UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS ARARANGUÁ
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO
ENGENHARIA DA COMPUTAÇÃO

Eduardo Zambotto da Silva
João Pedro Blanco

**Relatório Final**

Relatório Final da Disciplina

Matéria: Redes sem Fios (DEC 7563)
Professora:Analúcia Schiaffino Morales

Araranguá

2025

# Introdução

O projeto tem como objetivo desenvolver um sistema de monitoramento de fluxo de pessoas em ambientes fechados utilizando o sistema embarcado do ESP32 que se comunica com um sensor de presença humana, a fim de captar esses dados, estruturá-los, e enviá-los para uma nuvem, permitindo o monitoramento remoto em tempo real.

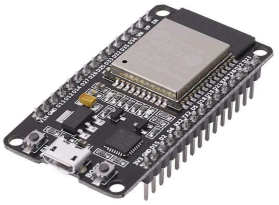
# Descrição Detalhada do Projeto

## Componentes de Hardware

Foi utilizado um ESP 32 como microcontrolador principal, aproveitando seu módulo Wi-Fi integrado para realizar a comunicação com a plataforma Adafruit IO via protocolo MQTT. O sensor LD 2410, um radar de ondas milimétricas de 24 GHz com capacidade de detecção de presença e movimento, foi conectado à porta serial Serial1 do ESP 32, utilizando os pinos GPIO 16 (RX) e GPIO 17 (TX).

- **ESP32:** microcontrolador composto por diversos módulos integrados, sendo os relevantes para o sistema desenvolvido o módulo de Wi-fi, Bluetooth, UART, além dos módulos correspondentes à sua capacidade de processamento e armazenamento, que dispõe de um microprocessador dual-core Tensilica Xtensa LX6 operando a 240 MHz, e uma memória flash externa de 4 MB .

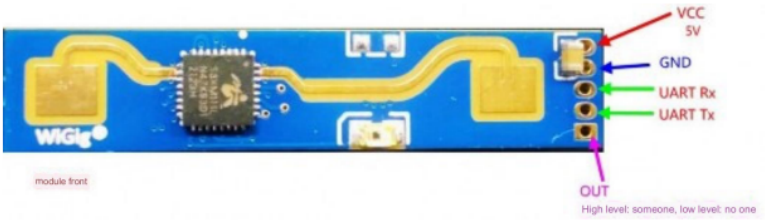
**Figura 01 -** ESP32 Dev Kit V1



Fonte: https://vidadesilicio.com.br/produto/esp32-esp-wroom-32-nodemcu/

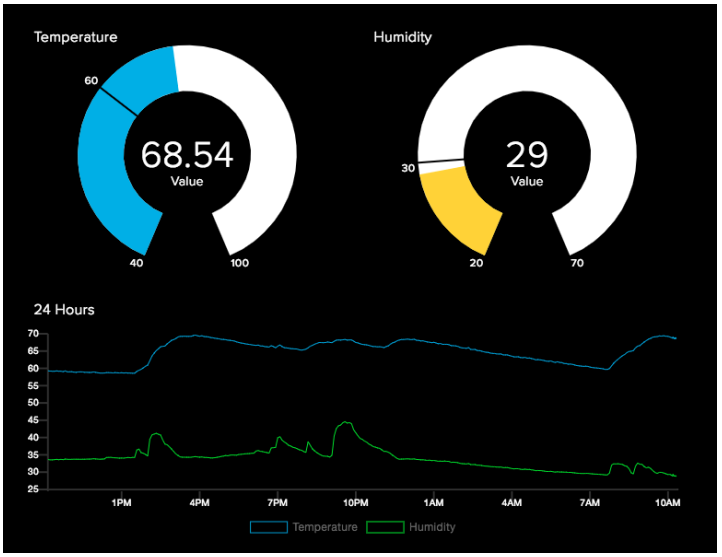
- **HLK-LD2410B:** sensor de presença humana que realiza a comunicação via protocolo UART e Bluetooth Low Energy (BLE), tecnologia que será explorada no projeto. Por meio do seu protocolo de comunicação é possível configurar alguns parâmetros como a distância de alcance do sensor, o tempo de intervalo de leituras, entre outros. O componente é alimentado com 5V, e possui uma pinagem simples, listada a seguir:
    - GND/VCC: pinos utilizados na alimentação do sensor, sendo GND ligado à 0V e VCC à 5V;
    - RX/TX: pinos utilizados na comunicação via UART;
    - OUT: tem como saída o valor 1 se for detectada alguma presença e 0 caso contrário.

