



SISTEMA DE MONITORAMENTO REMOTO DE ECG UTILIZANDO IOT:
INOVAÇÃO TECNOLÓGICA EM INFRAESTRUTURA DE SAÚDE

Thiago Oliveira, André Oliveira, Leandro Fernandes

Universidade Presbiteriana Mackenzie (UPM) Rua da Consolação, 930 Consolação, São
Paulo - SP, 01302-907 – Brazil

{10408614}@mackenzista.com.br

Abstract.

This paper presents an IoT-based project aimed at improving the efficiency and sustainability of industrial infrastructure, aligning with the Sustainable Development Goal 9 (SDG 9), which focuses on building resilient infrastructure, promoting inclusive industrialization, and fostering innovation. The proposed solution involves the use of sensors and actuators to monitor key parameters in industrial environments, such as temperature, vibration, and energy consumption, enabling predictive maintenance and reducing operational downtime. The communication between devices is facilitated through the MQTT protocol, ensuring real-time data transmission. This project demonstrates the potential of IoT technologies to optimize industrial processes and reduce environmental impacts.

RESUMO

Este artigo apresenta um projeto baseado em IoT com o objetivo de melhorar a eficiência e a sustentabilidade das infraestruturas industriais, alinhado com o Objetivo de Desenvolvimento Sustentável 9 (ODS 9), que foca na construção de infraestruturas resilientes, promoção da industrialização inclusiva e incentivo à inovação. A solução proposta envolve o uso de sensores e atuadores para monitorar parâmetros críticos em ambientes industriais, como temperatura, vibração e consumo de energia, permitindo a manutenção preditiva e a redução de tempo de inatividade operacional. A comunicação entre os dispositivos é realizada por meio do protocolo MQTT, garantindo a transmissão de dados em tempo real. Este projeto demonstra o potencial das tecnologias IoT para otimizar processos industriais e reduzir impactos ambientais.

1 INTRODUÇÃO

A industrialização sustentável é essencial para enfrentar os desafios ambientais e econômicos do século XXI. O Objetivo de Desenvolvimento Sustentável 9 (ODS 9), definido pela Organização das Nações Unidas, visa promover uma industrialização sustentável, construir infraestruturas resilientes e incentivar a inovação tecnológica. Para que essas metas sejam atingidas, é necessário explorar soluções tecnológicas que aumentem a eficiência e reduzam os impactos ambientais.

Um dos grandes desafios enfrentados por indústrias e infraestruturas modernas é a necessidade de manutenção eficiente e o monitoramento em tempo real de equipamentos. A falha de maquinário pode acarretar enormes custos operacionais, além de impactos negativos no meio ambiente devido ao uso ineficiente de energia e recursos. Nos últimos anos, a **Internet das Coisas (IoT)** tem se mostrado uma solução promissora, permitindo a coleta e análise de dados em tempo real por meio de sensores distribuídos em ambientes industriais (AYAZ et al., 2019).

Este projeto propõe a implementação de um sistema IoT para monitoramento e automação de infraestrutura industrial. Utilizando sensores para medir parâmetros como temperatura, vibração e consumo de energia, é possível aplicar manutenção preditiva e otimizar o uso de recursos, resultando em maior eficiência e sustentabilidade. A comunicação entre os dispositivos será realizada por meio do protocolo MQTT, devido à sua leveza e confiabilidade em aplicações IoT.

2.1 Objetivo do Projeto

O principal objetivo deste projeto é desenvolver um sistema de monitoramento em tempo real, utilizando IoT, que permita a otimização da operação de maquinários industriais. Isso envolve a coleta de dados de sensores distribuídos pela infraestrutura, com foco na identificação precoce de falhas (manutenção preditiva) e na redução do consumo de energia.

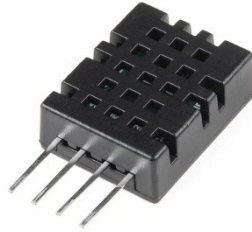
Para a implementação do projeto, serão utilizados os seguintes componentes:

- 1. NodeMCU (ESP8266):** A plataforma NodeMCU será utilizada para controlar o sistema IoT. Esta placa baseada no ESP8266 oferece conectividade Wi-Fi e é adequada para o envio de dados dos sensores para a nuvem via protocolo MQTT. Além disso, sua facilidade de programação torna-a ideal para prototipagem rápida.



2.3 Descrição dos Componentes de Hardware e Software

Sensor de Temperatura e Umidade (DHT11): Esse sensor será responsável por monitorar as condições ambientais da planta industrial, permitindo o controle do ambiente de trabalho das máquinas, evitando superaquecimento, usando como base o Mosquitto MQTT para fazer a troca de informação tanto com minha rede de WI-FI como a própria rede interna do computador via Mosquitto.



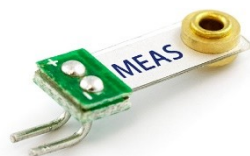
<https://www.sparkfun.com/products/18364>

Atuador (Relé): Este relé será utilizado para acionar dispositivos de controle, como desligar automaticamente máquinas que apresentam condições perigosas ou acionar sistemas de alerta.



<https://www.sparkfun.com/products/14538>

Sensor de Vibração: Utilizado para detectar vibrações em máquinas. Vibrações anormais são um indicador de desgaste ou falha iminente, permitindo que a manutenção preditiva seja realizada. Usando como base o Mosquitto MQTT para fazer a troca de informação tanto com minha rede de WI-FI como a própria rede interna do computador via Mosquitto.



<https://www.sparkfun.com/products/9199>

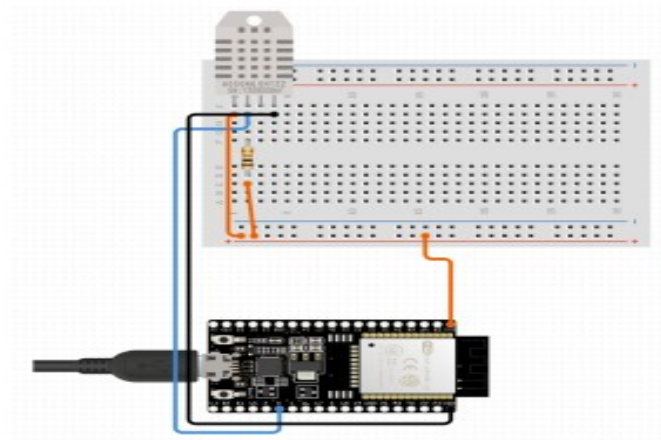
Protocolo MQTT: Para garantir uma comunicação eficiente entre os sensores e o servidor central, o protocolo MQTT será implementado. O MQTT é amplamente utilizado em aplicações de IoT devido à sua leveza e confiabilidade, permitindo a troca de mensagens em tempo real. Usando como base o Mosquitto MQTT para fazer a troca de informação tanto com minha rede de WI-FI como a própria rede interna do computador. O MQTT é um dos melhores protocolos de rede para dispositivos de Internet das Coisas (IoT), tendo sido projetado para um transporte extremamente leve de mensagens. Com o MQTT, o dispositivo pode “ouvir” e ser notificado apenas quando houver alguma alteração na variável.

Ferramentas e Métodos:

Programação: A plataforma NodeMCU será programada utilizando o **Arduino IDE**. Os sensores serão conectados à placa e os dados transmitidos via Wi-Fi para um servidor central utilizando o protocolo MQTT.

Coleta e Análise de Dados: Os dados coletados em tempo real pelos sensores serão transmitidos para um servidor central onde serão armazenados e analisados. Essa análise permitirá identificar padrões de operação anormais que indicam a necessidade de manutenção ou ajustes no uso de energia.

Esquema de Montagem: O esquema de montagem do sistema será desenvolvido utilizando o software **Fritzing**, que facilita a visualização das conexões entre os componentes.



Descrição do Funcionamento

1. Conexão

Wi-Fi:

- O ESP32 conecta-se à rede Wi-Fi especificada (STUX_NET2 . 4) utilizando as credenciais fornecidas.
- Se a conexão falhar, ele tenta novamente até ser bem-sucedido.

