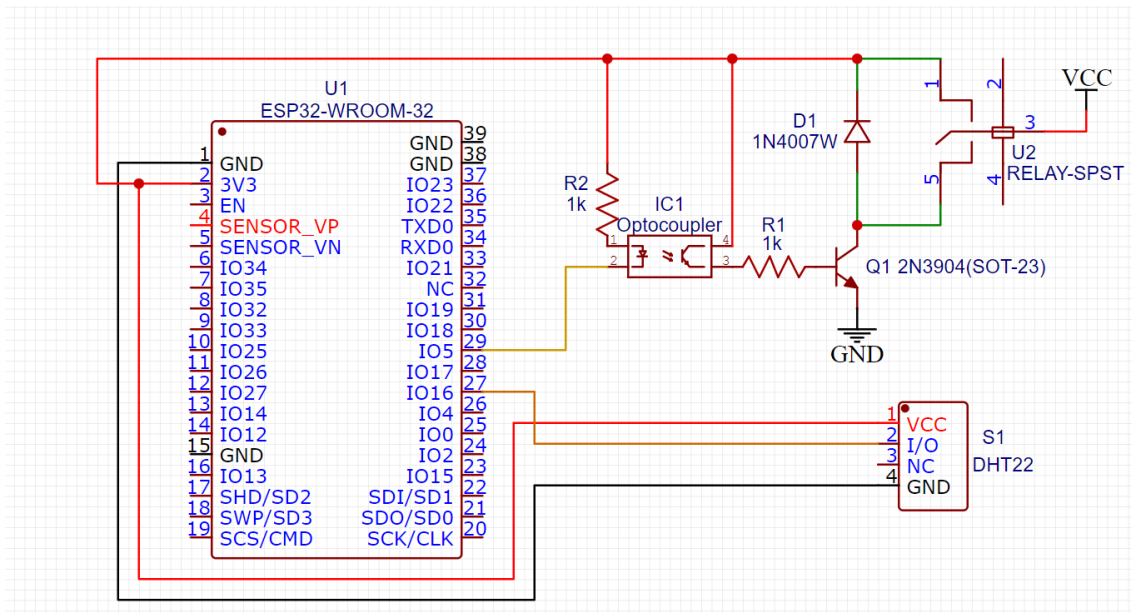
GND – Saida negativa dos circuitos para as ligações dos componentes;

3.3V – Saida positiva do circuito para as ligações dos componentes;

D15 – Entrada do sinal do sensor DHT;

