Protótipo de detecção de movimentos para casa inteligente de baixo custo

Felipe Matheus Mattoso Torres[1], Leandro Camara Ledel[2]

[1]Curso de Tecnologia em Análise e Desenvolvimento de Sistemas - Campus Hortolândia - Instituto Federal de São Paulo (IFSP)

[2]Área de Informática – Campus Hortolândia – Instituto Federal de São Paulo (IFSP)

t.felipe@aluno.ifsp.edu.br, ledel@ifsp.edu.br

Abstract. *Nowadays, there is a large increase in the use of smart devices for home automation, with the market value potentially reaching $163.24 billion by 2028. Adding to this the high cost of installing a residential security system in Brazil, this highlights the importance of low-cost home automation projects for the general population. The goal of this project is to develop an Arduino prototype connected to a motion sensor and integrated with a network protocol responsible for sending notifications to the user whenever any movement is detected.*

Resumo. *Atualmente, nota-se um grande aumento no uso de dispositivos inteligentes para automatização residencial, podendo chegar a um valor de mercado de US$ 163,24 bilhões em 2028. Somando isso ao alto custo no Brasil para instalar um sistema de segurança residencial, chega-se à conclusão da importância que projetos de automação residencial de baixo custo possam ter para a população em geral. O objetivo deste trabalho é desenvolver um protótipo Arduino ligado a um sensor capaz de detectar movimentos conectado a um protocolo de rede responsável por enviar notificações ao usuário quando qualquer movimentação for detectada.*

1. Introdução

Atualmente, a utilização de dispositivos inteligentes para automatização residencial, desde processos simples, como acender ou apagar uma lâmpada, até processos complexos como elaborar sistemas de monitoramento de irrigação de solo, vem crescendo de forma significativa no Brasil, visto que o número de residências com qualquer tipo de sistema automatizado saltou de 300 mil, em 2016, para 2 milhões, em 2020 (PACETE, 2022). Segundo Pacete (2022), o crescimento projetado é de 22% ao ano até 2025, gerando uma receita estimada em R$ 16 bilhões. A partir da análise desses números, percebe-se a notável tendência de crescimento de automações residenciais em qualquer nível, seja ela apenas para fornecer conforto à casa com um simples comando de voz para ligar a televisão, ou até mesmo para oferecer segurança através de sensores capazes de identificar movimentos, portas ou janelas se abrindo.

Um ponto que impacta diretamente na decisão de brasileiros que desejam implementar qualquer tipo de sistema de segurança é o custo do mesmo, podendo

ficar entre R$ 5.000,00 e R$ 7.000,00 (HABITÍSSIMO, 2024), não contando ainda com possíveis gastos adicionais de manutenção e taxas de mensalidade que podem ser cobradas, o que acaba pesando muito no âmbito financeiro do brasileiro.

De acordo com a IDC (*International Data Corporation*), é possível observar um grande crescimento da *IoT* ("*Internet of Things*" ou "Internet das coisas", em tradução para o português brasileiro) nos últimos anos. Segundo a IDC, o número de dispositivos IoT era de mais de 10 bilhões em 2021. Ainda assim, esses números devem aumentar até 2025, fazendo com que a geração de dados global exceda 73 trilhões de gigabytes (SAP, 2024). Os números também evidenciam que o IoT tem grande importância na integração do mundo real com o virtual. Essa informação é corroborada por uma votação, que mostrou que 47% das empresas na América Latina estão em processo de implementação de IoT, enquanto 35% ainda planejam investir em projetos com a tecnologia (IOT SNAPSHOT, 2022).

O objetivo deste projeto é desenvolver um protótipo Arduino de baixo custo de maneira a aplicar o conceito de casa inteligente com automação através do sensor de movimento PIR HC-SR505, que ao identificar qualquer movimentação dentro de sua área de detecção, deverá enviar uma mensagem para um aplicativo - com a finalidade de notificar uma possível invasão de um ambiente, por exemplo. Também deverá ser realizada, ao final do projeto, uma análise de custos envolvendo os dispositivos de hardware necessários para a construção do protótipo.

O artigo está estruturado na seguinte maneira: a Seção 2 apresenta os trabalhos correlatos a este projeto; na Seção 3 são relatados os referenciais teóricos; a Seção 4 contém a metodologia usada no projeto, bem como as tecnologias e meios para a reprodução do mesmo; a modelagem de funcionamento e o desenvolvimento do projeto são apresentados na Seção 5; enquanto a Seção 6 contém os testes e resultados do protótipo; e por fim, a Seção 7 conclui o artigo.

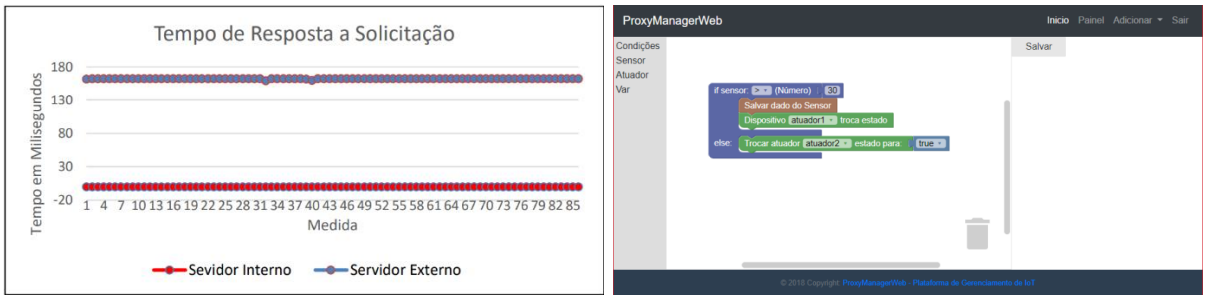
2. Trabalhos Correlatos

Nesta seção são abordadas aplicações que possuem o propósito de desenvolver protótipos Arduino de baixo custo com funcionalidades programáveis, bem como a instalação de sensores inteligentes. Esta seção também irá abordar o estado atual da arte de pesquisas relacionadas ao IoT.

2.1 Gerenciador de Coleta de Dados para Redes de Sensores sem Fio

Este trabalho conta com uma infraestrutura baseada em *Fog Computing* (Computação em névoa, em tradução livre), e utiliza o protocolo MQTT *(Message Queuing Telemetry Transport*) para transferir dados entre servidor e dispositivos. Na Figura 1(a), é apresentada a comparação de latências entre o servidor interno e o servidor externo, enquanto na Figura 1(b) é apresentada a tela de tarefas do dispositivo.

Tem como principal objetivo a redução da latência (tempo de resposta) e o tráfego de pacotes na rede entre um servidor de processamento e uma WSN, tendo sua aplicação voltada para projetos em que a baixa latência seja essencial. (COSTA, 2019).