2. Leitura dos Sensores:

- O sensor DHT11 coleta informações de temperatura e umidade.
- Os dados brutos são obtidos por meio da biblioteca DHT.

3. Processamento Local:

- Os valores de temperatura e umidade são armazenados em arrays circulares para cálculo da média móvel dos últimos 5 valores.

- Se o sensor falhar ao fornecer dados, o sistema espera 2 segundos antes de tentar novamente.

4. Cálculo da Média Móvel:

- Um algoritmo percorre as leituras armazenadas e calcula a média apenas dos valores válidos.
- Os resultados ajudam a filtrar flutuações indesejadas nos dados.

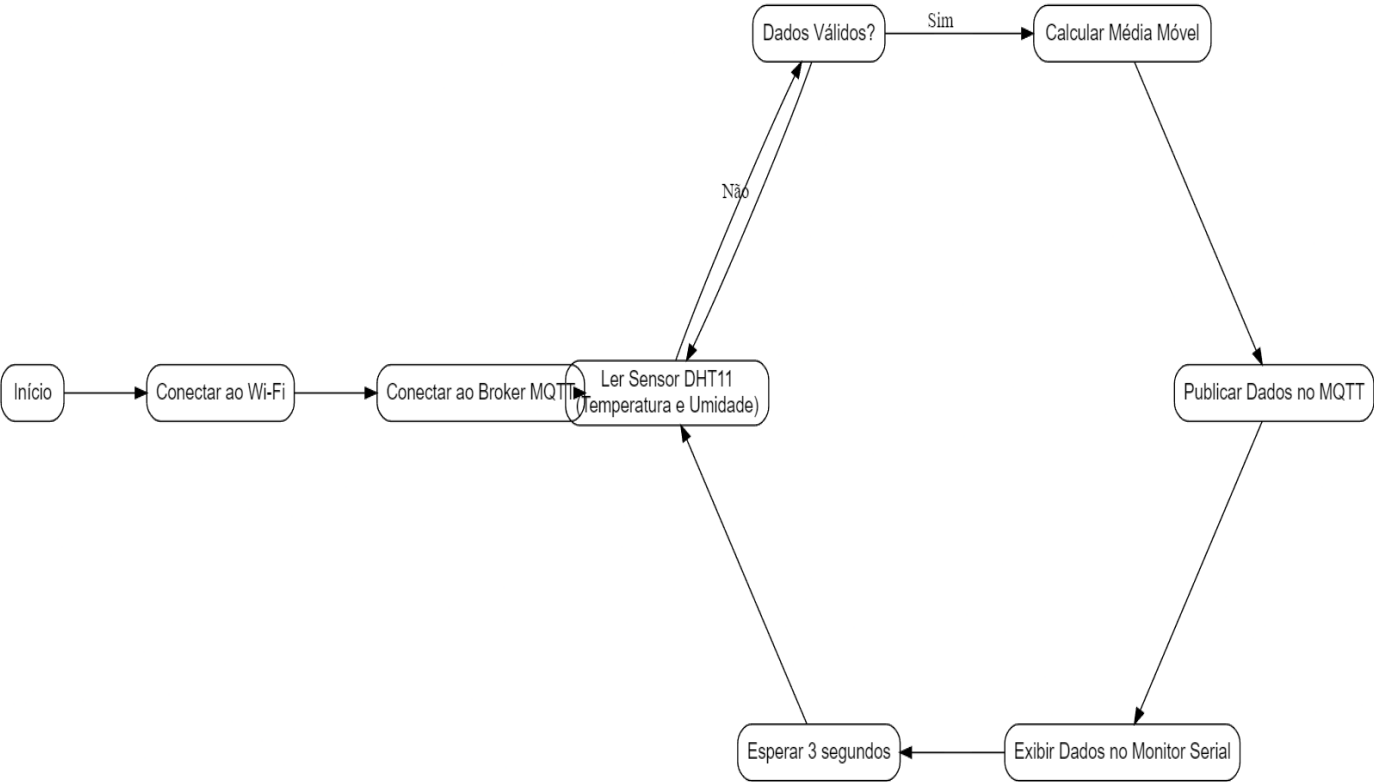
5. Envio de Dados via MQTT:

- As leituras instantâneas e as médias móveis são convertidas para strings.
- As informações são publicadas em tópicos específicos no broker MQTT (`esp32/temperature`, `esp32/humidity`, etc.).
- Se a conexão com o broker estiver perdida, o sistema tenta reconectar automaticamente.

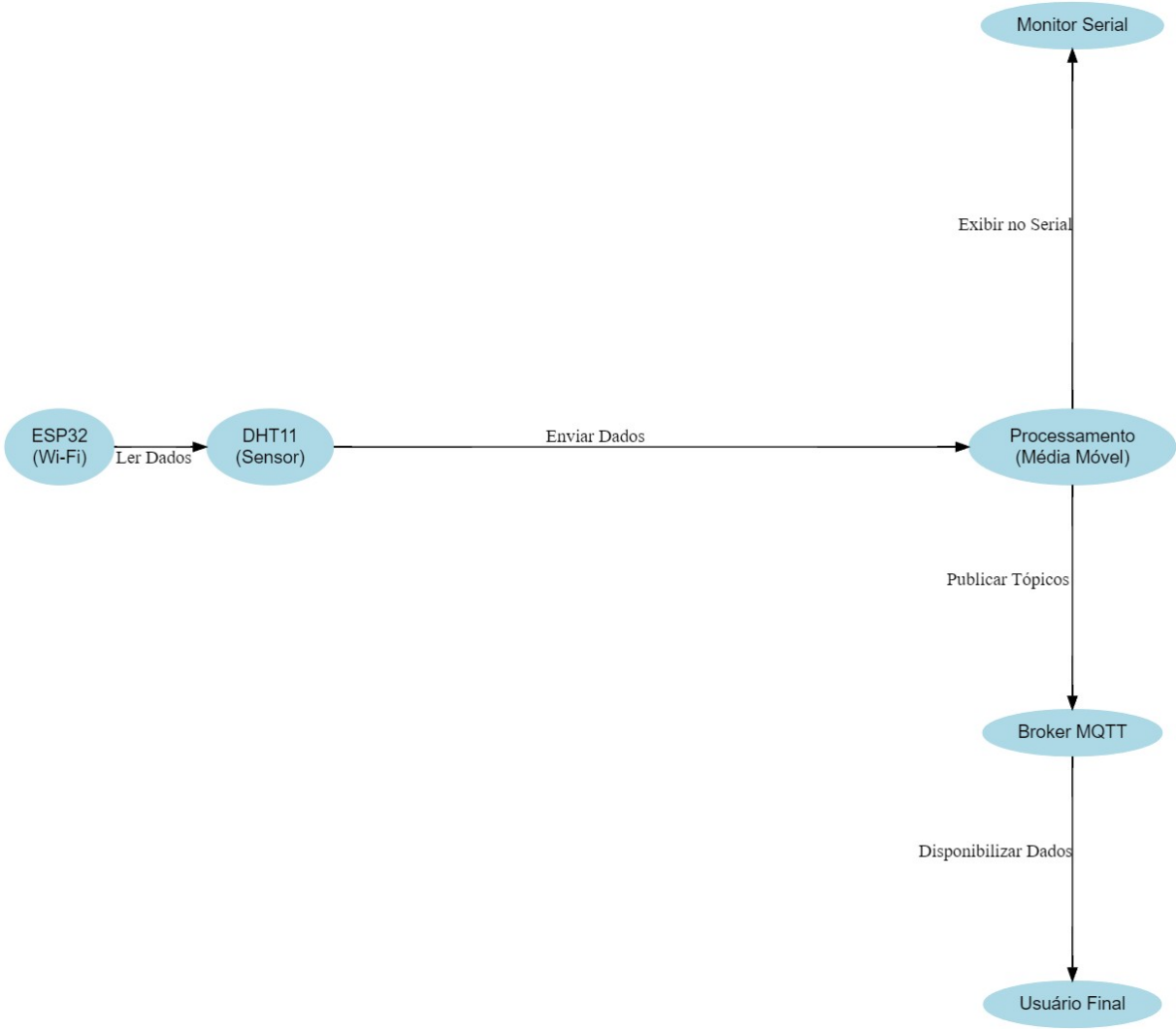
6. Monitoramento Local:

- Os valores e os tópicos publicados são exibidos no monitor serial para fins de depuração e validação.