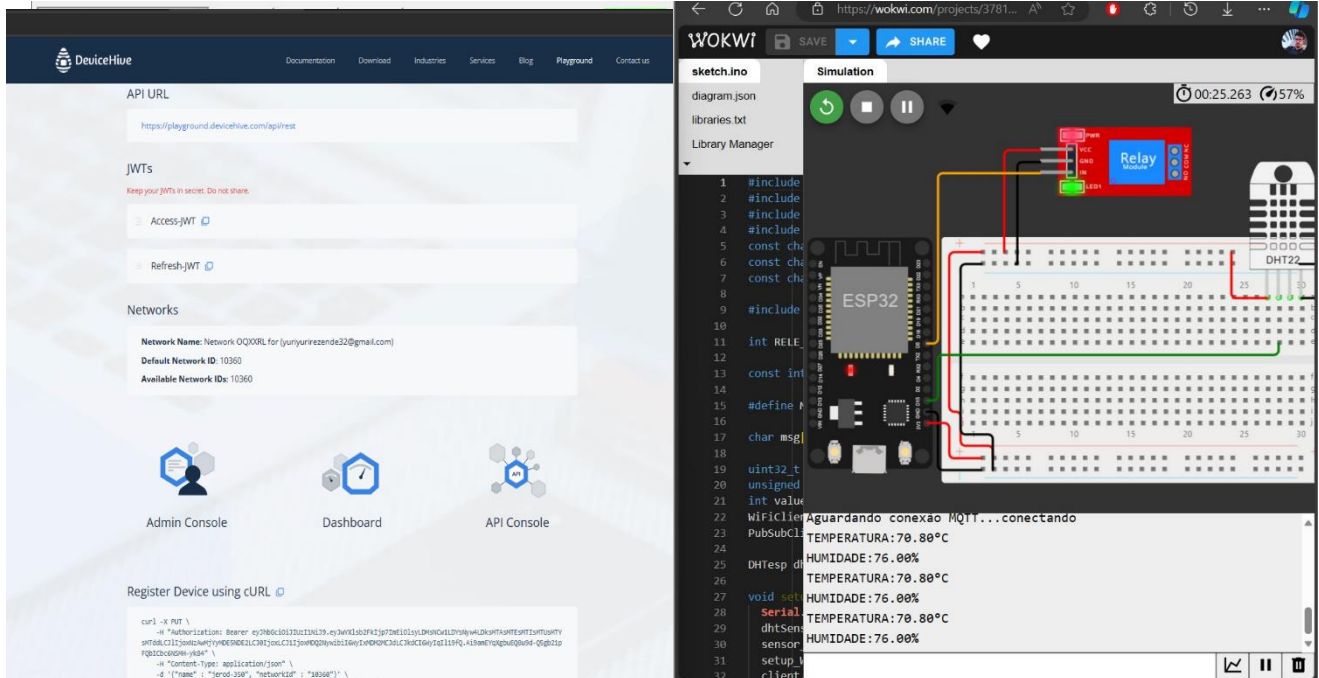
D6 – Saida para ligação do relé.

## 5. Circuito eletrônico de interfaceamento com o meio externo:



## 6. Utilizar o software de simulação para comprovar o funcionamento do circuito e da programação:

## 7.



## 8. Configurar o dispositivo IoT no broker:

ces / ulrz2afCpHakntMwhp69Q1ySqYCbfoPdZJkD

Device

Name: EAD

Network: Network OQXXRL for (yuriyurirezende32@gmail.com)

Device type: Thermostats

Data (json):

Operation: Normal

Edit

Commands

Notifications

Send new command

Name ⓘ	Time (UTC) ⓘ	Parameters ⓘ	Status ⓘ	Result ⓘ
This device does not have any commands				

## 9. Funcionamento do circuito de acordo com a programação do projeto:

O circuito coleta dados de temperatura e umidade com o sensor DHT22 e os envia para um servidor MQTT para monitoramento e controle. O relé permite que o ESP32 tome decisões com base nas leituras da temperatura, como ligar ou desligar um dispositivo externo para manter a temperatura desejada.