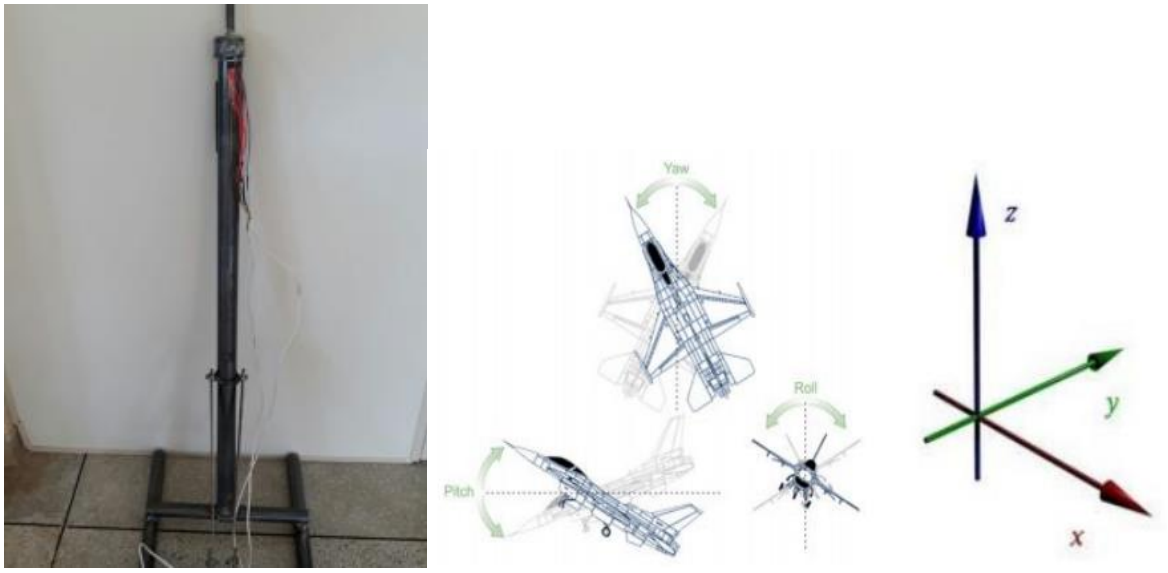


(a) (b)

Figura 1. Imagens do desenvolvimento do projeto (a) amostra de tempo entre coletas e (b) tarefas do dispositivo. Fonte: (COSTA, 2019).

2.2 Sistema Domiciliar de Cuidados em Saúde: Mecanismo de Detecção de Quedas de Pessoas.

O projeto tem como objetivo desenvolver um dispositivo de *hardware* e *software* de baixo custo, evidenciado na Figura 2(a), cuja finalidade é detectar quedas de idosos e enviar um alerta de socorro ao telefone de pessoas registradas previamente. O envio é feito a partir dos protocolos de rede MQTT – sendo este implementado junto ao *broker* Eclipse Mosquitto - e HTTP (*Hypertext Transfer Protocol*). Para a detecção de quedas, o dispositivo utiliza um sensor acelerômetro junto a fórmulas físicas que calculam possíveis quedas baseando-se em propriedades físicas como dimensão, velocidade e aceleração, estas representadas a partir da Figura 2(b) (DA SILVA, 2018).



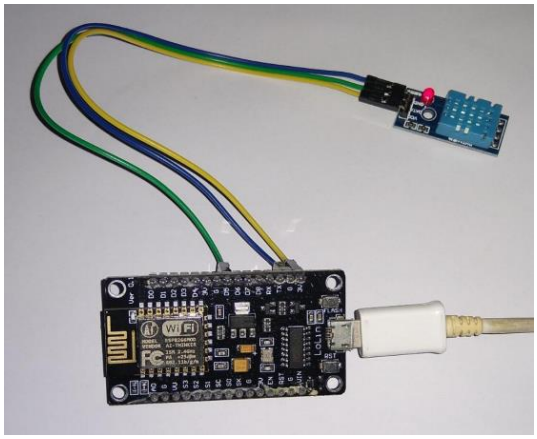
(a) (b)

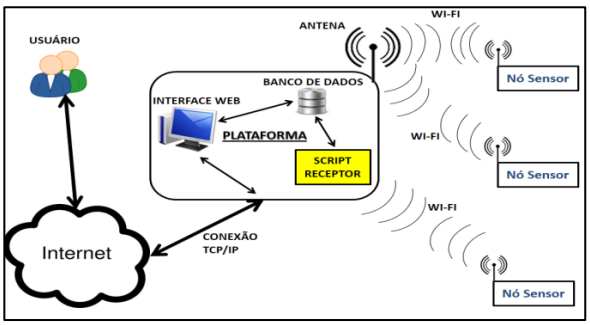
Figura 2. Imagens do desenvolvimento do projeto (a) mecanismo de simulação de quedas e (b) ângulos de atitude (Yaw, Pitch e Roll). Fonte: (DA SILVA, 2018).

2.3 Automação Residencial utilizando uma Plataforma Open-Source

A partir da utilização da placa de desenvolvimento NodeMCU, apresentada na Figura 3(a), este projeto tem como objetivo desenvolver uma interface w*eb* para controlar, alterar e adicionar conjuntos ao sistema, tendo como finalidade automatizar ações realizadas no cotidiano, tais quais acender e apagar lâmpadas, ligar e desligar alarmes, entre outros - a Figura 3(b) apresenta a visão geral do sistema desenvolvido (CANDIDO, 2020).

Ademais, o projeto visa alcançar a automatização com custo acessível e código aberto, tornando-o mais fácil para que o usuário tenha liberdade para modificar e implementar diversos equipamentos e funcionalidades. (CANDIDO, 2020).



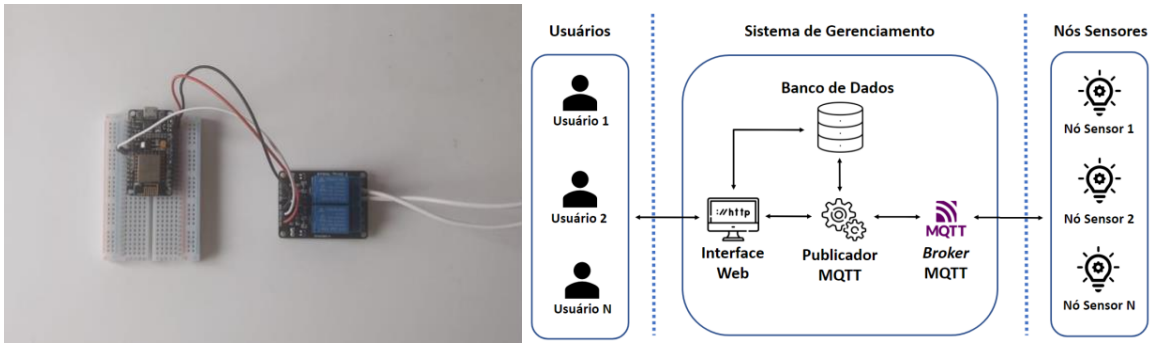


(a) (b)

Figura 3. Imagens do desenvolvimento do projeto (a) Foto do Nó Sensor e (b) Visão geral. Fonte: (CANDIDO, 2020).