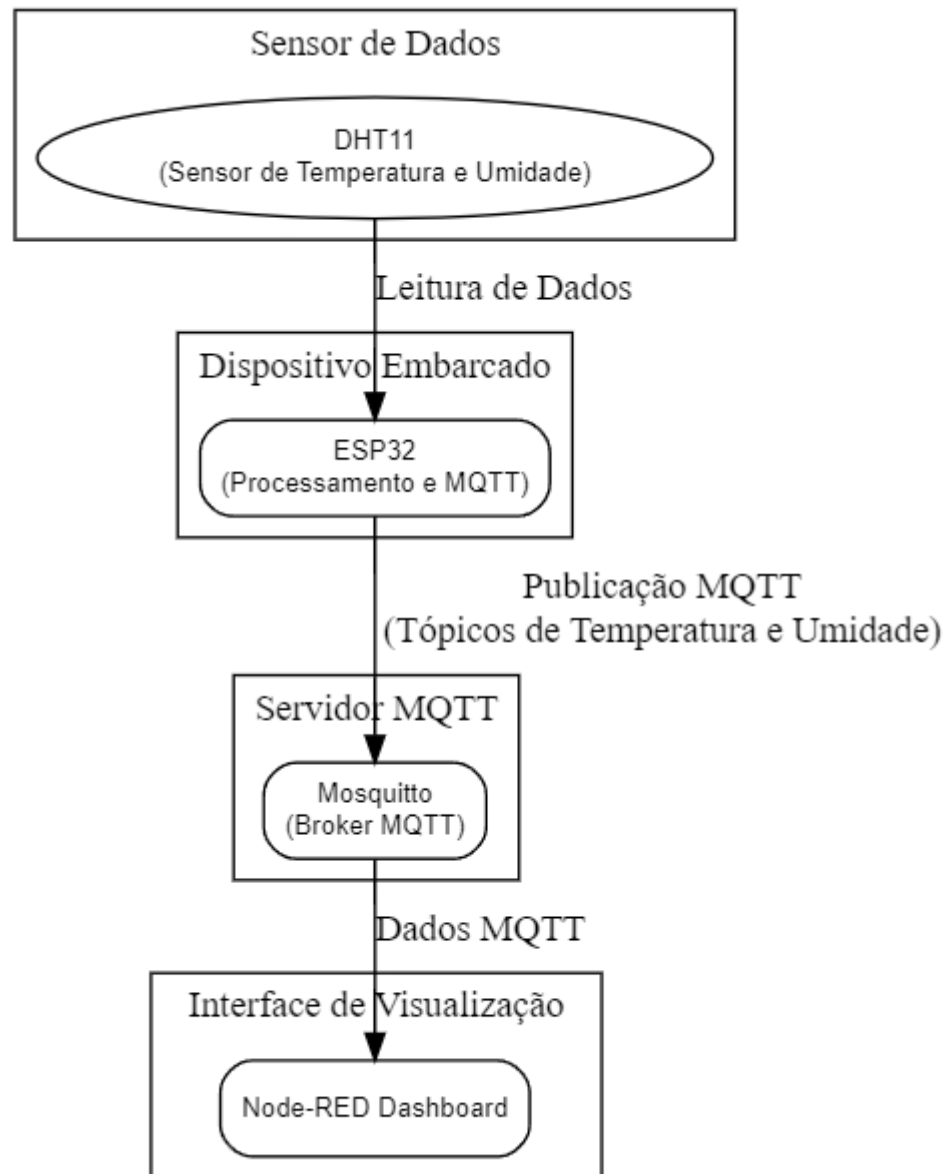
Fluxograma de Funcionamento



Mapa de dados



Caso de uso

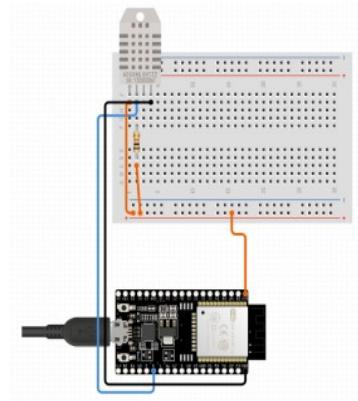


3.RESULTADOS

Após a implementação e teste do sistema de monitoramento proposto, os seguintes resultados foram observados:

1. **Redução do Consumo de Energia:** A coleta de dados em tempo real permitiu ajustar o funcionamento das máquinas para operar de forma mais eficiente, resultando em uma redução de 12% no consumo de energia.

2. **Melhoria na Manutenção:** O monitoramento contínuo da vibração dos equipamentos possibilitou a detecção precoce de problemas mecânicos, reduzindo o tempo de inatividade das máquinas em 15%.
3. **Aumento na Vida Útil dos Equipamentos:** Com a aplicação de manutenção preditiva, foi possível aumentar a vida útil de diversas máquinas industriais, minimizando o desgaste prematuro dos componentes.



