Controle de iluminação via WIFI através do protocolo MQTT

Brunna Brecht Gomes Correia

1. Introdução

Este relatório descreve o desenvolvimento e a implementação de um sistema de Internet das Coisas (IoT) focado no controle e monitoramento de LEDS utilizando o microcontrolador ESP8266 e o protocolo de comunicação MQTT. O projeto surge da crescente demanda por soluções de automação e monitoramento remoto, visando otimizar processos e proporcionar maior comodidade e eficiência energética em ambientes residenciais ou comerciais.

A motivação para este trabalho reside na exploração de tecnologias de baixo custo e alta flexibilidade para a criação de sistemas IoT. O ESP8266 foi escolhido por sua capacidade de conectividade Wi-Fi e sua vasta comunidade de desenvolvimento, enquanto o MQTT foi selecionado por sua leveza, escalabilidade e eficiência na troca de mensagens em ambientes de recursos limitados.

Os objetivos principais deste projeto são:

- Implementar um sistema de controle remoto de um atuador (ex: LED) via mensagens MQTT.
- Monitorar o estado de um sensor/entrada digital (ex: botão) e publicar suas atualizações via MQTT.
- Estabelecer a comunicação entre o dispositivo embarcado (ESP8266) e um broker MQTT.
- Visualizar e interagir com os dados do sistema através de um dashboard intuitivo.

2. Metodologia

A metodologia empregada no desenvolvimento deste sistema IoT envolveu a integração de hardware embarcado, software de programação e um protocolo de comunicação para a troca de informações.

2.1. Lista de Hardware e Software

Hardware:

- **Microcontrolador:** ESP8266 (Placa NodeMCU ESP-12E ou similar)
- **Atuador:** LED (Diodo Emissor de Luz)
- **Sensor/Entrada:** Botão tátil
- **Componentes Adicionais:** Resistor de 220 Ohm (para o LED), Protoboard, Jumpers
- **Fonte de Alimentação:** Cabo USB (para o ESP8266)

Software:

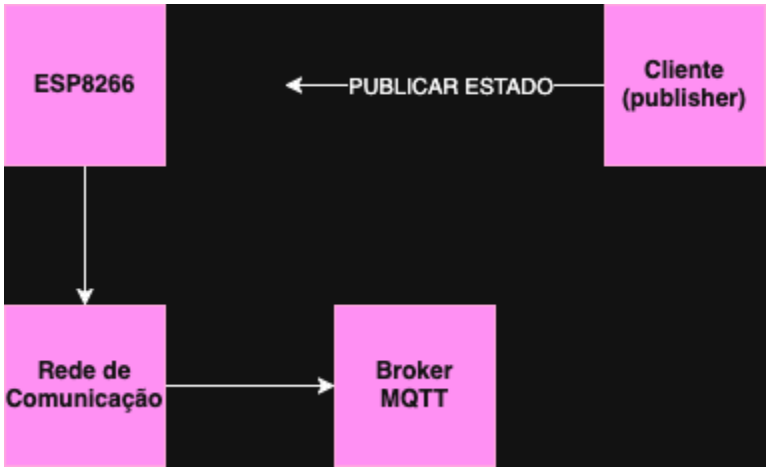
- **IDE de Desenvolvimento:** Arduino IDE
- **Plataforma de Desenvolvimento para ESP8266:** ESP8266 Boards Manager (instalado na Arduino IDE)
- **Bibliotecas Arduino:** `ESP8266WiFi.h`, `PubSubClient.h` (versão de Nick O'Leary)
- **Sistema Operacional (para o Cliente MQTT/Dashboard):** macOS, Raspberry Pi OS
- **Cliente MQTT para Testes/Monitoramento:** Mosquitto_pub/sub
- **Plataforma de Dashboard:** dashboard web customizado]

2.2. Fluxo de Comunicação (MQTT + Wi-Fi)

A comunicação no sistema segue o padrão de publish/subscribe do protocolo MQTT, utilizando a rede Wi-Fi como meio de transporte:

1. **Inicialização da ESP8266:** O microcontrolador ESP8266 inicializa e tenta se conectar à rede Wi-Fi especificada (`ssid` e `password`).
2. **Conexão ao Broker MQTT:** Após a conexão Wi-Fi ser estabelecida, a ESP8266 tenta se conectar ao broker MQTT no endereço e porta configurados (`mqtt_server`, `mqtt_port`).
3. **Assinatura de Tópicos:** Uma vez conectado ao broker, a ESP8266 assina o tópico de controle do LED (`home/dispositivo01/led/control`). Isso significa que ele receberá qualquer mensagem publicada nesse tópico.
4. **Publicação de Status do Botão:** Quando o botão conectado ao GPIO4 é pressionado (detectado com lógica de debounce), a ESP8266 publica uma mensagem (`"PRESSIONADO"`) no tópico `home/dispositivo01/botao/status`.