**Figura 02 -** sensor HLK-LD2410B e sua pinagem




Fonte: https://assets.super.so/79c0d2a8-d37a-438f-8fbe-c44778f3b0dd/files/7c3607bd-f703-43f5-9f22-f369f00c37bd.pdf

**Figura 03 -** Exemplo de dashboard do Adafruit IO




Fonte: https://io.adafruit.com/blog/notebook/2019/02/25/updated-charts-and-gauges/

## Componentes de Software

O software foi desenvolvido em C++, utilizando o ambiente Arduino IDE, por sua compatibilidade com o ESP 32 e ampla comunidade de suporte.

As seguintes bibliotecas de terceiros foram utilizadas:

- **`ld2410.h`**
  - **Função:** Comunicação com o sensor radar LD2410
  - **Justificativa:** Biblioteca oficial e compatível com o sensor, permitindo leitura direta dos dados de distância, presença e movimento, com comandos simples e suporte à Serial1.
- **`WiFi.h`**
  - **Função:** Conexão do ESP 32 à rede Wi-Fi
  - **Justificativa:** Biblioteca nativa do ESP 32, que permite gerenciamento da conexão à internet, fundamental para o envio dos dados ao broker MQTT.
- **`Adafruit_MQTT.h`** e **`Adafruit_MQTT_Client.h`**
  - **Função:** Comunicação com a plataforma Adafruit IO via protocolo MQTT.

- **Justificativa:** Bibliotecas oficiais da Adafruit, simplificam a autenticação, publicação e manutenção da conexão com o broker MQTT, sendo ideais para integração com os feeds da plataforma.

## Funcionamento do Projeto

O código principal inicia estabelecendo uma conexão Wi-Fi com a rede especificada e, em seguida, conecta-se ao broker MQTT da plataforma Adafruit IO. Após essa etapa, o sistema inicializa o sensor radar LD2410 por meio da interface serial secundária do ESP 32 (Serial1). A lógica central do projeto consiste na detecção de presença e movimento de pessoas, seguida da contagem e publicação desses dados em tempo real.

**Fluxo de execução:**

1. **Inicialização**: o ESP 32 conecta-se à rede Wi-Fi e configura a comunicação serial com o sensor.
2. **Leitura de dados**: o radar LD2410 fornece continuamente dados de detecção de presença e distância.
3. **Processamento**: caso seja detectado movimento, após um período de inatividade, incrementa-se o contador de pessoas.
4. **Publicação MQTT**: a cada 5 segundos, o valor do contador é enviado à Adafruit IO, por meio de um feed MQTT.

**Trechos de código relevantes:**

*Conexão à rede Wi-Fi:*

```
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("WiFi conectado!");
```

*Conexão ao broker MQTT:*

```
void MQTT_connect() {
  int8_t ret;
  if (mqtt.connected()) return;
  while ((ret = mqtt.connect()) != 0) {
    Serial.println("Falha ao conectar MQTT");
    delay(5000);
  }
  Serial.println("Conectado ao MQTT!");
}
```