## 10. Comentar a programação para a documentação:

```
#include <PubSubClient.h> // Essa biblioteca permite que dispositivos baseados em Arduino se conectem a sistemas de mensagens e publiquem/assinem tópicos em um sistema de mensagens. //
#include <WiFi.h> // Essa biblioteca de Arduino que permite que dispositivos baseados em Arduino se conectem a redes Wi-Fi, possibilitando a comunicação com a Internet e outros dispositivos. //
#include "DHTesp.h" // Essa biblioteca específica para trabalhar com sensores de temperatura e umidade da série DHT (como DHT11, DHT22, etc.) //
const char* ssid = "yuri"; // E o nome da minha rede WIFI que eu quero conectar ao ESP //
const char* password = "123456789"; // A senha da minha rede wifi que eu quero conectar ao ESP //
const char* mqtt_server = "test.devicehive.org"; // Endereço do servidor MQTT real ao meu ESP. //
#define TOKEN "BBUS-yuNdFVTrbooy9bhA2w1Ej3rz38wV1"; // Token de acesso //
#define DEVICEID "65580362f088df06e19718a6"; // nome do meu ID //
int RELE_PIN = 5; // Estou definindo a porta do meu relé //
const int DHT_PIN = 15; // Estou definindo a porta do meu sensor de temperatura //
#define MSG_BUFFER_SIZE (50) // Essa linha especifica o tamanho de um buffer ou array de mensagens, garantindo que o tamanho seja consistente em todo o programa. //
char msg[MSG_BUFFER_SIZE]; // Cria um array de caracteres chamado msg com um tamanho específico, determinado pela constante MSG_BUFFER_SIZE que foi definida anteriormente no código com o tamanho de 50. //
uint32_t delayMS; // Variável para controlar um atraso ou temporização em milissegundos. //
unsigned long lastMsg = 0; // Variável que rastreia o tempo, em milissegundos ou outra unidade, na qual ocorreu a última ação ou evento. //
int value = 0; // Armazenar valores inteiros e a inicializa com o valor 0. //
WiFiClient espClient; // Serve para se conectar a um servidor remoto e enviar ou receber dados pela rede. //
PubSubClient client(espClient); // Comunica com um serviço de mensagens ou um sistema de publicação e assinatura, como o protocolo MQTT (Message Queuing Telemetry Transport) //
DHTesp dhtSensor; // Interage com um sensor de temperatura e umidade do tipo DHT //

void setup() { // É uma função que é executada automaticamente uma única vez quando o microcontrolador ESP é inicializado //
    Serial.begin(115200); // Configura a taxa de transmissão (baud rate) da porta serial, permitindo que você comunique-se com o ESP por meio da porta serial //
    dhtSensor.setup(DHT_PIN, DHTesp::DHT22); // Configura e inicializa um sensor de temperatura e umidade DHT22 conectado ao ESP //
    setup_WiFi(); // Chamo minha função setup_WiFi para funcionar 1 vez se conectada //
    client.setServer(mqtt_server, 1883); // Estabelece uma conexão com um servidor MQTT //
