FLÁVIO DE MATOS MENEZES

INTERNET DAS COISAS APLICADA A DOMÓTICA

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 - Projeto de Engenharia II e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 29 de janeiro de 2025.

COMISSÃO EXAMINADORA

Prof. Dr. André Gomes Torres - Orientador
Universidade Federal de Viçosa

Prof. Dr. Denílson Eduardo Rodrigues - Membro
Universidade Federal de Viçosa

Prof. Dr. Mauro de Oliveira Prates - Membro
Universidade Federal de Viçosa

INTERNET DAS COISAS APLICADA À DOMÓTICA

FLÁVIO DE M. MENEZES[1]

[1] *Graduação em Engenharia Elétrica - Universidade Federal de Viçosa – Av. Peter Henry Rolfs s/n, Campus UFV, CEP: 36570-000, Viçosa, MG*

E-MAIL: flavio.menezes@ufv.br

Abstract— The Internet of Things (IoT) represents a transformative paradigm in the digital age, connecting a vast array of physical devices, vehicles, home appliances, and other objects embedded with sensors, software, and network connectivity. This interconnected network facilitates the collection and exchange of data, enabling unprecedented levels of automation and intelligence. Residential Automation, or Domotics, is one of this technology's many applications. This paper presents the design and implementation of a home automation prototype that makes use of a minuature house equipped with four leds, a fan and a temperature sensor. Cloud services and SCADA are provided by Thingable! and server side programming is done via Node-RED. The hardware interface between the loads and the server is provided by an ESP32 microcontroller and uses the MQTT Protocol for communication.

Keywords— Internet of Things, Domotics, Node-RED, MQTT Protocol.

Resumo— A Internet das Coisas (IoT) representa um paradigma transformador na era digital, conectando uma vasta gama de dispositivos físicos, veículos, eletrodomésticos e outros objetos incorporados com sensores, software e conectividade de rede. Essa rede interconectada facilita a coleta e a troca de dados, permitindo níveis sem precedentes de automação e inteligência. A Automação Residencial, ou Domótica, é uma das muitas aplicações dessa tecnologia. Este artigo apresenta o design e a implementação de um protótipo de automação residencial que faz uso de uma casa em miniatura equipada com quatro leds, um ventilador e um sensor de temperatura. Os serviços de nuvem e SCADA são fornecidos pela Thingable! e a programação do lado do servidor é feita via Node-RED. A interface de hardware entre o servidor e as cargas é fornecida por um microcontrolador ESP32 e usa o protocolo MQTT para comunicação.

Palavras-chave— Internet das Coisas, Domótica, Node-RED, Protocolo MQTT.

1 Introdução

A Internet das Coisas (*IdC*), do inglês Internet of Things (*IoT*), consiste numa rede de objetos físicos como sensores e atuadores, software e outras tecnologias com conexão à Internet para troca de dados. Embora não haja uma definição única para o que constitui a *IoT*, a invenção do termo é atribuída a Kevin Ashton, Diretor Executivo do laboratório Auto-ID do Instituto Tecnológico de Massachusetts (MIT) em 1999. Embora o termo tenha ganhado popularização em meados de 2003, o primeiro utensílio conectado à Internet foi uma máquina de refrigerantes localizada no campus da Universidade Carnegie Melon no início dos anos 80 [1]. Os programadores que trabalhavam no prédio onde a máquina estava instalada podiam verificar o status de bebidas na máquina a qualquer momento, que comunicava a informação pela Internet.

1.1 Panorama

Destacam-se abaixo alguns eventos significativos na história da *IoT*:

1980 – Primeiro dispositivo conectado a Internet: Máquina de refrigerantes no campus da Universidade Carnegie Melon.

1990 – Romkey e Hackett conectam uma torradeira à internet, permitindo ligá-la e desligá-la a distância. O projeto foi apresentado na conferência *Interop '89* [2].

1999 - O Laboratório Auto-ID do MIT é fundado e cunha o termo Internet das Coisas (*IoT*), lançando grande parte das bases para a padronização da tecnologia RFID (Radio Frequency Identification) e a introdução do EPC (Electronic Product Code) [3].

2000 - LG lançou a primeira geladeira conectada a internet, a Internet Digital DIOS. Conhecida como geladeira inteligente (Smart refrigerator), o aparelho foi programado para detectar que tipos de produtos estão armazenado e manter um rastreamento do estoque por meio de código de barras ou leitura *RFID* [4].

2002 – A empresa Ambient Devices, spin-off do MIT Media Lab criada por David Rose lança o Ambient Orb, nomeado como uma das Idéias do Ano pelo jornal New York times [1].

2003-2004 - *RFID* é implantado em grande escala pelo Departamento de Defesa dos EUA em seu programa Savi e pelo Wal-Mart [1].

2005 - A União Internacional de Telecomunicações (UIT) da ONU publicou seu primeiro relatório sobre o tópico Internet das Coisas [1].

2008 - Reconhecimento pela União Européia e a Primeira Conferência Europeia de *IoT* é realizada [1].

2008 - Um grupo de empresas lançou a IPSO Alliance para

promover o uso de IP em redes de “Objetos Inteligentes” e habilitar a Internet das Coisas [1].