*Lógica de contagem e envio:*

```
void ProcessDistance()
{
  if (FindLast() < 5)
  {
    return;
  }

  int count = 0;
  int movement = buffer[FindLast()] - buffer[0];

  for (int i = 1; i < FindLast(); i++)
  {
    if (buffer[i] < buffer[i - 1])
    {
      count++;
```

```
    }
    else
    {
      count--;
    }
  }

  if (count > 4)
  {
    peopleCount++;
  }
  else if (count < -4)
  {
    peopleCount = max(0, peopleCount - 1);
  }
  else
  {
    return;
  }
}

if (radar.presenceDetected()) {
  unsigned long now = millis();
  if (now - lastMovementTime > 3000) {
    peopleCount++;
    lastMovementTime = now;
  }
```

```
}

if (millis() - LastSendTime > 5000) {
  Serial.print("Pessoas detectadas: ");
  Serial.println(peopleCount);
  peopleFeed.publish(peopleCount);
  LastSendTime = millis();
}
```

**Interação entre os componentes:**

O ESP 32 é responsável por coordenar todo o funcionamento do sistema. Ele coleta os dados do sensor LD2410, analisa o comportamento do ambiente e envia os resultados periodicamente para a nuvem, utilizando sua conectividade Wi-Fi e o protocolo MQTT. A plataforma Adafruit IO armazena e exibe os dados publicados, permitindo o monitoramento remoto em tempo real.

### Metodologia Experimental

Os experimentos foram realizados em ambientes fechados, tanto em salas de aula quanto em ambiente residencial, buscando simular cenários reais de detecção de presença em locais internos. A principal estratégia adotada foi a simulação de entradas e saídas de pessoas em frente ao sensor radar LD2410. Durante os testes, entre 1 e 3 pessoas se movimentaram pelo ambiente, possibilitando a verificação do funcionamento da lógica de contagem implementada no código.

A conectividade Wi-Fi foi mantida de forma estável durante os testes, apesar de não ter sido avaliada formalmente em diferentes distâncias do ponto de acesso. A comunicação com a plataforma Adafruit IO foi monitorada diretamente pelo Serial Monitor da Arduino IDE, confirmando o envio periódico das contagens registradas. Esses testes permitiram validar a detecção de presença e o envio dos dados via MQTT, reforçando a adequação do sistema para aplicações básicas de monitoramento de ocupação em tempo real.

## Resultados e Discussão

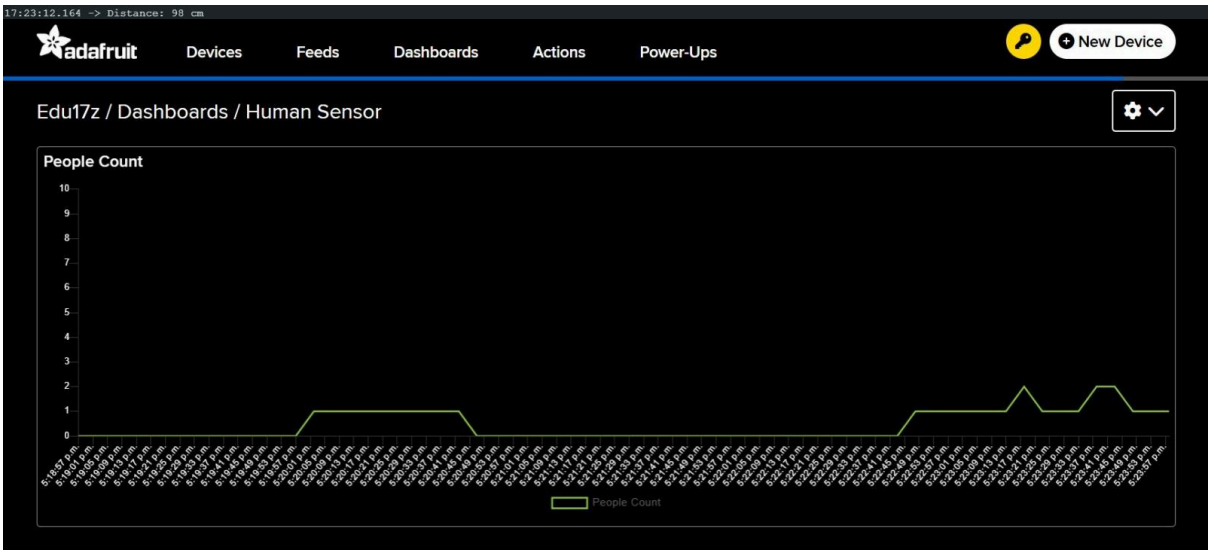
Durante os testes, o sistema demonstrou-se funcional, sendo capaz de detectar a presença de pessoas com precisão satisfatória em ambientes fechados. A contagem de pessoas detectadas foi corretamente incrementada em situações simuladas de entrada e saída de indivíduos pelo ambiente. Porém, pela natureza do sensor e a lógica implementada, o sistema desenvolvido é incapaz de diferenciar se mais de uma pessoa entrar por vez, e se houver muitas detecções simultâneas, como, por exemplo, uma pessoa entrando/saindo, porém, ao fundo, houver movimento de pessoas, o sistema pode apresentar instabilidade.