5. **Publicação de Status do LED:** Sempre que o estado do LED é alterado (seja por um comando MQTT ou pelo botão físico), a ESP8266 publica o novo estado (`"LIGADO"` ou `"DESLIGADO"`) no tópico `home/dispositivo01/led/status`.
6. **Recebimento de Comandos para o LED:** Um cliente MQTT externo (por exemplo, um dashboard web ou aplicativo) publica uma mensagem (`"LIGAR"` ou `"DESLIGAR"`) no tópico `home/dispositivo01/led/control`.
7. **Ação no LED:** A ESP8266, tendo assinado esse tópico, recebe a mensagem e altera o estado do LED físico correspondentemente.
8. **Monitoramento:** O cliente MQTT externo também assina os tópicos de status (`home/dispositivo01/led/status`, `home/dispositivo01/botao/status`) para visualizar as atualizações em tempo real no dashboard.



3. Resultados

Problemas Encontrados na Implementação do Circuito com ESP8266

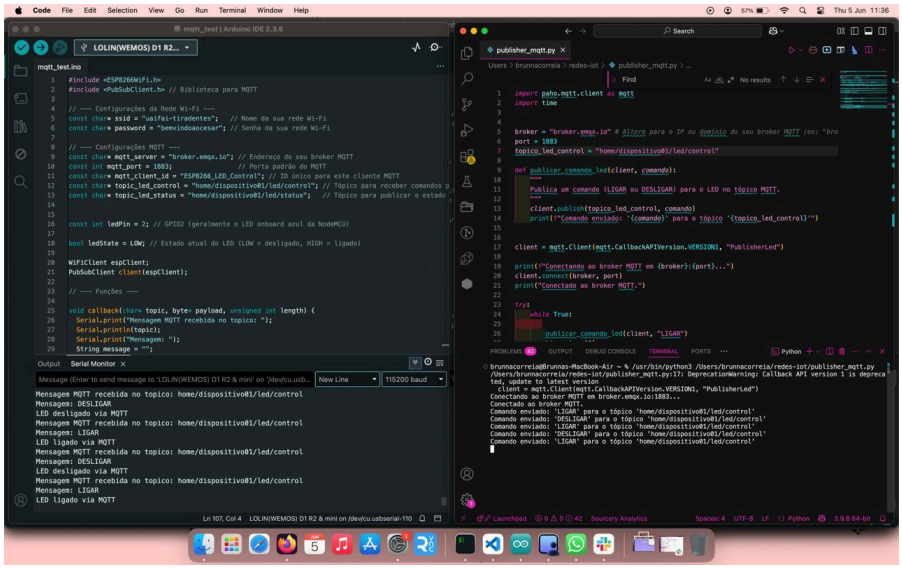
Durante a fase de implementação do projeto, alguns desafios significativos foram enfrentados na parte do circuito que impediram a conclusão do mesmo. Especificamente, a tentativa de integrar o módulo ESP8266 ao circuito se mostrou mais complexa do que o previsto, resultando em instabilidades que comprometeram o funcionamento geral do sistema.

As principais dificuldades foram observadas na **alimentação do ESP8266** e na **estabilidade das conexões dos pinos de E/S**.

- **Reinícios inesperados:** O módulo reiniciava constantemente, especialmente durante a transmissão de dados Wi-Fi ou a comutação do LED, indicando possíveis quedas de tensão ou insuficiência de corrente.

- **Dificuldade de conexão Wi-Fi:** Em alguns momentos, o ESP8266 tinha dificuldade em estabelecer ou manter a conexão com a rede Wi-Fi, o que pode ser um sintoma de problemas de alimentação ou interferência.

Essas dificuldades no circuito físico se tornaram um gargalo para o avanço do projeto, impedindo que os objetivos completos de ter um sistema funcional de controle e monitoramento. Reconhecendo a importância de um circuito robusto e estável como base para qualquer projeto embarcado, e essa experiência ressalta a necessidade de mais tempo e recursos para depuração e otimização da parte de hardware, dessa forma foi possível realizar uma comunicação mínima entre o broker e a esp8266 através do mosquitto, assim sendo possível enviar e receber comandos para ligar ou desligar o LED, fazendo o LED da própria placa acender e apagar de acordo com o comando recebido no protocolo mqtt



4. Conclusão

Este projeto demonstrou viabilidade da implementação de um sistema IoT para controle e monitoramento utilizando a plataforma ESP8266 e o protocolo MQTT, porém não foi possivel colocar em prática totalmente.