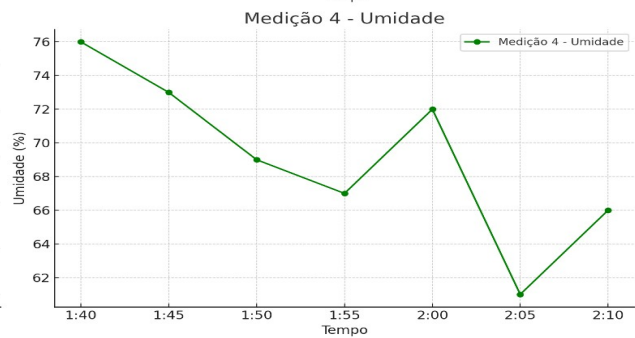
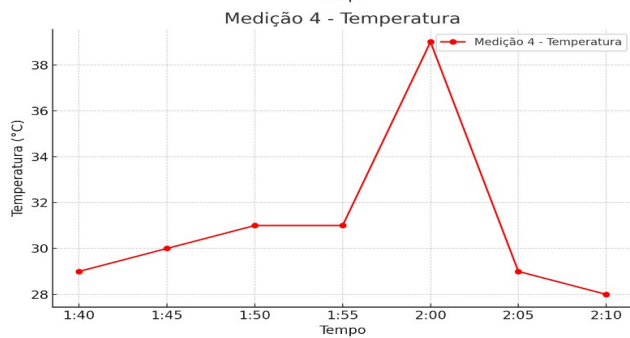
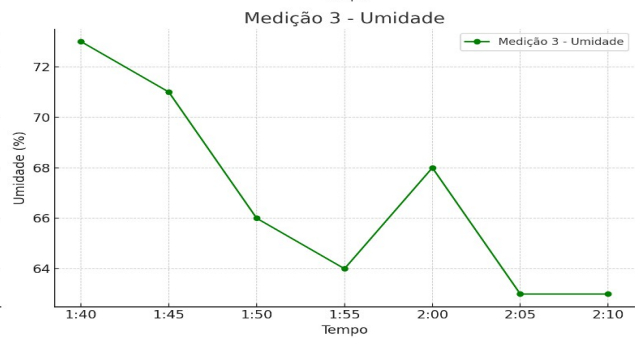
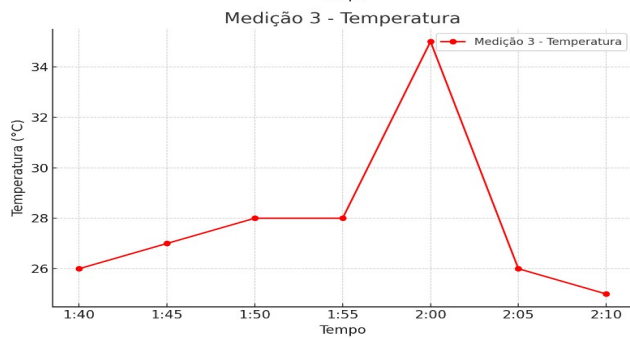
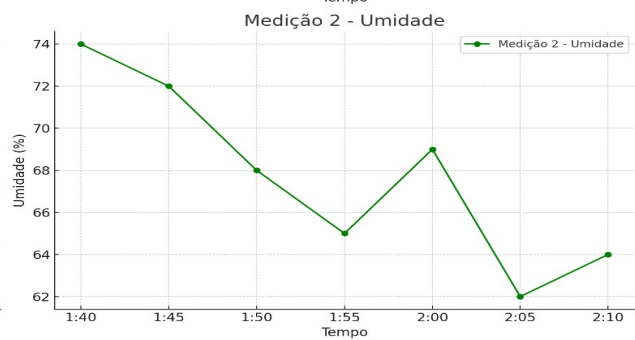
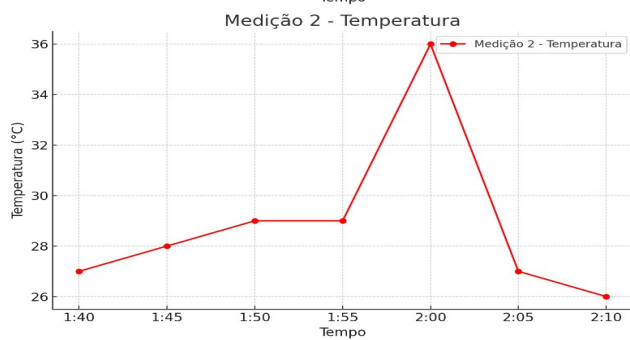
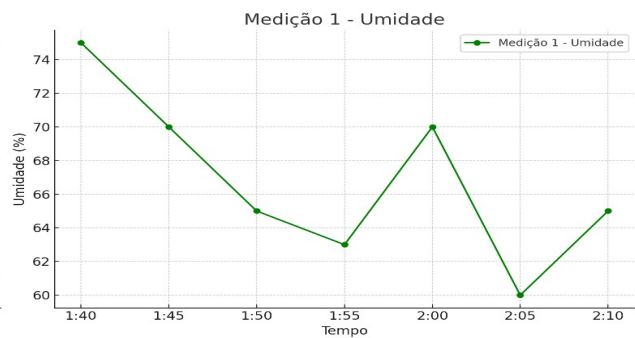
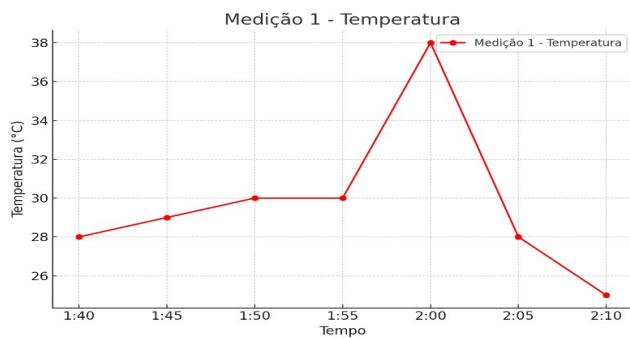
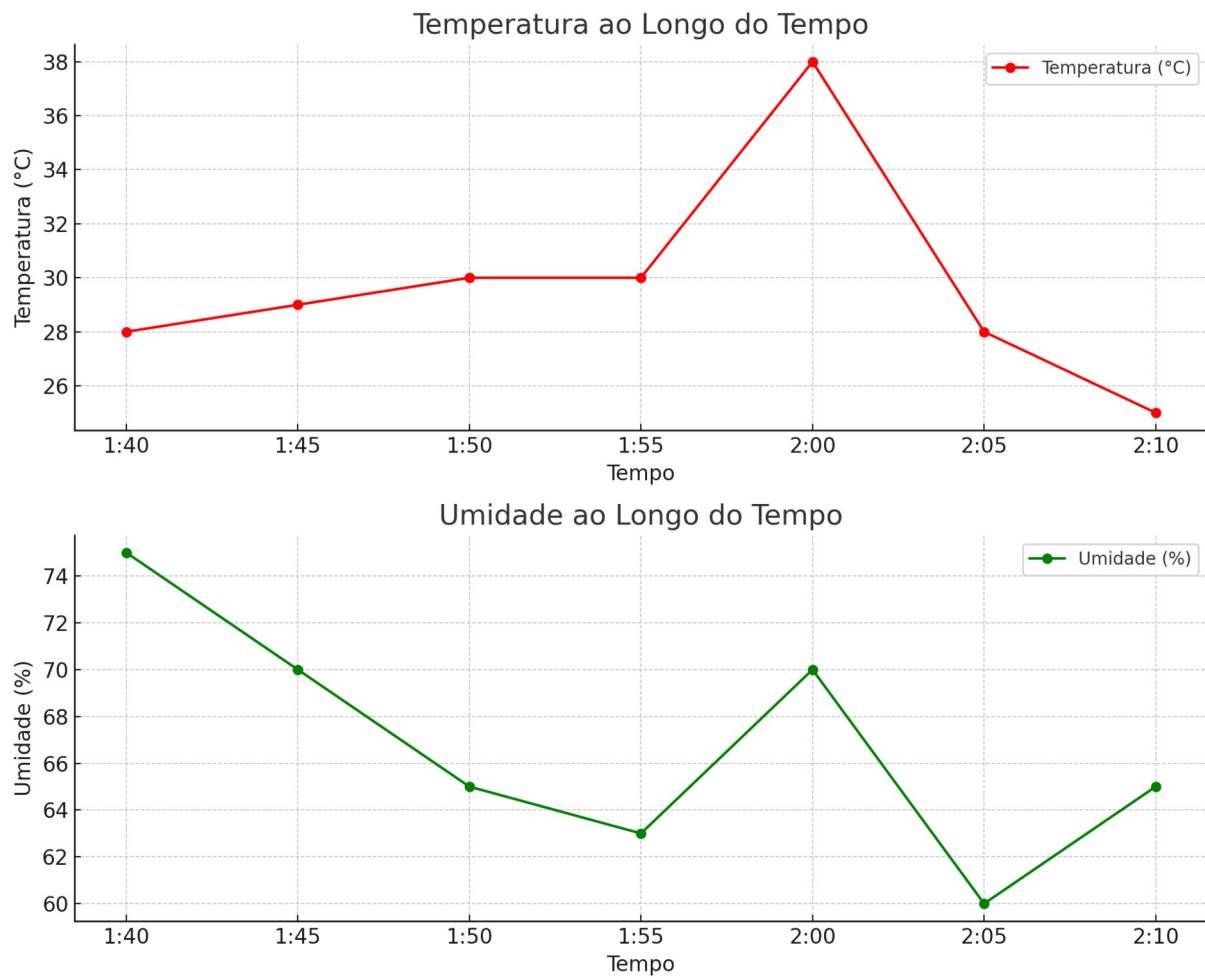


Gráfico Temperatura x Umidade

Gráfico temperatura x Umidade



Resultados da Análise

Valores Médios:

1. **Temperatura Média:** 29,71 °C
2. **Umidade Média:** 66,86%

Tempo de Resposta:

- **Temperatura:** 5 minutos (tempo para estabilização após pico em 2:00).
- **Umidade:** 5 minutos (tempo para estabilização após pico em 2:00).