2.4 Sistema de gerenciamento remoto de dispositivos residenciais

O artigo tem como objetivo criar o protótipo de um sistema de gerenciamento de dispositivos de forma remota em residências, este representado pela Figura 4(a), utilizando uma aplicação *web* para controlar os dispositivos e um sistema eletrônico para controlá-los de forma remota. Além disso, o protocolo de comunicação *MQTT* também é utilizado para enviar dados ao servidor, que por sua vez determina se o dispositivo está ligado ou desligado, armazena informações do dispositivo (como seu estado, por exemplo) e seu identificador – fazendo-o a partir de um banco de dados (LEITE, 2022). A Figura 4(b) apresenta a visão geral e o funcionamento do sistema e do servidor mencionado.



(a) (b)

Figura 4. Imagens do desenvolvimento do projeto (a) Protótipo final e (b) Arquitetura geral do sistema. Fonte: (LEITE, 2022).

3. Referencial Teórico

Esta seção tem o objetivo de apresentar os principais tópicos da fundamentação teórica, bem como os conceitos voltados para a *IoT* que serão utilizados neste projeto.

3.1 Internet of Things

O termo *IoT* corresponde a uma coleção de dispositivos conectados e a facilidade da conexão entre nuvem e dispositivos, viabilizada pela tecnologia. Graças à popularização e ao barateamento de *chips* de computador e disponibilidade de telecomunicações de alta largura de banda, hoje em dia temos diversos dispositivos

do dia a dia, como carros, máquinas, aspiradores e até escovas de dentes, que podem utilizar sensores com o objetivo de captar dados e responder de forma inteligente ao usuário (AMAZON, 2024).

Por mais que se pense em *IoT* como um tema extremamente novo, a primeira menção do termo foi em 1999, pelo cientista Kevin Ashton, que o descreveu como uma rede para conectar objetos do mundo físico à Internet (MEIO & MENSAGEM, 2022). Portanto, o termo que vem se popularizando cada vez mais, já existe há um tempo considerável.

3.2 Hardware utilizado

As subseções a seguir apresentam o hardware utilizado neste projeto.

3.2.1 Placa controladora ESP32

O ESP32, representado pela Figura 5, é um dispositivo denominado placa controladora ou microcontrolador desenvolvido pela empresa Espressif. Analisando suas especificações técnicas, é possível afirmar que ele possui um desempenho superior para aplicações que demandam maior processamento quando comparado a outras placas, como o Arduino Uno (MAKIYAMA, 2023).

Entre suas principais características e vantagens, destaca-se a presença de 34 pinos de entrada e saída (*GPIOs*) que podem ser configuradas para diversas funcionalidades, incluindo as necessárias para este projeto, como leitura de sensores, controles de *display* e até mesmo motores (MAKIYAMA, 2023). Além disso, já vem integrado com as tecnologias Wi-Fi e Bluetooth, garantindo grande flexibilidade, ampla conexão e tornando-se ideal para projetos de *IoT* (USINAINFO, 2023).

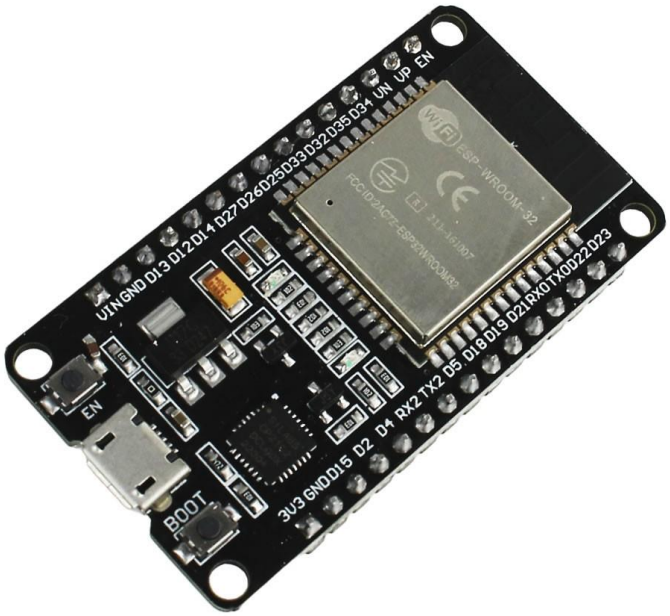


Figura 5. Imagem da placa de desenvolvimento ESP32 ESP-VROOM-32 com Wi-Fi e Bluetooth. Fonte: (MERCADO LIVRE, 2024).

3.2.2 Sensores inteligentes

Os sensores inteligentes são dispositivos usados em sua grande maioria para detectar características do ambiente e gerar sinais de controle para atuadores. As informações coletadas por esses sensores podem ou não ser transmitidas pela Internet, possibilitando dessa forma a comunicação entre diferentes aparelhos e seus controladores, sendo eles de *hardware* ou *software* (ABREU, 2022).

No presente projeto foi utilizado o sensor de movimento PIR HC-SR505, apresentado na Figura 6, que utiliza um sensor piroelétrico para detectar movimentos, baseando-se na variação da radiação infravermelha emitida pelo corpo humano (ELETROGATE, 2024).

Figura 6. Imagem do sensor de movimento PIR HC-SR505. Fonte: (ELETROGATE, 2024).

3.2.3 Cabos e conectores

Para o desenvolvimento deste projeto, foram utilizados os cabos micro USB para alimentação da placa ESP32, cabos Jumper (Macho x Macho e Macho x Fêmea) - fios elétricos utilizados para realizar conexões elétricas entre componentes de um circuito, possibilitando que a eletricidade corra de forma segura ao longo do mesmo (VIDA DE SILICIO, 2023) – e por fim resistores de 1k/4w.

3.2.4 Protoboard

As Protoboards são compostas por uma placa de material isolante com vias de cobre e são comumente dispostas em formato de grade, para que os componentes eletrônicos possam ser postos ou soldados através de terminais. São utilizadas como bases para construir circuitos eletrônicos (NUNES, 2023).

3.3 Casa inteligente

O conceito de casa inteligente, que vem sendo amplamente associado à *IoT*, é definido por uma série de dispositivos inteligentes (como câmeras, sensores e assistentes de voz) interconectados para automatizar tarefas e garantir praticidade e comodidade ao dia a dia do usuário (INTELBRAS, 2023).

Além do conforto gerado pelas casas inteligentes, elas também podem garantir a segurança da residência através, por exemplo, de sensores e câmeras de segurança que podem visualizar e monitorar o ambiente, garantindo maior tranquilidade aos moradores (RR ÁUDIO & VÍDEO, 2024).

Também chama a atenção o constante crescimento das casas inteligentes em todo o mundo. Segundo uma pesquisa realizada pela *Fortune Business Insights*, é previsto que este mercado salte de US$ 72,3 bilhões em 2021 para US$ 163,24 bilhões até 2028, correspondendo a um crescimento anual de 12,3% no período citado (SANTOS, 2022).