2008 - A FCC (Comissão Federal de Comunicações dos EUA) aprova a abertura do espaço branco nos EUA [1].

2008 - O Conselho Nacional de Inteligência dos EUA listou a *IoT* como uma das 6 “Tecnologias Civis Disruptivas” com impactos potenciais nos interesses dos EUA até 2025 [1].

2010 - O premiê chinês Wen Jiabao chama a *IoT* de uma indústria-chave para a China e tem planos de fazer grandes investimentos na área [1].

2011 - Lançamento público do IPv6 - O novo protocolo permite 2^{128} endereços [1].

2015 - Lançamento do Amazon Echo, dando início à revolução do alto-falante inteligente - O controle de voz se torna uma interface essencial para dispositivos *IoT* [5].

2017 - Ascensão de aplicações industriais de *IoT* em manufatura e logística [6].

2020 - Previsão de que gastos globais em *IoT* ultrapassem US$ 1 trilhão em 2024 [7].

2020 - Adoção generalizada de *IoT* em assistência médica para monitoramento remoto de pacientes. Durante a pandemia de COVID-19, [8] relata a utilização de sistema de monitoramento inteligente de saúde que emprega um aparelho conectado a internet capaz de monitorar parâmetros como ritmo cardíaco, temperatura corporal e saturação de oxigênio no sangue e transmitir as medições em tempo real, gerando um alerta para o profissional de saúde em caso de anomalia.

2022 - Lançamento do protocolo Matter para interoperabilidade de dispositivos domésticos inteligentes - Padroniza a comunicação entre diferentes marcas de casas inteligentes [9].

2023 - Avanços com Inteligência artificial (IA) integrada à *IoT* para análises avançadas - Permite insights mais profundos de dados de sensores para melhor tomada de decisão [10].

2024 – Estima-se que haja 17,08 bilhões de dispositivos de *IoT* em todo o mundo [11].

2 Objetivo

Automatizar iluminação e climatização de uma casa em miniatura através de tecnologias *IoT*, e demonstrar a viabilidade da construção de um sistema de automação residencial utilizando tecnologias *IoT*.

2.1 Objetivos Específicos

- Determinar tipos de sensores a serem utilizados.
- Determinar elementos de controle.
- Criar um programa para o microcontrolador *ESP32* que o torne capaz de trocar informações com a plataforma online de *IoT Thingable!*, fazendo uma ponte entre os dispositivos físicos instalados na maquete e o serviço de *IoT*.
- Criar um script de processamento dos dados trocados com o microcontrolador para controle dos dispositivos através do software *Node-RED* embutido na plataforma *Thingable!*.
- Testar o funcionamento do sistema como um todo.

3 Revisão Bibliográfica

O coração do sistema é o microcontrolador *ESP32*, ideal para aplicações *IoT* devido à sua integração wireless. Para a comunicação entre os dispositivos, o protocolo *MQTT* é utilizado; ele é leve e eficiente para publicar e subscrever mensagens em tópicos. A plataforma *Thingable!* serve como um ambiente de desenvolvimento e gerenciamento de soluções *IoT*, oferecendo uma interface intuitiva e ferramentas para conectar diversos dispositivos e visualizar dados em tempo real. A plataforma utiliza o *Node-RED*, uma ferramenta de programação visual que permite criar fluxos de trabalho para automatizar tarefas e integrar diferentes serviços.

3.1 Microcontrolador ESP32

O *ESP32* é um microcontrolador de baixo consumo de 32 bits com 34 GPIOs programáveis e Wi-Fi e Bluetooth integrados, que o tornam uma excelente escolha para aplicações de *IoT*. Criado pela Espressif Systems, seu núcleo contem dois processadores Tensilica Xtensa LX6 (504.85 CoreMark por *core* a 240 MHz). Ele possui 448 KB de ROM e 520 KB de SRAM [12].

3.2 Protocolo MQTT

Conforme o site oficial do protocolo *MQTT* [13], *MQTT* (Message Queuing Telemetry Transport, ou Transporte de Telemetria de Enfileiramento de Mensagens) é um protocolo de mensagens padrão do Comitê Técnico OASIS para a Internet das Coisas (*IoT*). Ele é projetado como um transporte de mensagens de publicação/inscrição (publish/subscribe) extremamente leve, ideal para conectar dispositivos remotos com pouco código e largura de banda de rede mínima. O *MQTT* hoje é usado em uma ampla variedade de indústrias, como automotiva, manufatura, telecomunicações, petróleo e gás, etc.

O *MQTT* é um protocolo push (método de comunicação de dados onde o servidor envia informações para o cliente de forma proativa, sem que o cliente precise solicitar explicitamente esses dados. É o oposto do modelo tradicional de solicitação-resposta, onde o cliente sempre inicia a comunicação)

lançado em pela IBM em 1999 [14]. No *MQTT*, os dispositivos se inscrevem em “tópicos” para publicar e receber mensagens (*payload*). Todos os dispositivos inscritos em um tópico recebem as mensagens enviadas ao mesmo.