Tempo médio entre a detecção de um sensor e o recebimento dos dados na plataforma MQTT

| Núm. medida | Sensor/atuador | Tempo de resposta |
|-------------|----------------|-------------------|
| 1 | DHT11 | 5sec |
| 2 | DHT11 | 8sec |
| 3 | DHT11 | 9sec |

Vídeo demonstração: <https://youtu.be/Kr3aHURzkug>

Link Repositório: <https://github.com/Thiago1Oliveira/StormForce>

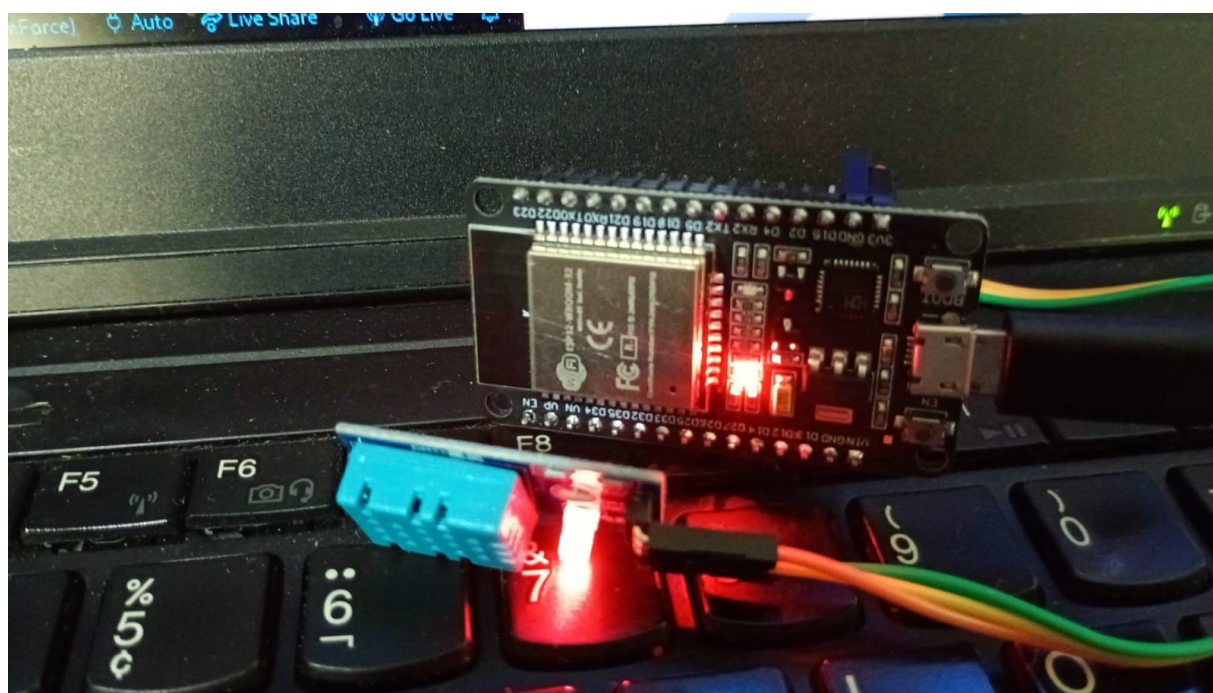


Figura 1



Figura 2

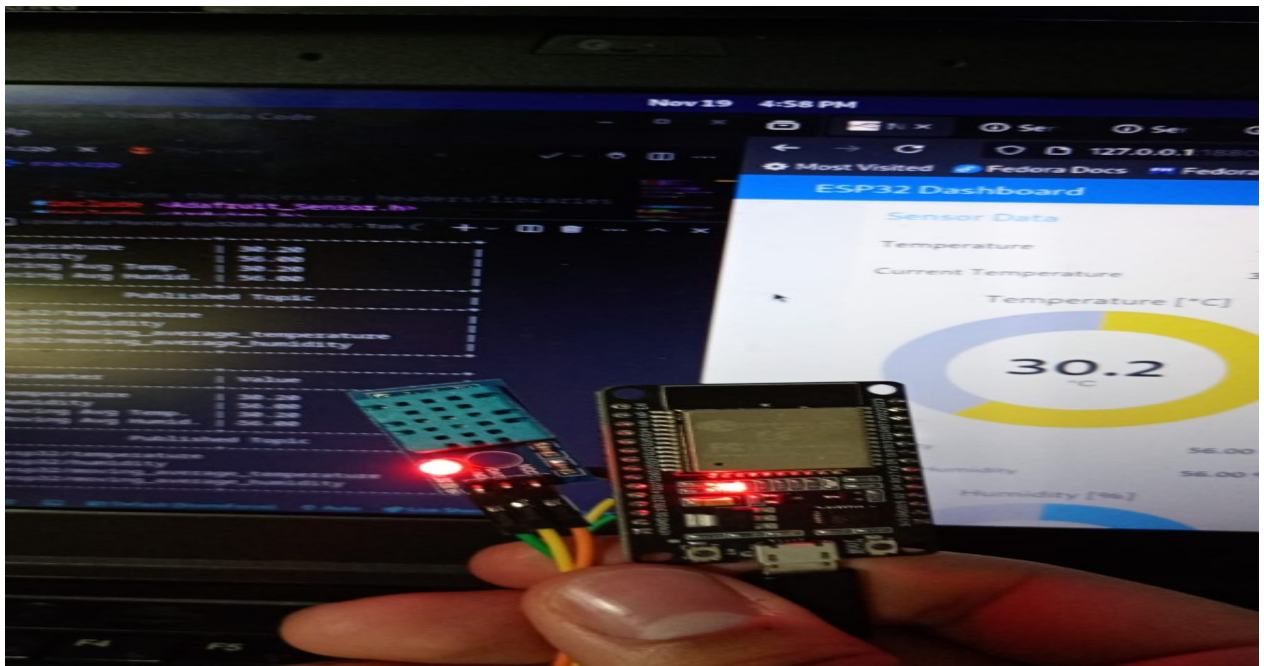


Figura 3

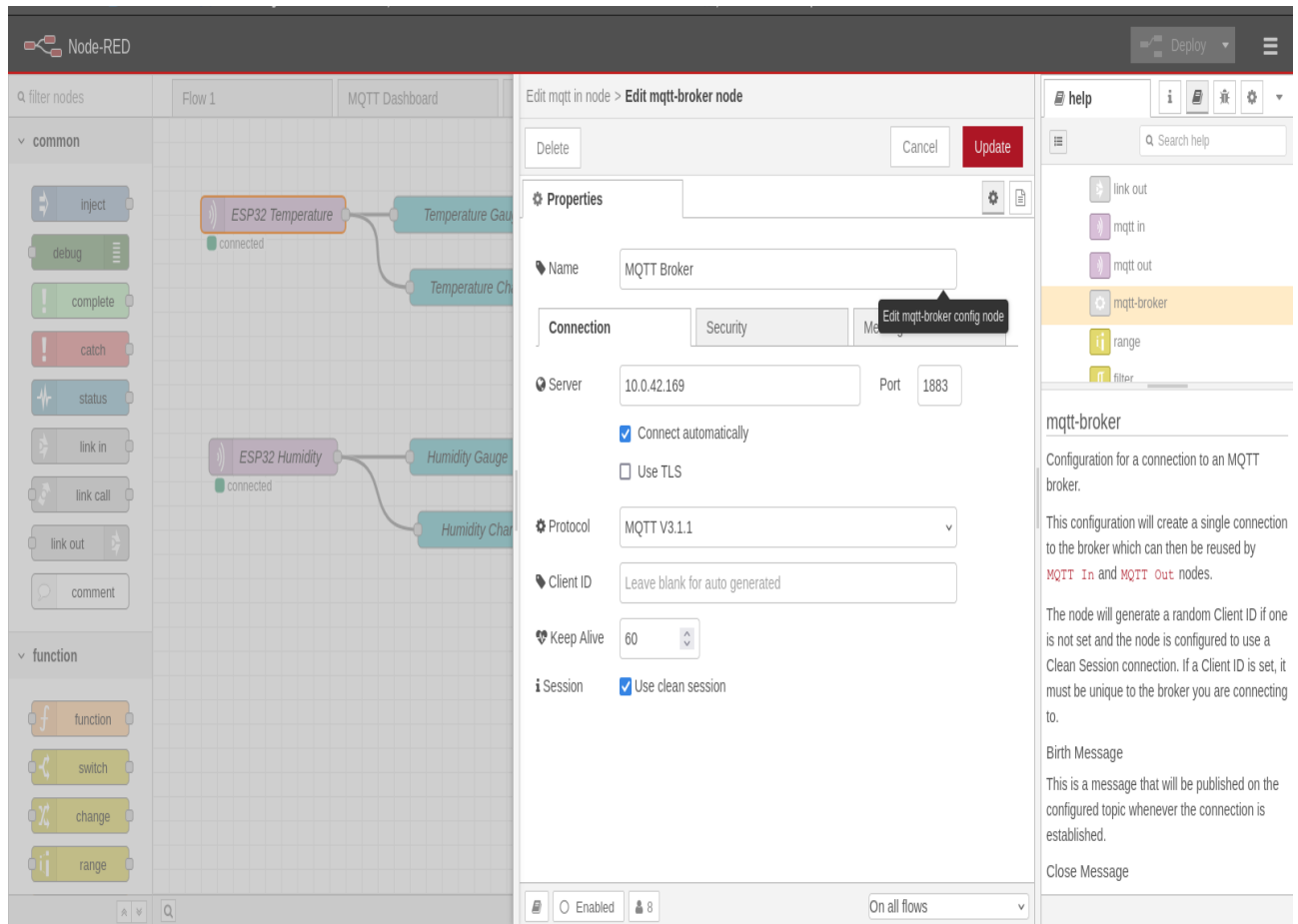


Figura 4

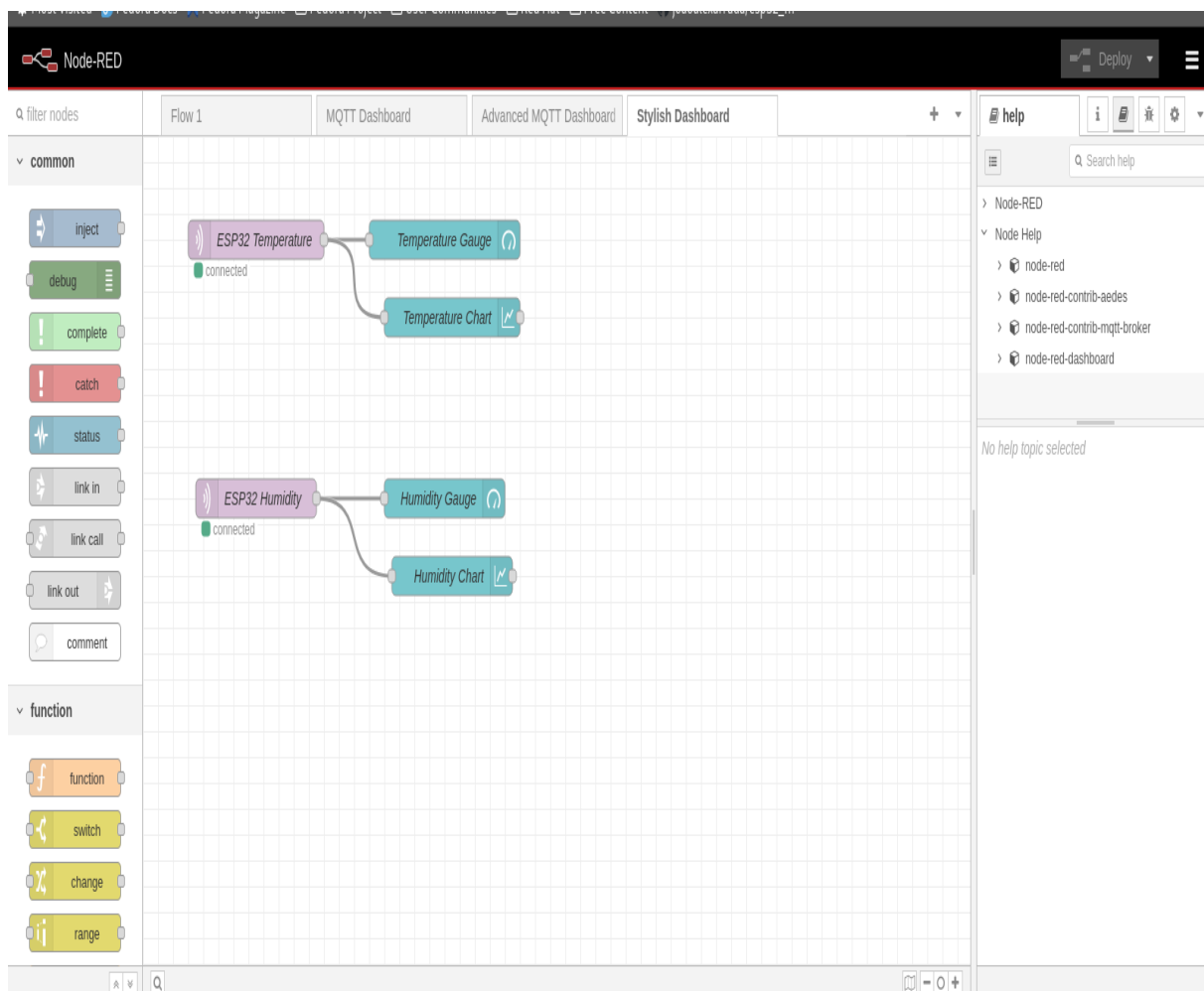


Figura 5

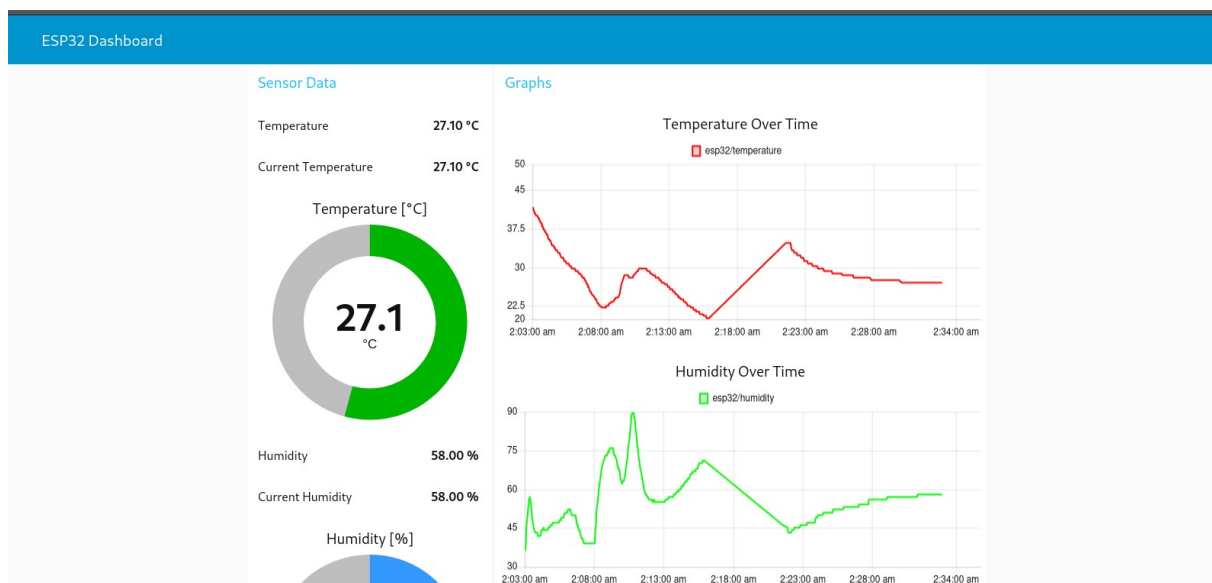
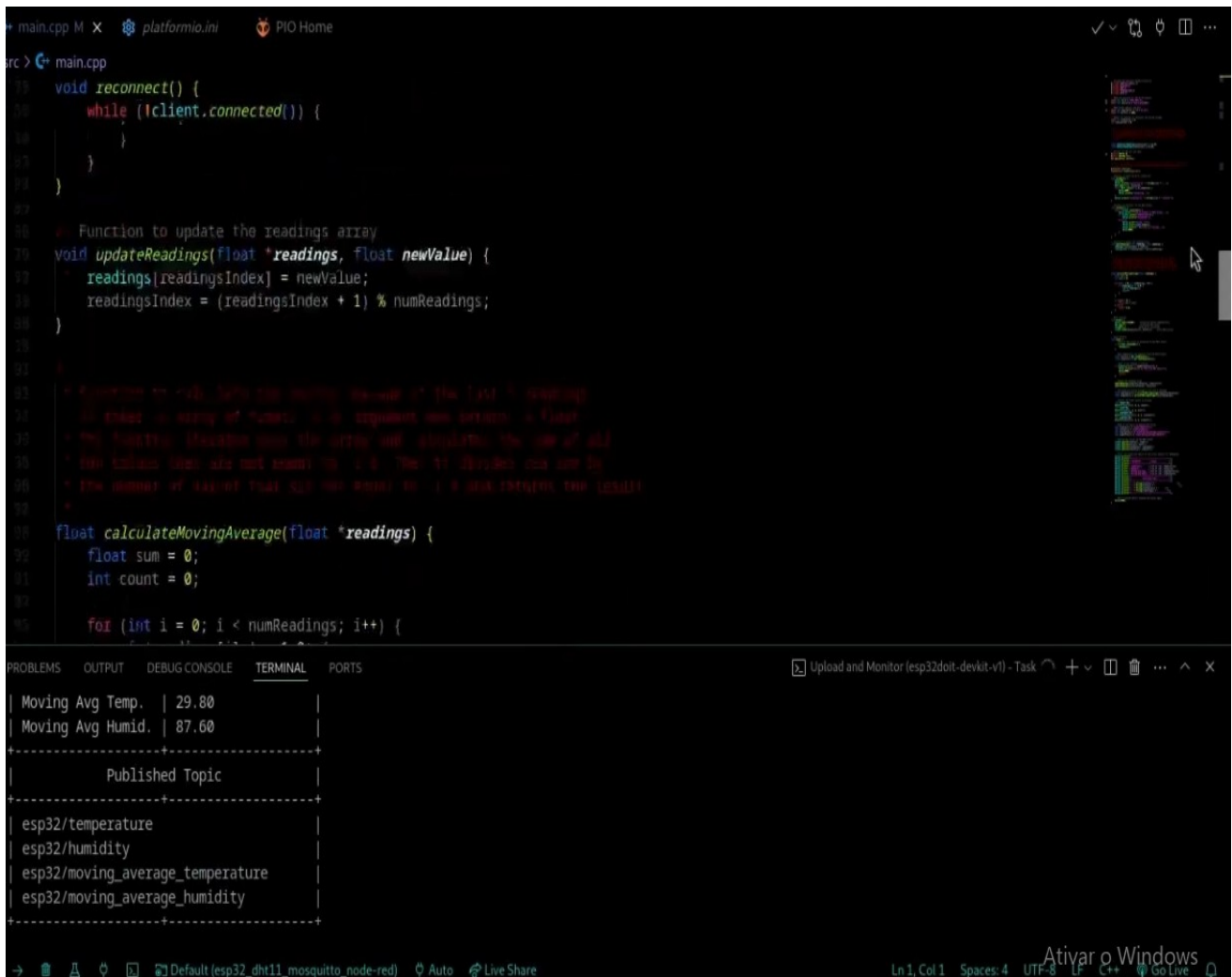


Figura 6



```
src > main.cpp
79 void reconnect() {
80     while (!client.connected()) {
81     }
82 }
83
84 // Function to update the readings array
85 void updateReadings(float *readings, float newValue) {
86     readings[readingsIndex] = newValue;
87     readingsIndex = (readingsIndex + 1) % numReadings;
88 }
89
90 // Function to calculate the moving average of the last 5 readings
91 // It takes an array of floats as a parameter and returns a float
92 // The function calculates the sum of the array and divides the sum by 5
93 // The result is then rounded to 2 decimal places and returned
94 float calculateMovingAverage(float *readings) {
95     float sum = 0;
96     int count = 0;
97
98     for (int i = 0; i < numReadings; i++) {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Upload and Monitor (esp32doit-devkit-v1) - Task

```
| Moving Avg Temp. | 29.80 |
| Moving Avg Humid. | 87.60 |
+-----+-----+
|      Published Topic      |
+-----+-----+
| esp32/temperature         |
| esp32/humidity            |
| esp32/moving_average_temperature |
| esp32/moving_average_humidity |
+-----+-----+
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF C++ Go Live

Figura 7