3.4 Protocolo Message Queuing Telemetry Transport

O protocolo MQTT é utilizado para transporte de mensagens, pois possibilita a comunicação *M2M* (*Machine to Machine*, ou Máquina para Máquina no português brasileiro), é altamente popular para conectividade IoT e é executado no protocolo de rede TCP/IP. Seu formato é o Cliente/Servidor e utiliza o modelo *Publish/Subscribe*, no qual o Cliente pode realizar postagens ou receber informações enquanto o Servidor (*Broker*) administra o envio e recebimento dos dados (CRAVO, 2024). A Figura 7 tem como objetivo exemplificar de maneira prática o funcionamento do protocolo *MQTT*, exibindo as relações dos modelos *Publish/Subscribe* com o *broker*.

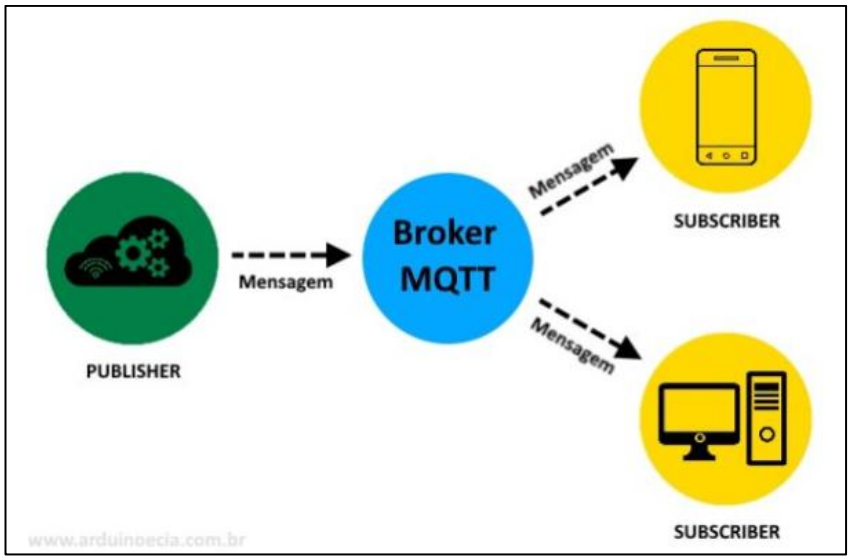


Figura 7. Exemplificação do funcionamento do protocolo MQTT. Fonte: (ARDUINO E CIA, 2022).

3.5 Broker Eclipse Mosquitto

O Eclipse Mosquitto é um *broker* de mensagens *open source* que implementa o protocolo MQTT. Para realizar a troca de mensagens, utiliza o modelo *Publish/Subscribe*, tornando-o adequado para dispositivos IoT como sensores, computadores, microcontroladores, entre outros (MOSQUITTO, 2024).

4. Metodologia

A metodologia escolhida para o projeto é a evolutiva incremental, que permitirá o desenvolvimento em incrementos pequenos e iterativos (Lima et al., 2023), ideal para adicionar a cada implemento as funcionalidades que serão escolhidas e implementadas no dispositivo. Além disso, a metodologia também permitirá que as pesquisas realizadas contribuam com o processo de entendimento do tema durante cada artigo, notícia reportagem, pesquisa e entre outros.

Durante o levantamento de requisitos, foram analisados projetos com propósitos semelhantes, com o objetivo de entender como dispositivos semelhantes podem se encaixar em temas relacionados a *IoT* e quais são suas principais funcionalidades. Pesquisas de programação, funcionalidades, instalação e entre outros poderão ser realizadas na Internet com o intuito de aproveitar demais funcionalidades para o dispositivo, caso disponíveis e/ou forem relacionadas e pertinentes ao tema do projeto. A análise de custo deverá ser realizada na última etapa do projeto.

O *Integrated Development Enviroment* (IDE) escolhido para programação do dispositivo foi o Arduino IDE, fornecido pelo próprio Arduino e que permite, a partir da codificação nas linguagens de programação C e C++, que programas sejam escritos, compilados e gravados na placa do dispositivo (MAKIYAMA, 2023).

Ademais, espera-se desenvolver um sistema conectado à rede utilizando o protocolo de rede MQTT com o objetivo de enviar notificações para o usuário uma vez que qualquer presença ou movimento seja identificado pelo protótipo.

5. Desenvolvimento

A seguinte seção tem como objetivo descrever todas as etapas do projeto, seus protótipos, funcionalidades e explicar como foram construídas.

5.1 Modelagem do funcionamento

Para o início do projeto, foi necessário construir um protótipo a partir da utilização do *hardware* – este especificado anteriormente neste artigo – para que a identificação da presença de movimento fosse detectada pelo sensor utilizado. O modelo inclui também um LED vermelho com a funcionalidade de acender caso o sensor detecte qualquer movimentação, com a finalidade de obter uma confirmação de que a movimentação está sendo identificada.

A Figura 8 apresenta as conexões entre o sensor de movimento, a placa de desenvolvimento e o LED vermelho. Os fios representados nas cores vermelho e preto estão sendo utilizados para energizar o dispositivo (pinos 3.3V e GND, respectivamente). Por sua vez, o fio amarelo é utilizado como a conexão que compartilha dados entre o sensor de movimento e a placa de desenvolvimento, e por fim o fio azul é utilizado para compartilhar dados entre a placa de desenvolvimento e o LED, fazendo-o acender quando o movimento é detectado pelo sensor de movimento. Já a figura 9 exibe o protótipo final – este estando de acordo com a representação apresentada na figura anterior.

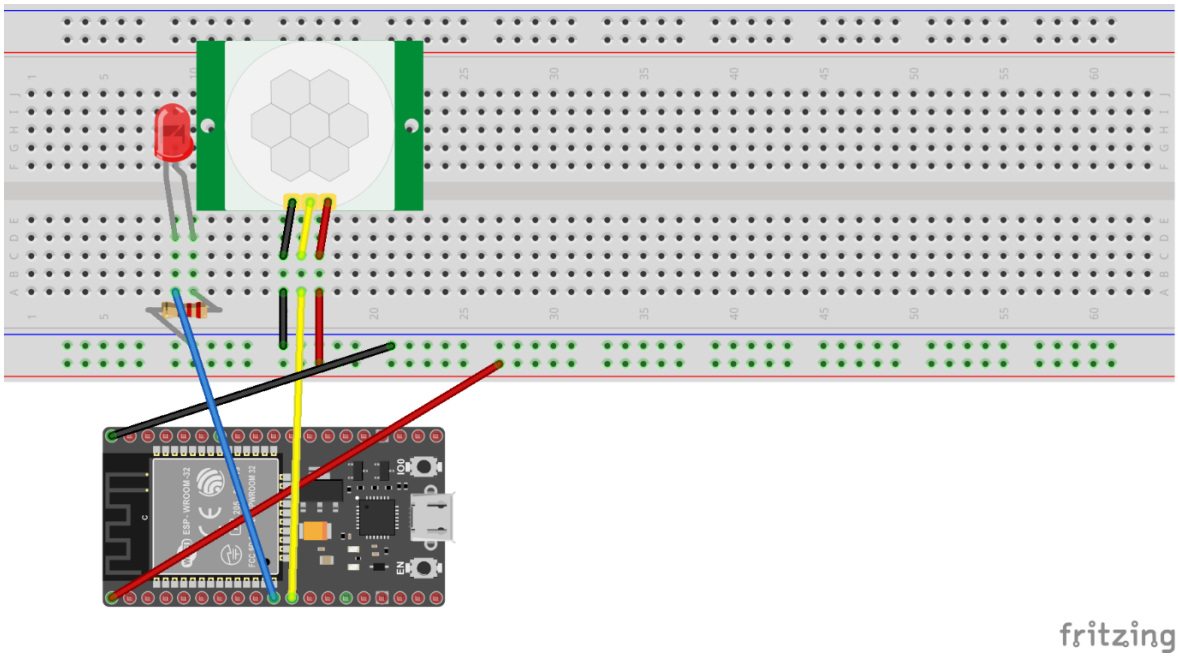


Figura 8. Diagrama de ligações do hardware. Fonte: elaborado pelo autor utilizando Fritzing.