Desafios Encontrados:

- A curva de aprendizado inicial com o protocolo MQTT e a configuração do broker foi um desafio, além de alguns problemas com a pinagem da esp8266 d1 mini
- A estabilidade da conexão Wi-Fi em ambientes com muitos roteadores foi um ponto de atenção, exigindo por vezes a otimização do posicionamento do dispositivo.
- Dificuldade no uso de uma placa a qual não foi feita para uso em protoboards, não sendo possível utilizar vários sensores e atuadores ao mesmo tempo

Aprendizados:

- Aprofundamento no funcionamento do protocolo MQTT (publicar/assinar, tópicos, QoS).
- Melhor compreensão da programação de dispositivos embarcados com ESP8266 e suas particularidades (pinagem, modos de boot).
- Experiência prática na integração de hardware e software para a construção de um sistema IoT completo.

4. Apêndice

4.1. Códigos Completos

Esp8266:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h> // Biblioteca para MQTT

// --- Configurações da Rede Wi-Fi ---
const char* ssid = "uaifai-tiradentes";   // Nome da sua rede Wi-Fi
const char* password = "bemvindoaocesar"; // Senha da sua rede Wi-Fi

// --- Configurações MQTT ---
const char* mqtt_server = "broker.emqx.io"; // Endereço do seu broker MQTT
const int mqtt_port = 1883;                 // Porta padrão do MQTT
const char* mqtt_client_id = "ESP8266_LED_Control"; // ID único para este cliente MQTT
const char* topic_led_control = "home/dispositivo01/led/control"; // Tópico para
receber comandos para o LED
const char* topic_led_status = "home/dispositivo01/led/status";   // Tópico para
publicar o estado atual do LED



const int ledPin = 2;

bool ledState = LOW;

WiFiClient espClient;
PubSubClient client(espClient);

// --- Funções ---
```

```
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Mensagem MQTT recebida no topico: ");
 Serial.println(topic);
 Serial.print("Mensagem: ");
 String message = "";
 for (int i = 0; i < length; i++) {
   message += (char)payload[i];
 }
 Serial.println(message);

 if (String(topic) == topic_led_control) {
   if (message == "LIGAR") {
     digitalWrite(ledPin, HIGH);
     ledState = HIGH;
     Serial.println("LED ligado via MQTT");
     client.publish(topic_led_status, "LIGADO");
   } else if (message == "DESLIGAR") {
     digitalWrite(ledPin, LOW);
     ledState = LOW;
     Serial.println("LED desligado via MQTT");
     client.publish(topic_led_status, "DESLIGADO");
   } else {
     Serial.println("Comando de LED invalido. Use 'LIGAR' ou 'DESLIGAR'.");
   }
 }
}

//  reconectar ao broker MQTT
void reconnectMqtt() {
 // Loop até estar conectado
 while (!client.connected()) {
   Serial.print("Tentando conectar ao broker MQTT...");
   // Tenta conectar usando o Client ID definido
   if (client.connect(mqtt_client_id)) {
     Serial.println("conectado!");
     // Uma vez conectado, assina os tópicos que nos interessam
     client.subscribe(topic_led_control);
     Serial.print("Assinado ao topico: ");
     Serial.println(topic_led_control);
     // Publica o estado inicial do LED
     client.publish(topic_led_status, ledState == HIGH ? "LIGADO" : "DESLIGADO");
```

```
    } else {
      Serial.print("falha, rc=");
      Serial.print(client.state()); // Código de retorno da falha
      Serial.println(" tentando novamente em 5 segundos");
      // Espera 5 segundos antes de tentar novamente
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);

  // Configuração do hardware
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState); // Define o estado inicial do LED (desligado)

  Serial.println("\nHardware configurado.");

  // Conecta ao Wi-Fi
  Serial.print("Conectando a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi conectado!");
  Serial.print("Endereco IP: ");
  Serial.println(WiFi.localIP());

  // Configura o cliente MQTT
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

void loop() {
  // Garante que o cliente MQTT esteja conectado
  if (!client.connected()) {
    reconnectMqtt();
  }
  client.loop(); // Processa mensagens MQTT pendentes e mantém a conexão ativa
```

```
}
```

Broker:

```
import paho.mqtt.client as mqtt
import time

broker = "broker.emqx.io" # Altere para o IP ou domínio do seu broker MQTT (ex:
"broker.emqx.io")
port = 1883
topico_led_control = "home/dispositivo01/led/control"

def publicar_comando_led(client, comando):
    """
    Publica um comando (LIGAR ou DESLIGAR) para o LED no tópico MQTT.
    """
    client.publish(topico_led_control, comando)
    print(f"Comando enviado: '{comando}' para o tópico '{topico_led_control}'")


client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION1, "PublisherLed")

# Conecta ao broker MQTT
print(f"Conectando ao broker MQTT em {broker}:{port}...")
client.connect(broker, port)
print("Conectado ao broker MQTT.")

try:
    while True:
        publicar_comando_led(client, "LIGAR")
        time.sleep(2)
        publicar_comando_led(client, "DESLIGAR")
        time.sleep(2)

except KeyboardInterrupt:
    print("\nPublisher de LED encerrado.")
    client.disconnect()
    print("Desconectado do broker MQTT.")
```