Conforme descreve o documento oficial de especificação do protocolo *MQTT* [15], ele é executado sobre TCP/IP ou sobre outros protocolos de rede que fornecem conexões ordenadas, sem perdas e bidirecionais. Seus recursos incluem:

- Uso do padrão de mensagem publicar/inscrever que fornece distribuição de mensagens um para muitos e desacoplamento de aplicativos.
- Um transporte de mensagens que é agnóstico ao conteúdo da carga útil.
- ·Três qualidades de serviço para entrega de mensagens:
    1. "No máximo uma vez", (*QoS0*) onde as mensagens são entregues de acordo com os melhores esforços do ambiente operacional. Pode ocorrer perda de mensagens. Este nível pode ser usado, por exemplo, com dados de sensores ambientais onde não importa se uma leitura individual é perdida, pois a próxima será publicada logo depois.
    2. "Pelo menos uma vez", (*QoS1*) onde as mensagens têm garantia de chegar, mas podem ocorrer duplicatas.
    3. "Exatamente uma vez", (*QoS2*) onde as mensagens têm garantia de chegar exatamente uma vez. Este nível pode ser usado, por exemplo, com sistemas de cobrança onde mensagens duplicadas ou perdidas podem levar à aplicação de cobranças incorretas.
- Um pequeno overhead de transporte e transferências de protocolo minimizadas para reduzir o tráfego de rede.
- Um mecanismo para notificar as partes interessadas quando ocorre uma desconexão anormal.

O *MQTT* possui 15 pacotes de controle, à saber [15]:

- *CONNECT*: Conectar - Inicia uma conexão de rede com um broker *MQTT*.
- *CONNACK*: Confirmação de Conexão - Confirma o pedido de conexão e indica sucesso ou falha.
- *PUBLISH*: Publicar - Envia uma mensagem de aplicação (payload) para um tópico específico.
- *PUBACK*: Confirmação de Publicação - Confirma o recebimento de um pacote PUBLISH.
- *PUBREC*: Recebimento de Publicação - Primeira parte de uma publicação QoS 2.
- *PUBREL*: Liberação de Publicação - Segunda parte de uma publicação QoS 2.
- *PUBCOMP*: Compleção de Publicação - Terceira parte de uma publicação QoS 2.
- *SUBSCRIBE*: Inscrever – Inscreve em um ou mais tópicos.
- *SUBACK*: Confirmação de Inscrição - Confirma o pedido de inscrição e retorna os níveis de *QoS* concedidos.
- *UNSUBSCRIBE*: Cancelar Inscrição - Cancela a inscrição de um ou mais tópicos.
- *UNSUBACK*: Confirmação de Cancelamento de Inscrição - Confirma o pedido de cancelamento de inscrição.
- *PINGREQ*: Solicitação de Ping - Mantém a conexão ativa.
- *PINGRESP*: Resposta de Ping - Confirma o pacote *PINGREQ*.
- *DISCONNECT*: Desconectar - Indica que o cliente está se desconectando do servidor.
- *AUTH*: Autenticação - Utilizado para autenticação.

A Figura 1 exibe a estrutura do pacote. A região em vermelho (dois primeiros bytes) é o cabeçalho fixo. Os quatro primeiros bits do primeiro byte informam o tipo do pacote, e a região em azul contém a carga útil de informação (*payload*).

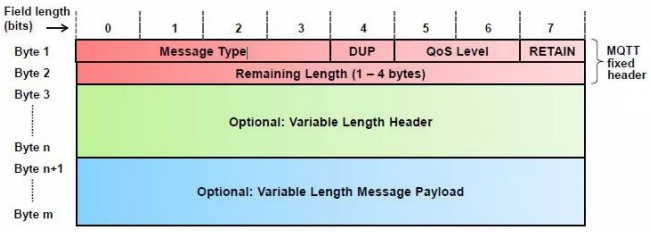

Figura 1. Estrutura de um pacote *MQTT* [16]

A Figura 2 ilustra a comunicação que o protocolo proporciona entre diversos dispositivos. Na figura, o sensor de umidade à esquerda publica suas leituras no tópico de umidade. Os smartphones, que estão inscritos neste tópico recebem o valor medido. Um dos smartphones então publica, no tópico de comando do irrgador, um comando para acionar o irrigador, que está inscrito neste tópico.

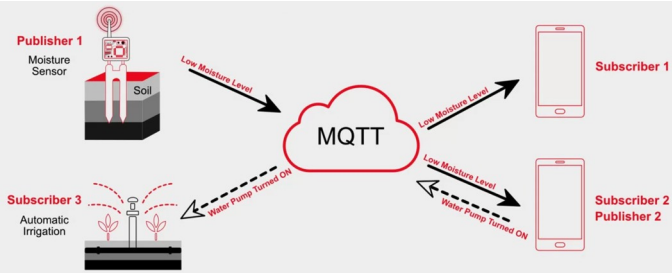

Figura 2. Comunicação entre diversos dispositivos através do *MQTT* [17].

3.3 *Plataforma Thingable!*

A *Thingable!* é uma plataforma avançada de *IoT* e telemetria que facilita o desenvolvimento de aplicações para monitoramento remoto e gestão inteligente de dados. Utilizando uma abordagem low-code, a *Thingable!* permite que empresas criem soluções de *IoT* de forma ágil, prática e intuitiva, sem a necessidade de programação complexa [18].