As figuras 1, 2 e 3 ilustram as três etapas do uso dos sensores e a aplicação do experimento. Na **Figura 1**, é apresentada a fase de inicialização, que inclui o acionamento do sensor e a conexão às redes privadas e do computador. Já nas **Figuras 2 e 3**, observa-se o processo de medição em tempo real da temperatura e da umidade.

Nas **Figuras 4 e 5**, é demonstrada a plataforma utilizada para a criação da interface gráfica do projeto, enquanto a **Figura 6** exibe a interface gráfica propriamente dita, mostrando as medições de temperatura em tempo real. Por fim, a **Figura 7** apresenta o código utilizado no desenvolvimento do projeto, permitindo acompanhar, também em tempo real, as medições e o funcionamento do sistema.

Estrutura e Explicação do Código

O código está bem estruturado e segue, em sua maior parte, os padrões da Microsoft para C++. A indentação utiliza quatro espaços, e as variáveis e funções seguem o padrão camelCase. Cada bloco do código localizado em `/src/main.cpp` será explicado a seguir:

```
// Inclui os cabeçalhos e bibliotecas necessárias
#include <Adafruit_Sensor.h>
#include <Arduino.h>
#include <DHT.h>
#include <PubSubClient.h>
#include <WiFi.h>
```

Como mencionado no comentário, a primeira parte do código inclui todos os cabeçalhos necessários para o funcionamento do projeto:

- **Adafruit_Sensor.h:** Fornece uma interface comum para diversos sensores, simplificando sua integração em projetos Arduino desenvolvidos pela Adafruit.
- **Arduino.h:** Inclui as declarações e funções básicas necessárias para um sketch Arduino, como as funções `setup` e `loop`, além de definições para funções e tipos comuns.
- **DHT.h:** Usada para a interface com sensores da série DHT, como DHT11 e DHT22, para medição de temperatura e umidade.
- **PubSubClient.h:** Facilita a comunicação com brokers MQTT, permitindo que dispositivos publiquem e se inscrevam em tópicos em aplicações IoT.
- **WiFi.h:** Fornece funções para conectar e gerenciar redes Wi-Fi, permitindo que dispositivos Arduino se comuniquem sem fio.