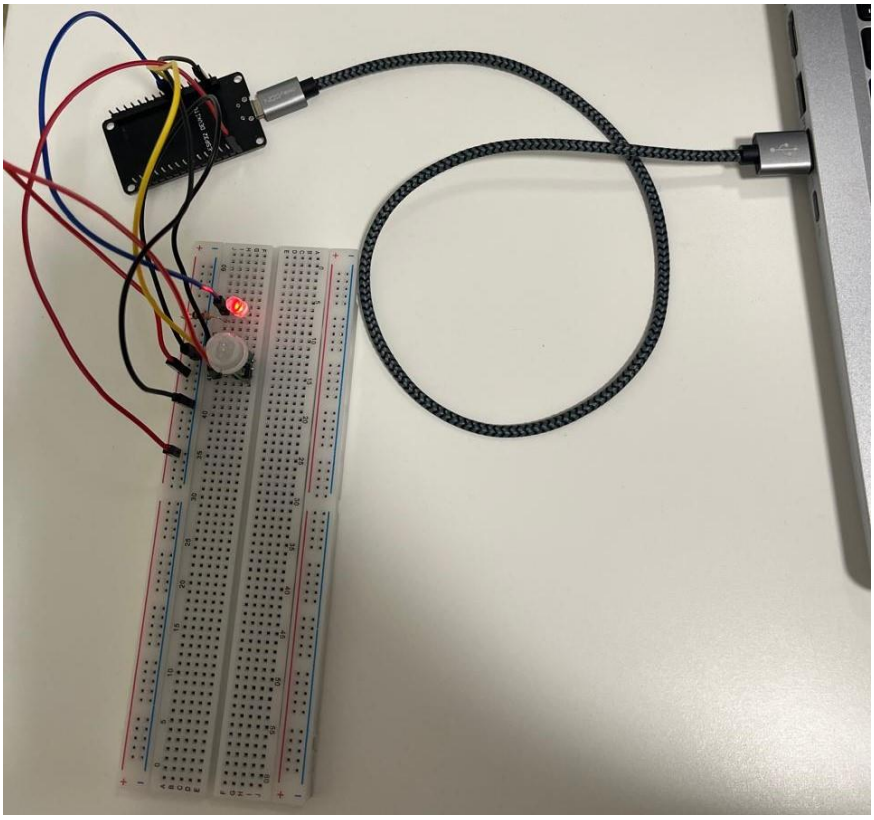


Figura 9. Protótipo final. Fonte: elaborado pelo autor.

5.2 Codificação

Para que o protótipo funcionasse corretamente, foi necessário desenvolver um código que permitisse as funcionalidades descritas na seção anterior. As subseções seguintes têm o objetivo de exibir e explicar o funcionamento do código em conjunto com o protótipo.

5.2.1 Bibliotecas utilizadas

Conforme citado anteriormente neste artigo, a placa controladora escolhida possui a tecnologia Wi-Fi integrada, permitindo a conexão com uma rede a partir da adição de uma biblioteca. Foram definidas então as bibliotecas WiFi.h e PubSubClient.h, essenciais para que o ESP32 possa se conectar a uma rede e se comunicar via protocolo MQTT.

5.2.2 Credenciais Wi-Fi e protocolo MQTT

Para a conexão com Wi-Fi e configurar o MQTT, foi necessário criar algumas variáveis (exibidas na Figura 10) com o objetivo de armazenar o SSID, senha, endereço de IP do servidor e definir qual seria o cliente utilizado e em qual tópico a mensagem seria publicada.

```
const char* ssid = "ID da rede";
const char* password = "Senha da rede";
const char* mqtt_server = "Endereço ipv4 da máquina";

const char* mqtt_client_id = "ESP32CustomClient";  // Define o Client ID
const char* mqtt_topic = "home/motion/FelipeTorres2001";  // Define o nome do tópico
```

Figura 10. Representação das credenciais Wi-Fi e configurações MQTT. Fonte: elaborado pelo autor.

Após definir as credenciais de conexão, foi criada a função setup_wifi (exibida na Figura 11(a)) para realizar a conexão do ESP32 com a rede Wi-Fi e a função connectMQTT (exibida na Figura 11(b)) para realizar a conexão com o *broker* MQTT, que também realiza a subscrição ao tópico caso necessário. Ambas as funções imprimem mensagens de status no serial monitor da IDE.

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Conectando a: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
    yield();
  }
  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
}
```

(a)

```
void connectMQTT() {
  while (!client.connected()) {
    Serial.print("Tentando conectar ao MQTT...");
    // Attempt to connect with custom Client ID
    if (client.connect(mqtt_client_id)) {
      Serial.println("conectado");
      mqttConnected = true;
      client.subscribe(mqtt_topic);  // Inscreve-se no tópico, se necessário
    } else {
      Serial.print("falhou, reason code=");
      Serial.print(client.state());
      Serial.println(" tente novamente em 5 segundos");
      delay(5000);  // Intervalo menor para evitar WDT timeout
      yield();  // Reinicia o watchdog de software durante o delay
    }
  }
}
```

(b)

Figura 11. Exibição das funções (a) setup_wifi e (b) connectMQTT. Fonte: elaborado pelo autor.

5.2.3 Função principal

Com todas as conexões realizadas, foi desenvolvida uma função principal para o código, que funciona como um *loop*. Para esta função, foram designadas as tarefas de manter a conexão com o *broker* MQTT ativa, verificar o status de detecção de movimento, gerenciar o LED de acordo com o tempo configurado, publicar mensagens no tópico escolhido e acender/apagar o LED. A função citada e suas funcionalidades são exibidas na Figura 12.