A plataforma integra dados de diferentes ativos em um único ambiente, oferecendo ferramentas para conectar, gerenciar e interagir com dispositivos *IoT*, como sensores e concentradores de dados. Isso possibilita a coleta, análise e visualização de informações em tempo real, promovendo uma gestão mais eficiente e produtiva [18]. Ela fornece o servidor/*broker MQTT* utilizado neste trabalho.

A Figura 3 apresenta a interface de abertura da plataforma, onde é possível cadastrar as credenciais do dispositivo *IoT* (neste caso, o *ESP32*). A opção Logic Builder dá acesso à ferramenta do *Node-RED*, que será usada para programar o comportamento do servidor em relação aos dados recebidos e enviados ao *ESP32*.

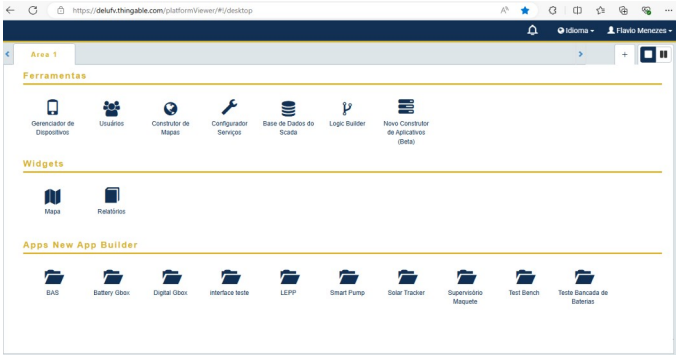

Figura 3. Interface da *Thingable!* [16]

3.4 *Node-RED*

Conforme [19], o *Node-RED* é uma ferramenta de programação para conectar dispositivos de hardware, APIs e serviços online. Ele fornece um editor baseado em navegador com um estilo bastante visual que permite programar através de fluxogramas, por meio de nós (blocos de funções) interligados. O *Node-RED* conta, ainda, com uma paleta de nós criados pela comunidade de usuários, que podem ser obtidos e acionados em tempo real. A linguagem de programação utilizada pelo *Node-RED* é o *Javascript*.

O *Node-RED* possui, entre suas inúmeras funcionalidades, nós prontos para a comunicação *MQTT* com dispositivos *IoT*.

A Figura 4 exibe a tela de programação do *Node-RED* embutida no ambiente da *Thingable!*

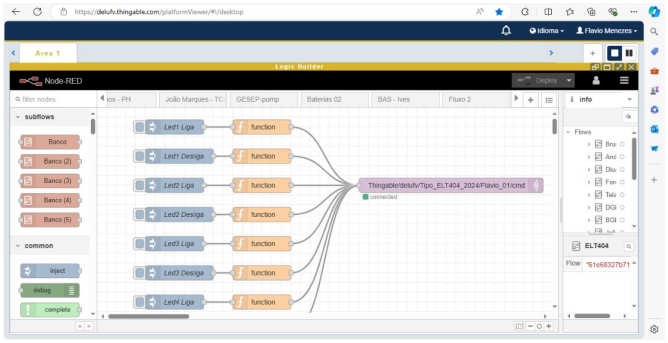

Figura 4. *Node-RED*

4 Materiais e Métodos

4.1 *Cadastro na Thingable!*

Inicialmente, foi criada uma conta de usuário na plataforma *Thingable!* associada ao Departamento de Engenharia Elétrica da Universidade Federal de Viçosa (DEL-UFV). Logo em seguida foram cadastradas as credenciais do dispositivo *IoT* utilizado (*ESP32*) conforme detalhado na Figura 5.

Credenciais de Dispositivo

Número de série	Flavio_01
Tipo de dispositivo	Tipo_ELT404_2024
Class	device
URL	mqtt.thingable.com
Cliente-ID	Thingable:delufv:Tipo_ELT404_2024:Flavio_01
Usuário	Thingable:delufv:Tipo_ELT404_2024:Flavio_01
Senha (Cliente)	
Tópico	Thingable/delufv/Tipo_ELT404_2024/Flavio_01/evt

Figura 5. Credenciais do dispositivo

4.2 *Programação do ESP32*

Em seguida, iniciou-se a criação do código para o *ESP32*. A escrita foi feita na IDE do Arduíno em linguagem C. Cabe ressaltar que o microcontrolador se encarrega apenas de enviar eventos (leituras de dispositivos de entrada) e receber comandos e executá-los (acionar dispositivos de saída). O processa-

mento de informação propriamente dito ocorre no ambiente *Node-RED* do servidor.