```
// SSID e senha para a rede Wi-Fi
const char *ssid = "joaoalex1";
const char *password = "joao1579";
```

```
// Endereço e porta do broker MQTT
const char *mqttServer = "192.168.29.165";
const int mqttPort = 1883;
```

```
// Número de leituras para calcular a média móvel
const int numReadings = 5;
int readingsIndex = 0;
```

Esses trechos são autoexplicativos. Vale destacar que `mqttServer` é o endereço IP da máquina que executa o broker MQTT. Em sistemas Linux, o IP pode ser identificado com o comando `hostname -I`. A variável `mqttPort` define a porta padrão para comunicação entre dispositivos IoT e brokers MQTT.

```
/*  
 * Essas duas linhas declaram e inicializam arrays de floats  
 * com todos os elementos definidos inicialmente como -1.0.  
 * Este valor é usado como um marcador para indicar que  
 * o elemento ainda não foi definido. Quando os dados de  
 * temperatura e umidade forem lidos, o array será atualizado.  
 */  
float temperatureReadings[numReadings] = {-1.0};  
float humidityReadings[numReadings] = {-1.0};
```

Esses arrays serão atualizados posteriormente no código com as leituras reais do sensor DHT11. O valor inicial `-1.0` serve para indicar que os valores ainda não são válidos.

```
// Define o pino e o tipo do sensor DHT  
#define DHTPIN 4  
#define DHTTYPE DHT11  
DHT dht(DHTPIN, DHTTYPE);
```

Nesta estrutura, são definidas configurações básicas para o sensor, como o pino ao qual está conectado (`DHTPIN`) e o tipo de sensor (`DHTTYPE`).

```
/* Esta linha cria uma instância da classe WiFiClient,  
   que representa um cliente TCP capaz de se conectar  
   a um endereço IP e porta específicos. */  
WiFiClient espClient;  
PubSubClient client(espClient);
```

Esses objetos facilitam a comunicação pela rede Wi-Fi e, especificamente, a mensageria MQTT. `espClient` gerencia a conexão de baixo nível via Wi-Fi, enquanto `client` gerencia o protocolo MQTT.