```
void loop() {
  // garante que o cliente continue conectado ao broker
  if (!client.connected()) {
    connectMQTT();
  }
  client.loop();  // mantém a conexão com o broker ativa

  // Processa o movimento detectado fora da interrupção
  if (motionDetected) {
    motionDetected = false;  // Reseta a flag
    Serial.println("Movimento detectado! Mensagem sendo enviada ao MQTT...");

    if (client.connected()) {
      client.publish(mqtt_topic, "Atenção: movimento identificado!");  // Publica a mensagem no tópico
      Serial.println("Mensagem enviada ao tópico MQTT.");
    } else {
      Serial.println("Falha ao enviar a mensagem, MQTT não conectado.");
    }
  }

  now = millis();

  if((digitalRead(led) == HIGH) && (motion == false)) {
    Serial.println("Movimento detectado!");
    motion = true;
  }

  // desliga o LED depois da quantidade de tempo definida em "timeSSeconds"
  if(startTimer && (now - lastTrigger > (timeSeconds * 1000))) {
    Serial.println("Não está sendo mais identificado nenhum movimento.");
    digitalWrite(led, LOW);
    startTimer = false;
    motion = false;
  }
}
```

Figura 12. Exibição da principal função do código. Fonte: elaborado pelo autor.

5.3 Configuração do broker Eclipse Mosquitto via CMD

O *broker* Eclipse Mosquitto implementa um servidor MQTT local na máquina utilizada para o projeto. A partir disso, é possível que outros dispositivos, como o ESP32, se conectem ao servidor e possam realizar a troca de dados.

Para a configuração no sistema operacional *Windows*, o primeiro passo foi iniciar o *broker* a partir do comando “net start mosquito” através do *prompt* de comando da máquina (popularmente conhecido como CMD). Também é possível realizar a verificação de quais portas estão ativas a partir do comando “netstat -a”, o qual permite checar se a porta 1883 – porta usada para o servidor local – está ativa e disponível para uso. Por fim, é possível também realizar a subscrição em um tópico utilizando o comando “m13osquito_sub -t [tópico] -h [endereço IP] -p [porta]” (MOSQUITTO, 2024).

5.4 Configuração do aplicativo iOS MyMQTT

Para acompanhar as mensagens de detecção de movimento em um dispositivo móvel, foi possível realizar a instalação do aplicativo MyMQTT através da App Store – loja de aplicativos para dispositivos iOS. O mesmo foi configurado a partir da utilização do endereço de IP, subscrição do tópico criado e porta 1883, disponibilizada pelo *broker* escolhido. A tela de configuração do aplicativo é mostrada na Figura 13.

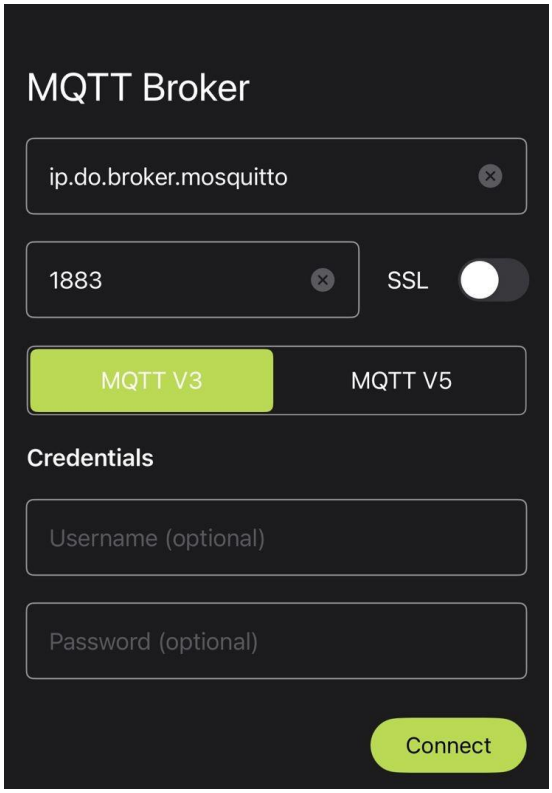


Figura 13. Exibição da configuração do aplicativo MyMQTT. Fonte: elaborado pelo autor.

5.5 Custo do Hardware

Para a construção do protótipo, foi necessário realizar a compra de alguns dispositivos para viabilizá-lo. Com o objetivo de não comprometer a proposta do projeto, isto é, construir um protótipo de baixo custo, foram comprados equipamentos também de baixo custo quando comparados, por exemplo, a um sistema de segurança residencial de nível profissional. A tabela 1 representa os dispositivos adquiridos, suas quantidades e valores, bem como o valor total gasto.

Tabela 1. Custos dos dispositivos de hardware. Fonte: (MERCADO LIVRE, 2024).

Dispositivo	Quantidade	Custo
Placa ESP32	1	R$ 39,99
Sensor PIR HC-SR505	1	R$ 17,40
Protoboard	1	R$ 13,10
Cabo Jumper	40	R$ 13,30
Led vermelho	5	R$ 8,99
Total	-	R$ 92,78

6. Resultados

Para entender o funcionamento e analisar os resultados do dispositivo, foi realizado um conjunto de testes, tendo como parâmetro a distância de detecção e o tempo em que o dispositivo permanece ativo antes de detectar qualquer movimento. As tabelas

2 e 3 indicam os sucessos e falhas de identificação através dos símbolos positivos em verde e negativos em vermelho, respectivamente.

Com o objetivo de padronizar os testes e criar cenários com as mesmas condições para cada testagem, para os testes de distância foram medidas distâncias correspondentes às metragens testadas, garantindo que nenhuma pessoa, animal ou qualquer tipo de movimentação pudesse influenciar e consequentemente afetar as testagens antes do momento ideal. Para os testes de tempo, o dispositivo foi colocado em um quarto trancado durante o tempo definido para cada testagem – este também sem qualquer presença que pudesse influenciar nos resultados dos testes.

Tabela 2. Testes de distância de detecção. Fonte: elaborado pelo autor.

1 metro					
Qtd. de testes	1	2	3	4	5
Resultado	✓	✓	✓	✓	✓
3 metros					
Qtd. de testes	1	2	3	4	5
Resultado	✓	✓	✓	✓	✓
5 metros					
Qtd. de testes	1	2	3	4	5
Resultado	✓	✓	✗	✗	✓
10 metros					
Qtd. de testes	1	2	3	4	5
Resultado	✗	✗	✗	✗	✗

Tabela 3. Testes de tempo em que o dispositivo permanece ativo antes da detecção. Fonte: elaborado pelo autor.

5 minutos					
Qtd. de testes	1	2	3	4	5
Resultado	✓	✓	✓	✓	✓
30 minutos					
Qtd. de testes	1	2	3	4	5
Resultado	✓	✓	✗	✓	✓
2 horas					
Qtd. de testes	1	2	3	4	5
Resultado	✗	✓	✓	✗	✓
10 horas					
Qtd. de testes	1	2	3	4	5
Resultado	✓	✓	✓	✗	✗