A seguir se encontram os trechos mais importantes do código do microcontrolador. A Figura 6 apresenta as bibliotecas utilizadas, como a *PubSubClient.h*, que contém o protocolo *MQTT* e a *ArduinoJson.h*, que interpreta mensagens em formato *JSON*.

```
#include <Arduino.h>
#ifdef ESP8266
       #include <ESP8266WiFi.h>
#endif
#ifdef ESP32
       #include <WiFi.h>
#endif

#include <PubSubClient.h>
#include <ArduinoJson.h>
```

Figura 6. Bibliotecas usadas pelo *ESP32*

A Figura 7 exibe as credenciais do dispositivo foram inseridas no código, na forma de variáveis do tipo *string*.

```
const char* mqttUser = "Thingable:delufv:Tipo_ELT404_2024:Flavio_01";
const char* mqttPassword = "[redacted]";
```

Figura 7. Credenciais no *ESP32*

Se encontra na Figura 8 a função que configura a comunicação *MQTT*. O servidor da *Thingable!* é definido, bem como a porta, 1883. A função callback (quando o servidor envia um comando ao dispositivo) também é definida aqui.

```
// MQTT client
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);
char *mqttServer = "mqtt.thingable.com";
int mqttPort = 1883;
void setupMQTT() {
  mqttClient.setServer(mqttServer, mqttPort);
  // set the callback function
  mqttClient.setCallback(callback);
}
```

Figura 8. Configuração do *MQTT* no *ESP32*

A Figura 9 exibe a função de *loop*. Ela lê o sinal analógico do sensor de temperatura e publica o valor em um evento. Ela também aciona os leds e o ventilador conforme informado pela função *callback* (esta recebe o comando para acionar as cargas dos botões presentes no fluxograma no *Node-RED*. O fluxograma será discutido posteriormente).

```
void loop()
{
   int temperature = 0;
   temperature = analogRead(36);
   temperature = 0.03*(2732 - temperature);

   if (!mqttClient.connected())
    reconnect();
   mqttClient.loop();

  pinMode(13, OUTPUT);
  pinMode(14, OUTPUT);
  pinMode(27, OUTPUT);
  pinMode(26, OUTPUT);
  pinMode(12, OUTPUT);

  digitalWrite(13, ledState1);
  digitalWrite(14, ledState2);
  digitalWrite(27, ledState3);
  digitalWrite(26, ledState4);
  if(cooler == 1) analogWrite(12, pwm);
  else analogWrite(12, 0);

// Send data
long now = millis();
if (now - last_time > 6000) {
  sprintf(mensagem, "{\"temperatura\":%d}", temperature);
  mqttClient.publish("Thingable/delufv/Tipo_ELT404_2024/Flavio_01/evt", mensagem);
  Serial.print("Mensagem enviada: ");
  Serial.println(mensagem);
  last_time = now;
}
```

Figura 9. Função de *loop*

Na Figura 10 se encontra a função *callback*. Ela é chamada sempre que o servidor envia um comando ao dispositivo (note que o dispositivo envia apenas 'eventos' ao servidor e o servidor envia apenas 'comandos' ao dispositivo). O comando chega como um objeto *JSON* com o estado que as saídas devem assumir. Este objeto é então destrinchado pela função, que armazena então os estados em variáveis que podem ser acessadas pela função de *loop*, que efetivamente acionará as saídas.

```
void callback(String topic, byte* payload, unsigned int length) {
  Serial.print("Callback - ");
  Serial.print("Message:");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);

    String messageTemp;

  for (int i = 0; i < length; i++) {
    messageTemp += (char)payload[i];
  }

   Serial.println("[SUB] Topic: " + String(topic) + "MSg: " + String(messageTemp));

   StaticJsonDocument<200> json;
   DeserializationError error = deserializeJson(json, messageTemp);

    if (json.containsKey("Led1"))  ledState1 = json["Led1"];
    if (json.containsKey("Led2"))  ledState2 = json["Led2"];
    if (json.containsKey("Led3"))  ledState3 = json["Led3"];
    if (json.containsKey("Led4"))  ledState4 = json["Led4"];
    if (json.containsKey("Cooler"))  cooler = json["Cooler"];
    if (json.containsKey("Speed"))  pwm = json["Speed"];
    last_time -= 5000;
  }
}
```

Figura 10. Função *callback*

4.3 *Programação do servidor via Node-RED*

O código atribuído ao servidor no formato *Node-RED* governa o comportamento dos dispositivos de saída em função dos dispositivos de entrada. Ele também é responsável por gerar um *dashboard* com botões e sliders a ser integrado à um programa supervisório oferecido pela *Thingable!*. A Figura 11 mostra o fluxo completo.

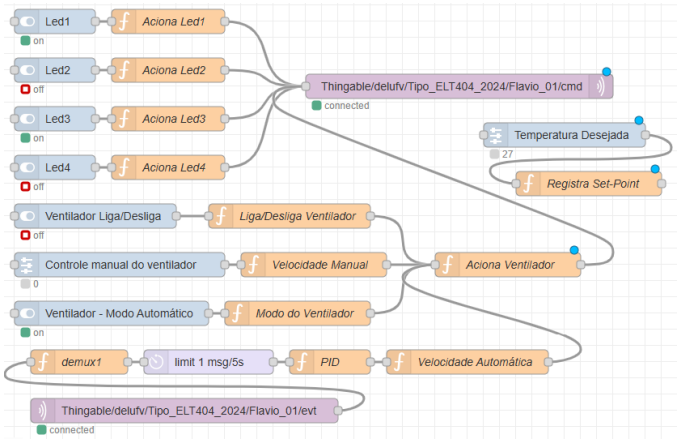

Figura 11. Fluxo completo do servidor

Os nós em azul correspondem aos botões e sliders do *dashboard*. O nó em roxo superior envia comandos ao microcontrolador, e o nó em roxo inferior recebe um evento do microcontrolador, que neste trabalho se trata da medição de um sensor analógico de temperatura.