    pinMode(RELE_PIN, OUTPUT); // Estou configurando minha porta digital como saída, ou seja meu relé, ele vai mandar ligado ou desligado //
} // Finaliza minha função

void setup_WiFi() { // Essa função vai configurar meu ESP a conectar na rede WIFI //
    delay(10); // Atraso de 10 milissegundos para seguir o paradigma da programação //
    Serial.println(""); // Imprime uma linha em branco (ou seja, uma linha vazia) na porta serial //
    Serial.print("conectando com"); // Simplesmente imprime a string "conectando com" na porta serial sem adicionar uma nova linha no final da mensagem. //
    Serial.println(ssid); // Imprime o nome da minha rede //
    WiFi.begin(ssid, password); // Inicia a conexão Wi-Fi //
    while (WiFi.status() != WL_CONNECTED) { // Esse while vai rodar infinitamente até que a conexão Wi-Fi seja estabelecida //
        delay(500);
        Serial.print(".");
    } // Finaliza meu loop while
    Serial.println(""); // Imprime uma linha em branco (ou seja, uma linha vazia) na porta serial //
    Serial.print("WiFi conectado"); // Simplesmente imprime a string "WiFi conectado" na porta serial sem adicionar uma nova linha no final da mensagem. //
    Serial.println("IP: "); // Simplesmente imprime a string "IP: " na porta serial //
    Serial.println(WiFi.localIP()); // Imprime na frente do "IP: " o IP que se conecta, e imprime à porta serial sem adicionar uma nova linha no final da mensagem //
} // Finaliza minha função //

void reconnect() { // Essa função vai conectar como meu protocolo MQTT //
    while (!client.connected()) { // Esse while é usado para aguardar até que uma conexão MQTT seja
        // estabelecida com sucesso antes de continuar com a execução do programa. Ele também lida com falhas na conexão, tentando novamente após um curto atraso //
        Serial.print("Aguardando conexão MQTT...");
        String clientId = "ESP32Client";
        clientId += String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("conectando");
        } else {
            Serial.print("falhou, rc=");
            Serial.print(client.state());
            Serial.print("tente novamente em 5s");
            delay(500);
        }
    }
} // Finaliza minha função //
```

```

void loop() { // Essa função executada continuamente após a inicialização do ESP e é responsável por conter o código principal do programa. //
  delay(delayMS); // Criar um atraso (ou pausa) na execução. //
  TempAndHumidity data = dhtSensor.getTempAndHumidity(); // Essa linha de código está lendo os dados de temperatura e umidade do sensor DHT e armazenando esses valores na variável data. //
  if (isnan(data.temperature)) { // É uma verificação de erro que testa se o valor da temperatura lido a partir do sensor DHT não é um número válido (NaN, "Not-a-Number"). //