Observando a tabela que contém os testes de distância de detecção, é possível notar que o sensor funcionou conforme esperado – isto é, detectou movimento e enviou a mensagem para o *broker* – sem qualquer problema para as distâncias de 1 metro e 3 metros. Para a distância de 5 metros, o sensor apresentou funcionamento correto em 3 dos testes, sendo que 2 deles funcionaram apenas com movimentos bruscos, como saltos, por exemplo. Durante os testes de distância de

10 metros, notou-se que o sensor não foi capaz de identificar nenhuma movimentação, nem mesmo os aqui mencionados movimentos mais bruscos.

Analisando os testes de tempo em que o dispositivo permanece ativo antes da detecção, obteve-se um cenário similar ao anterior. Dessa forma, os testes com menor minutagem como o de 5 minutos apresentaram funcionamento ideal, enquanto conforme o tempo de testagem aumentou, os erros também aumentaram gradativamente. Vale ressaltar que durante esses testes, entende-se como erro a não detecção do movimento e/ou não envio da mensagem ao *broker* após o tempo estipulado para teste.

Por fim, com a finalidade de comprovar que o envio de mensagens ao *broker* estava sendo realizado de forma correta, foi feito o acompanhamento dos envios durante a realização dos testes através do *prompt* de comando da máquina e do aplicativo MyMQTT. O acompanhamento realizado após a configuração dos mesmos é exibido pelas figuras 14 e 15.

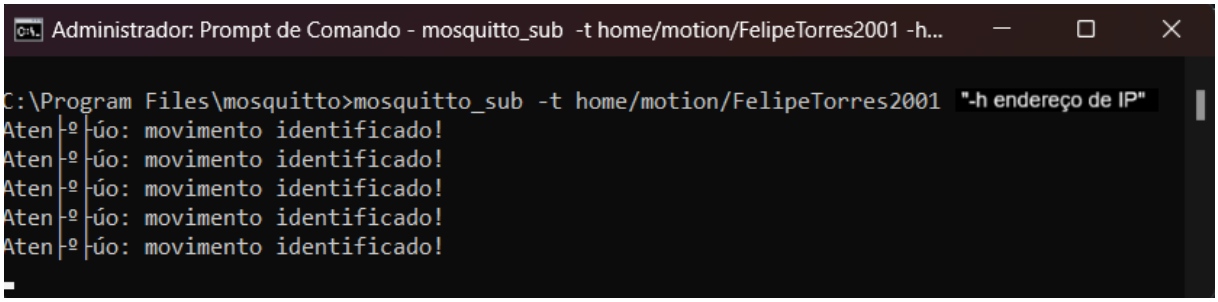


Figura 14. Exibição da notificação de detecção de movimento no prompt de comando. Fonte: elaborado pelo autor.

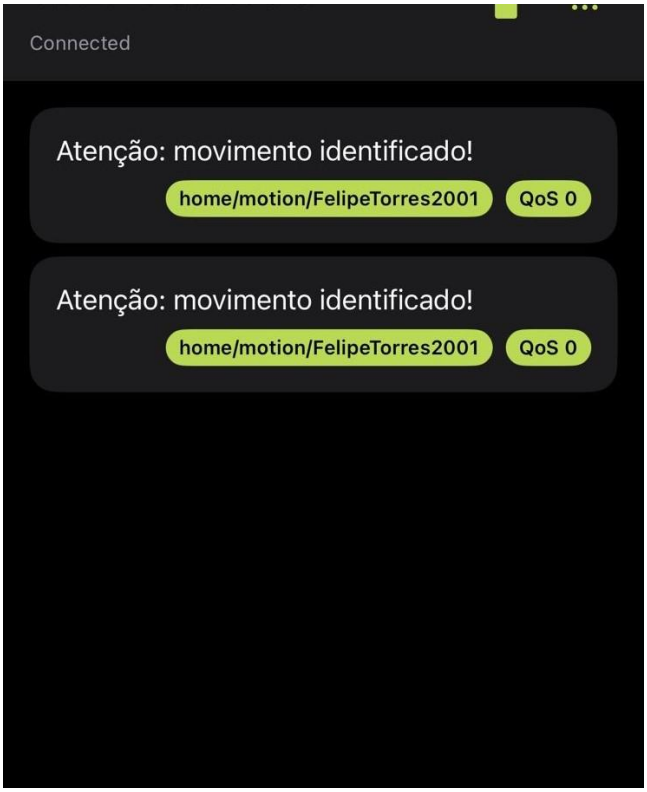


Figura 15. Exibição da notificação de detecção de movimento no aplicativo MyMQTT. Fonte: elaborado pelo autor.

7. Conclusão

Conforme apresentado ao longo deste artigo, a área de *IoT* e seus dispositivos vem crescendo de grande forma nos últimos anos, facilitando o acesso aos mesmos por usuários no Brasil e no mundo. Tais dispositivos podem ser usados para tarefas que visam auxiliar a segurança e até mesmo o conforto dos usuários.

Por este motivo, foram realizadas pesquisas e análises de diversos trabalhos correlatos, como os também citados nesse artigo anteriormente, através da metodologia escolhida para desenvolvimento do projeto. Após a análise, foram escolhidas também as tecnologias e ferramentas para viabilizar a construção do protótipo e foi desenvolvido o diagrama de ligações do *hardware*, com a finalidade de estruturar o protótipo. Com o diagrama completo, foi possível partir para a construção do protótipo, bem como o desenvolvimento do código a ser compilado na placa controladora ESP32, e por fim foi feita a configuração do *broker* Eclipse mosquito na máquina utilizada para o projeto.

Dessa forma, foi desenvolvido um protótipo Arduino de baixo custo para casa inteligente, com a finalidade de detectar qualquer movimentação em um determinado ambiente e enviar uma mensagem através de um *broker* que implementa o protocolo *MQTT*, visando a segurança do usuário.

Durante o desenvolvimento, o projeto contou com a aplicação de diversos conhecimentos ensinados ao longo do curso, sendo os que mais se destacam: Arquitetura de Sistemas, para o entendimento e criação da arquitetura do protótipo e seu relacionamento com o protocolo *MQTT*; Redes de Computadores, que auxiliou durante o uso e configuração do protocolo *MQTT*; e Linguagem de Programação, contribuindo diretamente na construção do código desenvolvido.

Ao fim do projeto, ficou claro que o dispositivo apresenta bom funcionamento e é capaz de realizar sua proposta, em termos gerais conseguindo conectar-se à rede, ao *broker*, detectar movimentações e realizar o envio de mensagens. Porém, nota-se também que o dispositivo apresenta algumas limitações – em sua maioria causadas pelas configurações e capacidades do sensor PIR HC-SR505 – sendo as principais o limite para distância de detecção e o tempo em que o dispositivo permanece conectado antes de detectar qualquer movimento, apresentando erros conforme esses dois parâmetros aumentam. Ainda assim, quando comparamos o funcionamento e os custos do hardware utilizado a um sistema comercial profissional, entende-se que o dispositivo é viável e acessível.

Por fim, para aumentar a eficiência do dispositivo e a experiência do usuário, é possível citar como trabalhos futuros o desenvolvimento de soluções que possam incrementar e ampliar as funcionalidades do trabalho aqui desenvolvido, por exemplo através da utilização de dados coletados pelo dispositivo. Também é possível citar que o desenvolvimento de uma API que pudesse coletar esses dados, fornecer uma interface mais responsiva e amigável e com mais funcionalidades poderia agregar como trabalho futuro visando o que foi desenvolvido nesse projeto.

8. Referências

ABREU, Leandro. **Sensores IoT: como funcionam, quais são e como beneficiam a sua vida e as empresas**. Rock Content, 2022. Disponível em: https://rockcontent.com/br/blog/sensores-iot/. Acesso em 09 jul. 2024.

AMAZON. **O que é IoT (Internet das Coisas)?** 2024. Disponível em: https://aws.amazon.com/pt/what-is/iot/. Acesso em 05 maio 2024.

ARDUINO E CIA. **Automação residencial usando MQTT**. 2022. Disponível em: https://www.arduinoecia.com.br/automacao-residencial-usando-mqtt-wio-terminal-esp8266/. Acesso em 02 nov. 2024.