A Figura 12 contém o código do nó *Aciona Led1*. Este nó é responsável por gerar um objeto *JSON* para ligar ou desligar o Led 1. Os nós para os leds restantes funcionam de maneira análoga.

```
if (msg.payload == true)
return {
  "deviceType": "Tipo_ELT404_2024",
  "deviceId": "Flavio_01",
  "payload": {
    "Led1":  1
  }
}
if (msg.payload == false)
return {
  "deviceType": "Tipo_ELT404_2024",
  "deviceId": "Flavio_01",
  "payload": {
    "Led1":  0
  }
}
```

Figura 12. Código do nó *Aciona Led1*.

Quanto ao ventilador, é possível controlá-lo manualmente ou deixá-lo em modo automático sob o comando de um controlador PID. A Figura 13 exibe o código criado para o nó do controlador PID.

```
let kp = 5;
let ki = 1;
let kd = 4;

let dt = 5;

let integral = 0;
let deriv = 0;
let erro_anterior = context.get('erro_anterior') || 0;

let setpoint = flow.get('setpoint') || 25;
let erro = msg.payload - setpoint;
integral += erro*dt;
deriv = (erro - erro_anterior)/dt;

msg.payload = kp*erro + ki*integral + kd*deriv;

erro_anterior = erro;
context.set("erro_anterior",erro_anterior);

return msg;
```

Figura 13. Código do nó do controlador PID.

O nó *demux* atribui à variável *msg.payload* o valor de temperatura extraíndo-o da carga de dados recebida pelo bloco de inscrição do tópico de eventos *'evt'*. O nó *limit 1msg/5s* permite a passagem de no máximo uma mensagem (correspondente a uma medição de temperatura) cada 5 segundos. Isto é necessário para um funcionamento adequado do controlador PID.

A Figura 14 exibe o código do nó *Aciona Ventilador*. Como as variáveis de um nó não persistem de nó a nó é necessário utilizar variáveis do tipo *flow*, que persistem durante toda a execução do fluxograma. Assim quando um nó registra um valor numa variável deste tipo, outros nós também podem acessar o valor. As variáveis de controle do ventilador são então lidas desta forma e um objeto *JSON* é criado para envio ao microcontrolador.

```
if (flow.get('vent_autom') == 0)
msg.payload = flow.get('vent_speed_man');

if (flow.get('vent_autom') == 1)
msg.payload = flow.get('vent_speed_auto');

if (msg.payload < 0)
 msg.payload = 0;
if (msg.payload > 255)
 msg.payload = 255;

let ligado = flow.get('vent_liga') || false;
if (ligado == false)
{
  msg.payload={
  "Cooler": 0,
  "Speed":  msg.payload
  }

 return {
  "deviceType": "Tipo_ELT404_2024",
  "deviceId": "Flavio_01",
  "payload": msg.payload
  }
}

if (ligado == true)
{
  msg.payload={
  "Cooler": 1,
  "Speed":  msg.payload
 }

 return {
  "deviceType": "Tipo_ELT404_2024",
  "deviceId": "Flavio_01",
  "payload": msg.payload
  }
 }
```

Figura 14. Código do nó *Aciona Ventilador*.

Os nós responsáveis pela comunicação com o dispositivo *IoT* necessitam apenas da inserção do endereço do servidor *IoT* e do tópico de origem das mensagens ou de destino das publicações, conforme exemplificado na Figura 15.

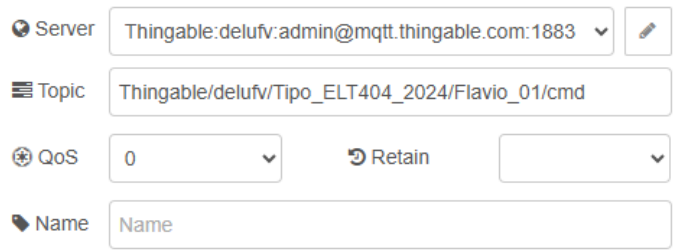

Figura 15. Configuração do nó associado ao tópico de comandos ‘cmd’.

4.4 Criação do supervisório pela Thingable!

O sistema supervisório contém o dashboard gerado pelo *Node-RED*, e dois *widgets* que exibem a temperatura instantânea e o histórico de temperatura. A Figura 16 apresenta a tela do sistema supervisório.

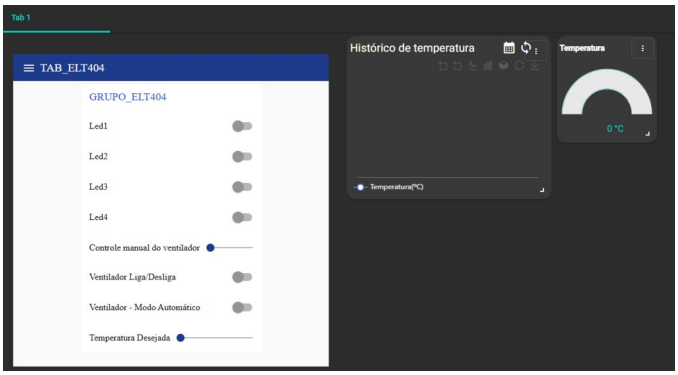

Figura 16. Tela do sistema supervisório

A Figura 17 exibe o gerenciador de base de dados *SCADA* com as *tags* do dispositivo *IoT* utilizadas pelo sistema supervisório.