    Serial.println(F(" ERRO NA LEITURA DA TEMPERATURA"));
  } else { // Esse else lida com as leituras de temperatura do sensor DHT, exibe a temperatura na porta serial, //
    // e publica em um tópico MQTT e toma decisões com base na temperatura em relação a um limite de 22 graus Celsius, //
    // que são representadas como "1" (ligado) ou "0" (desligado) no tópico MQTT "yuri/ligadotemp" //
    Serial.print(F("TEMPERATURA:"));
    Serial.print(data.temperature);
    Serial.println(F("°C"));
    sprintf(msg, "%f", data.temperature);
    client.publish("yuri/temperature", msg);

    if (data.temperature > 22) {
      sprintf(msg, "%i", 1);
      client.publish("yuri/ligadotemp", msg);
    } else {
      sprintf(msg, "%i", 0);
      client.publish("yuri/ligadotemp", msg);
    }
  }
}

if (isnan(data.humidity)) { // É uma verificação de erro que testa se o valor da humidade lido a partir do sensor DHT não é um número válido (NaN, "Not-a-Number"). //
  Serial.println(F(" ERRO NA LEITURA DA HUMIDADE"));
} else { // Esse else lida com as leituras de humidade do sensor DHT, exibe a humidade na porta serial, //
  // e publica em um tópico MQTT e toma decisões com base na humidade em relação a um limite de 22 graus Celsius, //
  // que são representadas como "1" (ligado) ou "0" (desligado) no tópico MQTT "yuri/ligar" //
  Serial.print(F("HUMIDADE:"));
  Serial.print(data.humidity);
  Serial.println(F("%"));
  sprintf(msg, "%f", data.humidity);
  client.publish("yuri/humidity", msg);

  if (data.humidity > 50) {
    sprintf(msg, "%i", 1);
    client.publish("yuri/ligar", msg);
  } else {
    sprintf(msg, "%i", 0);
    client.publish("yuri/ligar", msg);
  }
}

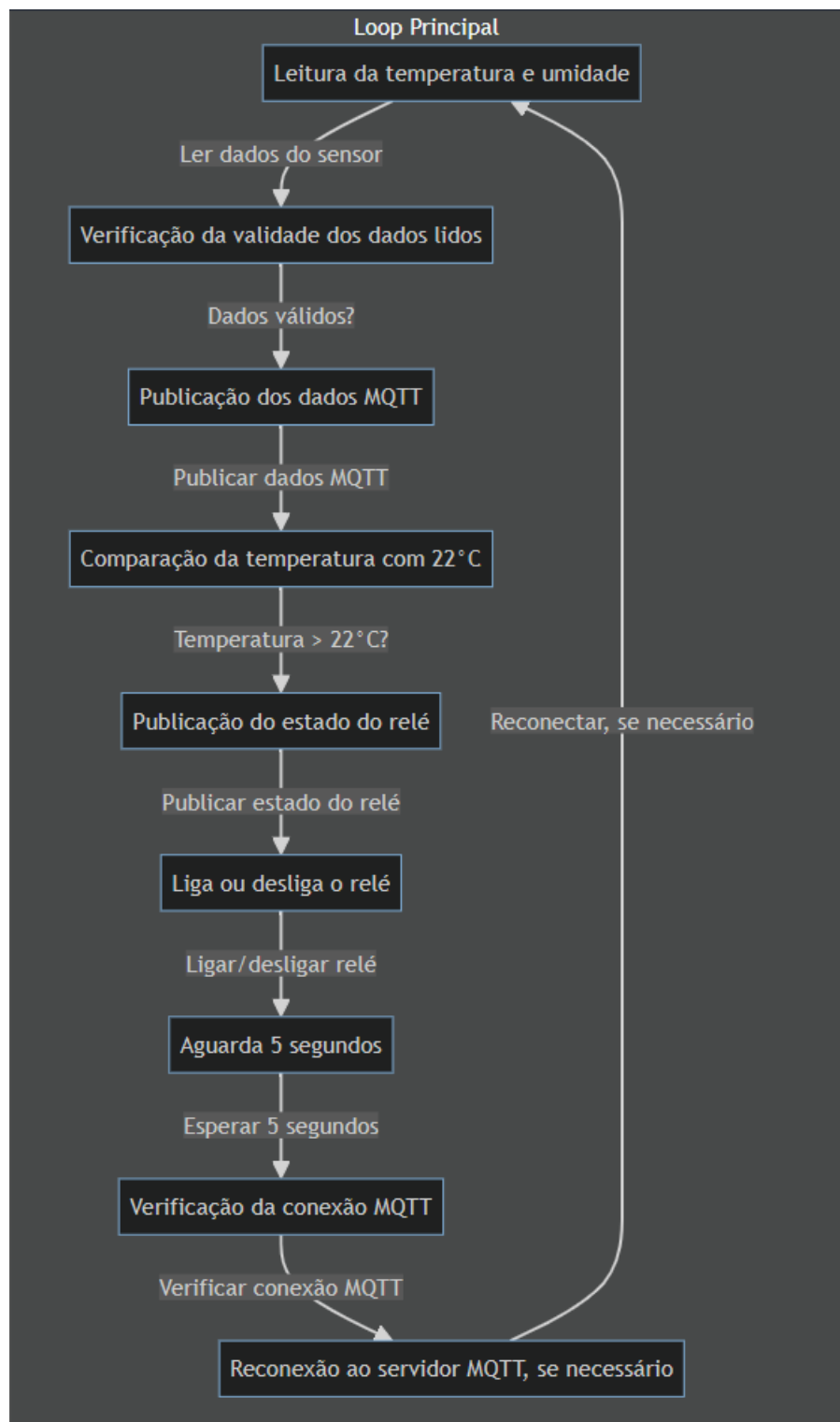
delay(2000); // Atraso de 2 segundos para seguir o paradigma da programação //

if (data.temperature >= 22) { // Esse if controla o Relé (led) com base na leitura da temperatura. Se a temperatura for maior ou igual a 22 graus Celsius, //
  // o dispositivo é ativado em 5 segundos, e se a temperatura estiver abaixo desse limite, o dispositivo é desativado em 5 segundos. //
  digitalWrite(RELE_PIN, HIGH);
  delay(5000);
} else {
  digitalWrite(RELE_PIN, LOW);
  delay(5000);
}

if (!client.connected()) { // If serve para garantir que o cliente MQTT esteja sempre conectado ao servidor. //
  // Se a conexão MQTT for perdida por algum motivo (por exemplo, devido a uma interrupção na rede, que de energia), //
  // o código verifica essa condição e chama a função reconnect() para tentar reconectar automaticamente o cliente. //
  // Isso é útil para manter a comunicação MQTT estável e confiável, mesmo em situações de falha temporária na rede. //
  reconnect();
}
client.loop(); // Permite que o cliente MQTT mantenha uma comunicação ativa e confiável com o servidor MQTT. //
}