O envio dos dados para a plataforma Adafruit IO ocorreu com sucesso a cada 5 segundos, sem apresentar falhas perceptíveis de comunicação. Isso confirma a eficiência do protocolo MQTT aliado à conectividade Wi-Fi do ESP 32, mesmo em ambientes com infraestrutura de rede doméstica ou de laboratório.

**Figura 04 -** Saídas no terminal


Fonte: Figura gerada pelo autor.

**Figura 04 -** Dashboard na plataforma Adafruit IO




Fonte: Figura gerada pelo autor,

Embora não tenham sido coletadas métricas específicas de throughput, latência ou taxa de perda de pacotes, a baixa frequência de envio (a cada 5 segundos) e o baixo volume de dados (inteiros simples) tornam o sistema leve e apropriado para aplicações IoT em que eficiência energética e estabilidade de conexão são importantes.

## Desafios e Aprendizados

Durante o desenvolvimento do projeto, enfrentamos diversos desafios técnicos que exigiram pesquisa, testes práticos e tomada de decisões em grupo. Essas dificuldades proporcionaram um aprendizado significativo sobre sensores, conectividade e fluxos de dados em sistemas embarcados com IoT.

Um dos primeiros obstáculos foi a utilização do sensor de radar LD2410, que apresentou instabilidades ao detectar múltiplas pessoas simultaneamente. Percebemos que o sensor funciona de maneira mais confiável quando apontado para ambientes com movimentação controlada — por exemplo, entradas e saídas com passagem de apenas uma pessoa por vez. Apesar de o problema não ter sido completamente resolvido, conseguimos contornar a limitação com ajustes no posicionamento e lógica de uso.

Na parte de comunicação serial, tentamos inicialmente configurar o sensor via Bluetooth, mas não obtivemos sucesso. Por isso, optamos pela comunicação serial via Serial1 do ESP 32, utilizando as portas RX e TX, o que funcionou de forma estável.

A conexão Wi-Fi com o ESP 32 não apresentou dificuldades. A rede do laboratório se mostrou confiável, e conseguimos realizar a conexão de forma rápida com os dados fornecidos.

O maior desafio técnico do projeto foi a integração com a plataforma Adafruit IO. No início, utilizamos o método io.run() para enviar os dados, mas enfrentamos problemas por ultrapassar o limite de requisições por minuto da plataforma gratuita. Isso causava instabilidades e falhas no envio. Após pesquisa e testes, substituímos essa abordagem pelo uso do protocolo MQTT, por meio da biblioteca Adafruit_MQTT_Client, o que resolveu os problemas de envio e nos ensinou sobre o funcionamento da arquitetura *publish/subscribe* em sistemas IoT.

Outra parte crítica foi a implementação do contador de pessoas, que exigiu o uso de buffers e controle temporal das leituras. Aprendemos na prática a limitar a taxa de leitura do sensor e a evitar múltiplas contagens de um mesmo evento de passagem. O uso de um array buffer e a lógica de leitura em intervalos definidos foram fundamentais para o funcionamento correto do sistema.

Também aplicamos o controle de tempo usando a função millis() no lugar de delay(), o que já conhecíamos de projetos anteriores, e que se mostrou útil para manter o programa responsivo.