PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Aplicação da IoT na automação de um porteiro eletrônico

Douglas Silva de França

GOIÂNIA
2022

DOUGLAS SILVA DE FRANÇA

Aplicação da IoT na automação de um porteiro eletrônico

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientadora: Prof.ª Ma. Angélica da Silva Nunes.

GOIÂNIA

2022

DOUGLAS SILVA DE FRANÇA

Aplicação da IoT na automação de um porteiro eletrônico

Este Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Engenharia de Computação, e aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, em ___/___/____.

_____

Prof. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Orientadora: Prof.ª Ma. Angélica da Silva Nunes

_____

Prof. Me. Rafael Leal Martins

_____

Prof. Me. Carlos Alexandre Ferreira de Lima

GOIÂNIA
2022

DEDICATÓRIA

A minha querida vó Antônia mesmo estando longe sempre me dando conselhos e apoiando para realizar esse sonho.

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me dado força, saúde e determinação para superar os obstáculos da vida.

Agradeço aos meus pais, Francisco e Carmem, e as minhas irmãs por me apoiarem desde sempre do meu lado, em especial minha mãe pelos conselhos e ensinamento.

À minha professora e orientadora acadêmica, Ma. Angélica da Silva Nunes, pelo apoio e dedicação na orientação deste trabalho.

Ao meu amigo Higor por todo suporte e auxílio em vários momentos na construção deste trabalho.

“Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento”

Albert Einstein

RESUMO

Este trabalho de conclusão de curso apresenta um estudo teórico e prático sobre automação residencial com uso de *Internet of Things*, *Internet* das Coisas (IoT) que tem por objetivo desenvolver um protótipo de automação residencial em cima do modelo 3D da maquete, a criação de um porteiro eletrônico que proporciona ao usuário a possibilidade de controlar e monitorar a abertura e o fechamento do portão remotamente através de plataforma *Web*. Em seguida é feita toda a documentação de todas as etapas de desenvolvimento do projeto, seguida do modelo representativo da maquete. Foi utilizado o microcontrolador ESP32 e dispositivos eletrônicos para montar um conjunto de comunicação e leitura de sensor integrado, possibilitando a criação do sistema do porteiro eletrônico. O gerenciamento de controle se dá pelo uso da plataforma *Web* que permite usuário visualizar e consultar históricos de acionamento e de presença capturada pelo sensor no portão.

Palavras-Chaves: Porteiro Eletrônico, Microcontrolador ESP32, IoT.

ASTRACT

This course completion work presents a theoretical and practical study on home automation using the Internet of Things, Internet of Things (IoT) which aims to develop a home automation prototype based on the 3D model of the model, the creation of a electronic intercom that gives the user the possibility to control and monitor the opening and closing of the gate remotely through a Web platform. Then all the documentation of all stages of development of the project is done, followed by the representative model of the model. The ESP32 microcontroller and electronic devices were used to assemble a set of communication and integrated sensor reading, enabling the creation of the electronic intercom system. Control management is carried out using the Web platform that allows the user to view and consult histories of activation and presence captured by the sensor at the gate.

Keywords: Electronic Intercom, ESP32 Microcontroller, IoT.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

A/D	*Analog to Digital* - Analógico para Digital
APIs	*Application Programming Interface* - Interface de Programação de Aplicação
CSV	*Comma Separated Values* - Valores Separados Por Vírgulas
CPU	*Central Processor Unit* - Unidade de Processador Central
D/A	*Digital to Analog* - Digital para Analógico
DML	*Data Manipulation Language* - Linguagem de Manipulação de Dados
DDL	*Data Definition Language* - Linguagem de Definição de Dados
LED	*Light Emitting Diode* - Diodo Emissor de Luz
EF	*Physical Entities* - Entidades Físicas
EV	*Virtual Entities* - Entidades Virtuais
EXI	*Efficient XML Interchange* - Intercâmbio XML Eficiente
FPGA	*Field Programmable Gate Array* - Matriz de Portas Programáveis em Campo
GND	*Graduated Neutral Density* - Filtro Graduado de Densidade Neutra
GHz	*Giga Hertz*
GPIO	*General Purpose Input/Output* - Entrada/Saída de Uso Geral
HTTP	*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto
I/O	*In/Out* - Dentro/Fora
I2C	*Inter-Integrated Circuit* - Circuito Inter-Integrado
IBSG	*Internet Business Solutions Group* - Grupo de Soluções de Negócios da *Internet*
IDE	*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado
IEEE	*Institute of Electrical and Electronics Engineers* - Instituto de Engenheiros Eletricistas e Eletrônicos
IoT	*Internet of Things* - *Internet* das Coisas
IP	*Internet Protocol* - Protocolo de *Internet*
ITU-T	*International Telecommunication Union* - União Internacional de Telecomunicações
M2M	*Machine-to-Machine* - Máquina a Máquina