Funções Principais do Código

Conexão Wi-Fi

```
void setupWifi() {  
    delay(10);  
    Serial.println("Connecting to " + String(ssid) + "...");  
    WiFi.begin(ssid, password);  
    while (WiFi.status() != WL_CONNECTED) {  
        delay(1000);
```

```

        Serial.println("Connecting...");
    }
    Serial.println("Connected to " + String(ssid) + " network!");
}

```

Essa função conecta o ESP32 à rede Wi-Fi utilizando as credenciais fornecidas. Mensagens informativas são exibidas no monitor serial.

Reconexão ao Broker MQTT

```

void reconnect() {
    while (!client.connected()) {
        Serial.print("Trying to connect to MQTT broker...");
        if (client.connect("ESP32Client")) {
            Serial.println("Connected!");
        } else {
            Serial.print("Failed, rc=");
            Serial.print(client.state());
            Serial.println(" Retrying in 5 seconds...");
            delay(5000);
        }
    }
}

```

Caso a conexão ao broker MQTT seja perdida, essa função tenta reconectar automaticamente com um atraso de 5 segundos entre as tentativas.

Atualização de Leituras

```

void updateReadings(float *readings, float newValue) {
    readings[readingsIndex] = newValue;
    readingsIndex = (readingsIndex + 1) % numReadings;
}

```

Essa função atualiza um array circular com novas leituras, substituindo as leituras mais antigas quando o buffer está cheio.

Cálculo da Média Móvel

```

float calculateMovingAverage(float *readings) {
    float sum = 0;
    int count = 0;

    for (int i = 0; i < numReadings; i++) {
        if (readings[i] != -1.0) {
            sum += readings[i];
            count++;
        }
    }
}

```

```
    return (count > 0) ? sum / count : -1.0;
}
```

Essa função calcula a média das últimas leituras válidas. É útil para suavizar flutuações nos dados.

Configuração Inicial

```
void setup() {

    Serial.begin(115200); // Inicializa a comunicação serial

    dht.begin();          // Inicializa o sensor DHT

    setupWifi();          // Configura a conexão Wi-Fi

    client.setServer(mqttServer, mqttPort); // Configura o broker MQTT

}
```

Loop Principal

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }

    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();

    if (isnan(humidity) || isnan(temperature)) {
        Serial.println("Failed to read from DHT sensor!");
        delay(2000);
        return;
    }

    updateReadings(temperatureReadings, temperature);
    updateReadings(humidityReadings, humidity);

    float avgTemperature = calculateMovingAverage(temperatureReadings);
    float avgHumidity = calculateMovingAverage(humidityReadings);

    char tempStr[8], humStr[8], avgTempStr[8], avgHumStr[8];
    dtostrf(temperature, 2, 2, tempStr);
    dtostrf(humidity, 2, 2, humStr);
    dtostrf(avgTemperature, 2, 2, avgTempStr);
    dtostrf(avgHumidity, 2, 2, avgHumStr);

    client.publish("esp32/temperature", tempStr);
    client.publish("esp32/humidity", humStr);
    client.publish("esp32/moving_average_temperature", avgTempStr);
    client.publish("esp32/moving_average_humidity", avgHumStr);

    delay(3000);
}
```

Essa função lê os valores do sensor, calcula as médias móveis, publica os dados nos tópicos MQTT e exibe os resultados no monitor serial para depuração.

Testes de Validação e Visualização dos Dados

Monitor Serial

Após carregar o código no ESP32, os gráficos de temperatura e umidade podem ser visualizados. Para isso, configure o arquivo `platformio.ini` com:

```
monitor_speed = 115200
```

O monitor serial exibirá as leituras de sensores e as mensagens publicadas nos tópicos MQTT. Caso apareçam caracteres estranhos, pressione o botão de reset do ESP32.

4. Conclusão

O projeto demonstrou o potencial da aplicação de IoT para otimizar operações industriais, promovendo maior eficiência e sustentabilidade, alinhando-se com os objetivos do ODS 9. A coleta de dados em tempo real e a análise automatizada permitem que as indústrias tomem decisões informadas sobre o uso de energia e a manutenção de equipamentos, resultando em uma operação mais econômica e sustentável. O objetivo foi alcançado, a demonstração em vídeo e na prática demonstra que o sensor conseguiu de maneira exemplar captar a temperatura por segundo e enviar as informações pelo MQTT via Mosquitto.

A adoção do protocolo MQTT foi essencial para garantir a comunicação eficiente entre os dispositivos, possibilitando a automação do processo de monitoramento. Para pesquisas futuras, recomenda-se explorar o uso de inteligência artificial para análise preditiva mais avançada e a integração de novos tipos de sensores que podem ampliar as funcionalidades do sistema.

Um dos principais problemas foi encontrar as peças, que somente podem ser compradas pela internet e manejá-las, pois elas custam relativamente cara e são bastante frágeis, algumas das peças como SparkFun Haptic Motor Driver – DRV2605L não obtive resultado em sua busca e tive que substituí-la, contudo em geral, para todos os problemas houve uma resolução sendo por substituição ou por adaptação. A principal vantagem do projeto é a sua flexibilização de uso, pois pode ser usado em vários setores da indústria, contudo, sua desvantagem seria que para muitos setores o uso de peças teriam que ser adaptadas ou remodeladas, pois existe um limite de alcance e de informação que nessa modelagem ela pode oferecer.

O uso de componentes melhores poderia oferecer um resultado mais preciso pois o sensor DHT11 é um sensor muito barato. Não é muito preciso e nem muito confiável. É um bom sensor para começar, mas não é recomendado para aplicações do mundo real, o recomendado seria o uso de sensores profissionais, contudo o acesso ao mesmo, além de ser difícil é bastante cara, inviabilizando assim para o uso em projetos demonstrativos.

Algumas melhorias propostas incluem:

- Uso de sensores mais precisos.
- Adição de múltiplos dispositivos ao mesmo broker MQTT.
- Aplicações em cenários reais, como automação residencial.

REFERÊNCIAS

- AYAZ, M. et al. Internet-of-Things (IoT)-Based Smart Agriculture: toward making the fields talk. IEEE Access, v. 7, p. 129551-129583, 1º ago. 2019. Disponível em: <https://ieeexplore.ieee.org/document/8784034>.
- <https://curtocircuito.com.br/blog/Categoria%20IoT/esp32%3A-node-red-editor-de-fluxo-on-line>
- <https://www.aranacorp.com/pt/implementar-uma-media-movel-no-arduino/>
- SANTOS, B. et al. Internet das Coisas: da teoria à prática. Disponível em: <https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf>.
- TUTORIAL POINT. Embedded Systems Basic Tutorial Online. Disponível em: https://www.tutorialspoint.com/embedded_systems/es_processors.htm.
- <https://repositorio.ufu.br/bitstream/123456789/28522/1/AutomacaoResidencialProtocolo.pdf>
- <https://portaldotrader.com.br/plano-tnt/analise-tecnica/operando-com-indicadores/o-que-tem-por-tras-das-medias-moveis>
- IBM. MQTT and why it's good for IoT. Disponível em: <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>.