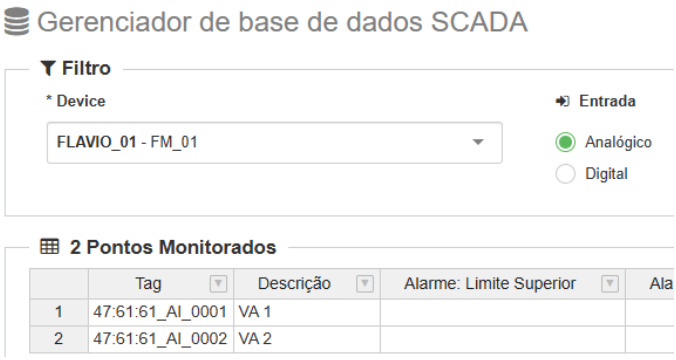

Figura 17. Gerenciador de base de dados SCADA com as tags.

A Figura 18 exibe a configuração do indicador de temperatura com a tag correspondente.

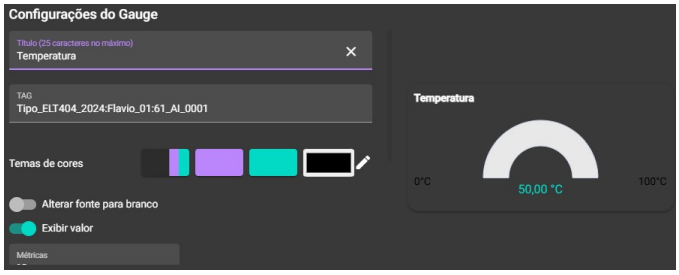

Figura 18. Configuração do indicador de temperatura.

5 Resultados

A Figura 19 exibe a maquete com os dispositivos instalados e acionados.

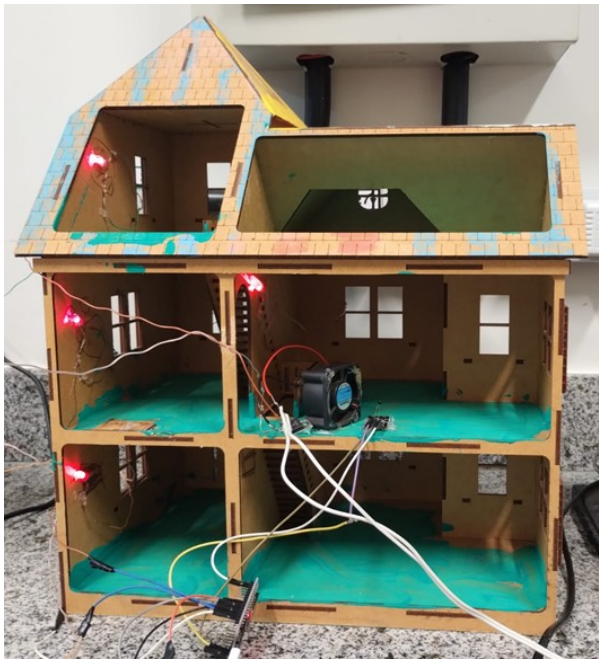

Figura 19. Maquete com os dispositivos instalados e acionados.

A Figura 20 exibe a tela do supervisório em funcionamento. O sensor de temperatura foi aquecido e resfriado durante o teste, e o ventilador, em modo automatico, teve sua velocidade gerenciada pelo controlador PID conforme a temperatura do sensor.

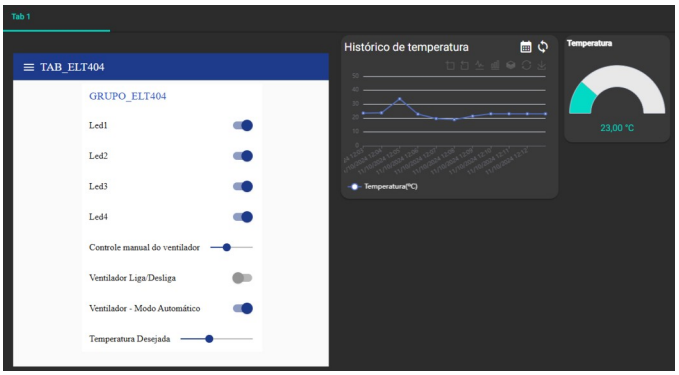

Figura 20. Tela do supervisório em funcionamento.

6 Conclusão

O protótipo desenvolvido demonstrou a capacidade de controlar dispositivos domésticos de forma remota e eficiente. Os resultados obtidos evidenciam o potencial da *IoT* para transformar a vida cotidiana, oferecendo maior conforto e conveniência aos usuários. No entanto, a pesquisa em automação residencial ainda enfrenta desafios como a segurança cibernética e a interoperabilidade entre diferentes sistemas. Futuras pesquisas podem explorar a integração de inteligência artificial para a tomada de decisões autônomas, a otimização do consumo de energia e a criação de interfaces mais intuitivas para os usuários.