```

## 11. Elaborar o Fluxograma do programa.



## 12. Configurar plataformas para utilização em nuvem.

### MQTT Dash



Default (automatically connect on start up).  
Note: this option is useful if you have just one connection configured.

☐ If you have more than one connection, you can create home screen shortcut for every connection.  
To create shortcut long press on any connection in connections list.

☒ Keep screen on when connected to this broker

Allow metrics management. If disabled, you can't add, edit, delete or rearrange metrics.

☒ This serves as protection from unintentional metrics changing.

Name

sensor

Address

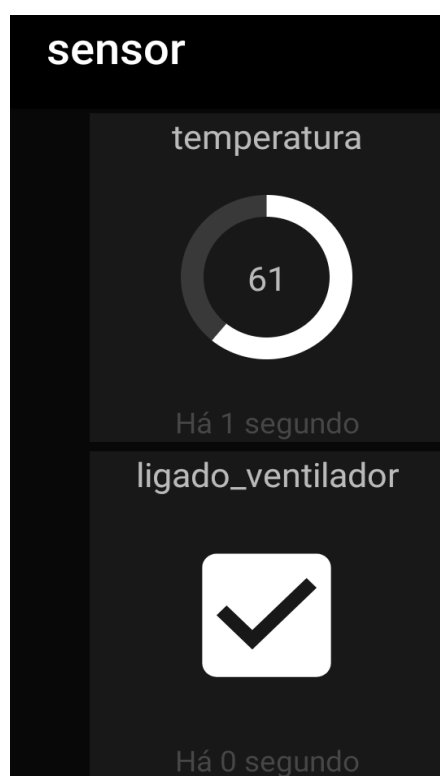
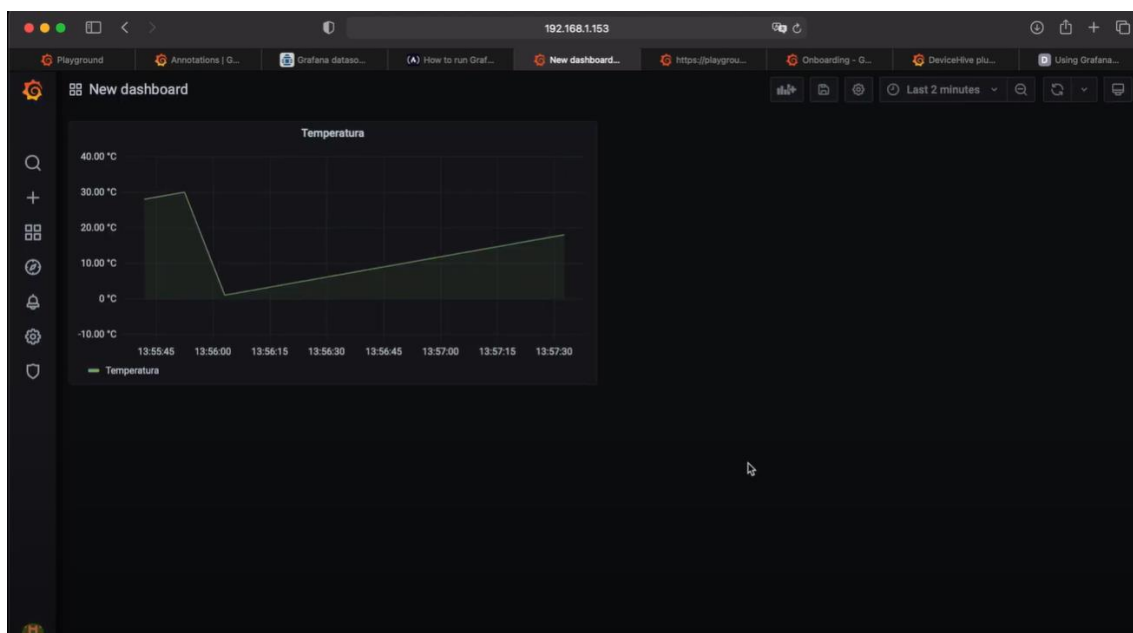
test.devicehive.org

Port

1883



### 13. Criar dashboard dos dados coletados por meio de plataformas em nuvem.



Este projeto é um exemplo prático de como um microcontrolador (ESP32) pode ser usado para criar um sistema de automação residencial simples. Ele pode ser adaptado para controlar sistemas de climatização, aquecimento ou resfriamento com base na leitura da temperatura ambiente. Além disso, fornece a capacidade de monitorar e controlar o sistema remotamente por meio do servidor MQTT, tornando-o adequado para aplicações de IoT residencial.