CANDIDO, Marcelo Sousa. **Automação Residencial utilizando uma Plataforma Open-Source.** Hortolândia, 2020. Disponível em: https://hto.ifsp.edu.br/cloud/s/pPYDMAR2HHibSyM. Acesso em 01 nov. 2024.

COSTA, Eduardo Porfiro. **Gerenciador de Coleta de Dados para Redes de Sensores sem Fio**. Hortolândia, 2019. Disponível em: https://hto.ifsp.edu.br/institucional/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2019/TCC_Eduardo_Porfiro_Costa.pdf. Acesso em 01 nov. 2024.

CRAVO, Edilson. **Protocolo MQTT: como funciona, dicas e informações importantes.** Kalatec, 2024. Disponível em: https://blog.kalatec.com.br/protocolo-mqtt/. Acesso em 02 nov. 2024.

DA SILVA, Renata Gomes. **Sistema Domiciliar de Cuidados em Saúde: Mecanismo de Detecção de Quedas de Pessoas.** Hortolândia, 2018. Disponível em: https://hto.ifsp.edu.br/portal/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2018/TCC_RenataGomesdaSilva_HT1520792.pdf. Acesso em 04 nov. 2024.

ELETROGATE. Mini Sensor de Movimento Presença PIR HC-SR505. 2024. Disponível em: https://www.eletrogate.com/mini-sensor-de-movimento-presenca-pir#:~:text=O%20Sensor%20PIR%20HC%2DSR505,pela%20radia%C3%A7%C3%A3o%20do%20corpo%20humano. Acesso em 15 jul. 2024.

HABITISSIMO. **Orçamento para instalação de sistema de alarme residencial ou comercial**. 2024. Disponível em: https://www.habitissimo.com.br/orcamentos/alarmes#:~:text=O%20valor%20a%20ser%20investido,materiais%20e%20m%C3%A3o%20de%20obra. Acesso em 01 mar. 2024.

INTELBRAS. **Tudo o que você precisa saber sobre Casa Inteligente e suas tecnologias.** 2023. Disponível em: https://blog.intelbras.com.br/casa-inteligente/. Acesso em 02 jul. 2024.

IOT SNAPSHOT. **El futuro de IOT**. 2022. Disponível em: https://resources.logicalis.com/iot-snapshot-2022#formulario. Acesso em 22 mar. 2024.

LEITE, João Paulo Coelho. **Sistema de gerenciamento remoto de dispositivos residenciais.** Hortolândia, 2022. Disponível em: https://hto.ifsp.edu.br/cloud/s/tP2TPYijgQWZEEX. Acesso em 03 nov. 2024.

LIMA, Caio Ryann Conceição; CARR, Caroline Nunes; MARGARIDO, Jean Jerome Pereira; DA SILVA, Ryan Dias. **O modelo incremental no desenvolvimento de software: uma maneira estruturada e interativa de entregar produtos de qualidade.** Disponível em: https://rsdjournal.org/index.php/rsd/article/view/40934. Acesso em 15 jun. 2024.

MAKIYAMA, Marcio. **Placa ESP32: Descubra o que é, para que serve e muito mais!** Victor Vision, 2023. Disponível em: https://victorvision.com.br/blog/placa-esp32/#:~:text=Quais%20as%20vantagens%20do%20ESP32%20em%20rela%C3%A7%C3%A3o%20ao%20Arduino%20Uno%3F,-A%20ESP32%20e&text=Com%20um%20processador%20dual%2Dcore%20de%2032%20bits%20e%20clock,outros%20dispositivos%20e%20a%20Internet. Acesso em 27 jun. 2024.

MEIO & MENSAGEM. **Internet das Coisas: o que é, como funciona e como é utilizada.** São Paulo, 2022. Disponível em: https://www.meioemensagem.com.br/proxxima/internet-das-coisas#:~:text=Sua%20primeira%20men%C3%A7%C3%A3o%20foi%20em,caso%20da%20conectividade%20sem%20fio. Acesso em 07 jun. 2024.

MOSQUITTO. **Eclipse Mosquitto™ An open source MQTT broker.** 2024**.** Disponível em**:** https://mosquitto.org/. Acesso em 25 out. 2024.

NUNES, Caio. **Para que serve uma Protoboard.** Fábrica de Bolso, 2023. Disponível em: https://fabricadebolso.com.br/para-que-serve-uma-protoboard-2/#:~:text=Para%20resumir%20uma%20protoboard%20%C3%A9,presos%20por%20meio%20de%20terminais. Acesso em 22 jun. 2024.

PACETE, Luiz Gustavo. **Montar uma casa inteligente: quanto custa e do que você precisa?** Forbes, 2022. Disponível em: https://forbes.com.br/forbes-tech/2022/03/quanto-custa-montar-uma-casa-inteligente-no-brasil/. Acesso em 22 fev. 2024.

RR ÁUDIO & VÍDEO. **Segurança x sistema para casa inteligente: entenda a relação!** São Paulo, 2024. Disponível em: https://rraudioevideo.com.br/sistema-para-casa-inteligente/#pesquisa-blog-categorias. Acesso em 05 jul. 2024.

SANTOS, Francisco. **Mercado da automação residencial deve crescer 12,3% ao ano até 2028**. 2022. Disponível em: https://portalnorte.com.br/noticias/2022/12/31/mercado-automacao-residencial-crescer-2028/. Acesso em: 05 jul. 2024.

SAP - Desenvolvimento de Programas para Análise de Sistema. **O que é Internet das Coisas (IoT)?** 2024. Disponível em: https://www.sap.com/brazil/products/artificial-intelligence/what-is-iot.html. Acesso em 13 jul. 2024.

USINAINFO. **ESP32 NodeMCU Iot com WiFi e Bluetooth - 30 Pinos**. 2023. Disponível em: https://www.usinainfo.com.br/nodemcu/esp32-nodemcu-iot-com-wifi-e-bluetooth-30-pinos-5147.html. Acesso em 17 maio 2024.

VIDA DE SILICIO. **Protoboard e Jumpers.** 2023. Disponível em: https://www.vidadesilicio.com.br/categoria/prototipagem-e-ferramentas/protoboard-e-jumpers/. Acesso em: 07 jun. 2024.

Cópia de documento digital impresso por Michele Barion (1958699) em 6 de Março de 2025 às 16:03.

Documento Digitalizado Público

Artigo Final de TCC do Felipe M. Torres

Assunto: Artigo Final de TCC do Felipe M. Torres
Assinado por: Leandro Ledel
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Leandro Camara Ledel**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 20/12/2024 16:08:31.

Este documento foi armazenado no SUAP em 21/02/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1948517
Código de Autenticação: f3106229bd