Referências Bibliográficas

[1] Somayya Madakam, R. Ramaswamy, Siddharth Tripathi. *Internet of Things (IoT): A Literature Review*. 2015. IT Applications Group, National Institute of Industrial Engineering (NITIE), Vihar Lake, Mumbai, India. Disponível em: <Internet of Things (IoT): A Literature Review (scirp.org) >. Acessado em Agosto de 2024.

[2] John Romkey. Toast of the IoT: The 1990 Interop Internet Toaster. IEEE Consumer Electronics Magazine (Volume: 6, Issue: 1, January 2017). Disponível em:<Toast of the IoT: The 1990 Interop Internet Toaster | IEEE Journals & Magazine | IEEE Xplore>. Acessado em Agosto de 2024.

[3] MIT AUTO-ID LABORATORY. 2024. Disponível em: <https://autoid.mit.edu/>. Acessado em Agosto de 2024.

[4] Mahajan, Mukesh P.; Nikam, Rohit R.; Patil, Vivek P.; Dond, Rahul D. *Smart Refrigerator Using IOT*. 2017. International Journal of Latest Engineering Research and Applications. Disponível em <31.201703127.pdf (ijlera.com)>. Acessado em Agosto de 2024.

[5] Internet Archive. *Amazon Echo is now available for everyone to buy for $179.99, shipments start on July 14*. 2015. Disponível em: <Amazon Echo is now available for everyone to buy for $179.99, shipments start on July 14 | Android Central (archive.org)>. Acessado em Agosto de 2024.

[6] L. Barreto, A. Amaral, T. Pereira. *Industry 4.0 implications in logistics: an overview*. 2017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2351978917306807>. Acessado em Agosto de 2024.

[7] Shadi Al-Sarawi; Mohammed Anbar; Rosni Abdullah; Ahmad B. Al Hawari. *Internet of Things Market Analysis Forecasts, 2020–2030*. 2020. Disponível em: <Internet of Things Market Analysis Forecasts, 2020–2030 | IEEE Conference Publication | IEEE Xplore>. Acessado em Agosto de 2024.

[8] Vaneeta Bhardwaj, Rajat Joshi & Anshu Mli Gaur. *IoT-Based Smart Health Monitoring System for COVID-19*. 2022. Disponível em: <IoT-Based Smart Health Monitoring System for COVID-19 | SN Computer Science (springer.com) >. Acessado em Agosto de 2024.

[9] CSA. *Matter Arrives Bringing A More Interoperable, Simple And Secure Internet Of Things to Life*. 2022. Disponível em: <Matter Arrives Bringing A More Interoperable, Simple And Secure Internet Of Things to Life - CSA-IOT >. Acessado em Agosto de 2024.

[10] Eshrat E. Alahi et al. *Integration of IoT-Enabled Technologies and Artificial Intelligence (AI) for Smart City Scenario: Recent Advancements and Future Trends*. 2023. Disponível em <Sensors | Free Full-Text | Integration of IoT-Enabled Technologies and Artificial Intelligence (AI) for Smart City Scenario: Recent Advancements and Future Trends (mdpi.com)>. Acessado em Agosto de 2024.

[11] Demandsage. *How Many IoT Devices Are There (2024-2032)*. 2024. Disponível em <How Many IoT Devices Are There(2024-2032) (demandsage.com)>. Acessado em Agosto de 2024.

[12] Espressif Systems. ESP32 Series Datasheet Version 4.6. 2024. Disponível em: <esp32_datasheet_en.pdf (espressif.com)>. Acessado em Agosto de 2024.

[13] MQTT. *MQTT: The Standard for IoT Messaging.* 2024. Disponível em: <MQTT - The Standard for IoT Messaging>. Acessado em Agosto de 2024.

[14] Dipa Soni; Ashwin Makwana. A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS (IOT). . Disponível em: <https://www.researchgate.net/profile/Dipa-Soni/publication/316018571_A_SURVEY_ON_MQTT_A_PROTOCOL_OF_INTERNET_OF_THINGSIOT/links/58edafd4aca2724f0a26e0bf/A-SURVEY-ON-MQTT-A-PROTOCOL-OF-INTERNET-OF-THINGSIOT.pdf>. Acessado em Agosto de 2024.

[15] OASIS. *MQTT Version 5.0 - OASIS Standard*. 2019. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html#_Toc3901019>. Acessado em Agosto de 2024.

[16] RF Wireless World. *MQTT Tutorial | MQTT architecture, MQTT protocol use cases*. 2012. Disponível em: <MQTT Tutorial | MQTT architecture, MQTT protocol use cases (rfwireless-world.com)>. Acessado em Agosto de 2024.

[17] Understanding MQTT and Its Role in Event-Driven API System Architecture. Disponível em: <https://medium.com/@santhoshziyam/understanding-mqtt-and-its-role-in-event-driven-api-system-architecture-ea478d666520>. Acessado em 30/01/2025.

[18] Thingable! *IoT e Telemetria Avançada a serviço da sua operação!* 2024. Disponível em: <Soluções de IoT e Telemetria Avançada | thingable!>. Acessado em Agosto de 2024.

[19] Node-RED. Low-code programming for event-driven applications. 2024. Disponível em: <https://nodered.org/>. Acessado em Agosto de 2024.