MHz	*Mega Hertz*
MM	*Marketsand Markets* – Mercados e Mercados
MQTT	*Message queue telemetry transport* - Transporte de Filas de Mensagem de Telemetria
NFC	*Near Field Communication* - Comunicação de Campo Próximo
OWL	*Web Ontology Language* - Linguagem de Ontologia da *Web*
PIR	*Passive Infrared* - Infravermelho Passivo
PWM	*Pulse Width Modulation* - Modulação de Largura de Pulso
PVC	*Polyvinyl Chloride* - Policloreto de Vinila
RFID	*Radio Frequency Identification* - Identificação por Radiofreqüência
RSSF	*Wireless sensor network* - Redes de Sensores sem Fio
SGBD	*Database Management Systems* - Sistemas de Gerenciamento de Bancos de Dados
SPI	*Serial Peripheral Interface* - Interface Periférica Serial
TCP	*Transmission Control Protocol* - Protocolo de Controle de Transmissão
TCP/IP	*Transmission Control Protocol/Internet Protocol* - Protocolo de Controle de Transmissão/Protocolo de *Internet*
URL	*Uniform Resource Locator* - Localizador Uniforme de Recursos
USB	*Universal Serial Bus* - Barramento Serial Universal
Wi-Fi	*Wireless Fidelity* - Fidelidade sem Fio
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A *Internet*, desde sua origem em 1969, tem um crescimento e evolução considerada de alta expressão, o seu uso está cada vez mais diversificado. Estima-se que, em dezembro de 2008, a *Internet* já havia ultrapassado a marca de um bilhão e meio de usuários usando a tecnologia, o grande avanço da pelo crescimento acelerado de informações ao redor do mundo (CLARK et al., 2021).

Segundo Clark et al. (2021), este grande crescimento também reflete a evolução no Brasil. No entanto, apesar da expansão do uso da rede indicar aprovação e aceitação por parte dos usuários, algumas limitações começam a surgir para atender novos requisitos como segurança, mobilidade e qualidade de serviço. Tais limitações se devem à ossificação do projeto inicial da *Internet*, que não permite grandes modificações no núcleo da rede.

Com o objetivo de interligar os ambientes acadêmicos e de pesquisas, com seu crescimento exponencial, a *Internet* foi expandida para a área comercial, aumentando ainda mais o seu uso com o surgimento da forma de busca *web* e das redes sociais.

Esta forma de utilização da *Internet* visa interligar aparelhos de uso cotidiano, executando assim uma comunicação entre coisas, objetos e aparelhos, possibilitando maior automatização do dia a dia, por meio do uso de um dispositivo eletrônico, por exemplo, aparelho celular (CLARK et al., 2021).

O crescimento exponencial da *Internet* possibilita o incremento de automatizar de forma inteligente em diversos setores, como: Prédio, Casa, Cidade e Indústria.

A *Internet* e a automação são recorrentes dos acréscimos de processamento, memória e rede de comunicação nos objetos envolvidos. A *Internet of Things*, *Internet* das Coisas (IoT) projeta uma nova transformação digital, conectando dispositivos automatizados, incrementando valores de negócios, redefinindo organizações e gerando enorme quantidade de oportunidades (MAGRANI, 2018).

Para Atzori et al. (2010), a IoT é como uma estrutura de rede global e dinâmica que possibilita a autoconfiguração, levando-se em consideração protocolos padronizados de comunicação, no qual seus componentes físicos e virtuais possuem identidades, desde então usando *interfaces* inteligentes ligadas a *Internet*.

A importância da IoT e seus componentes é que são capaz de interagir entre si, trocando informações em diferentes ambientes, além de reagir automaticamente aos eventos

do mundo real, de modo que não haja eventual intervenção direta do ser humano em sua estrutura.

A tecnologia IoT tem como objetivo conectar diferentes *softwares*, dispositivos eletrônicos, máquinas industriais, “coisas” no termo geral, fazendo uso de métodos dinâmicos, além do uso de sensores, atuadores, nanotecnologia, *wireless*, com vista a comunicação remota dos componentes conectados via *Internet* (YOUSIF et al., 2021).

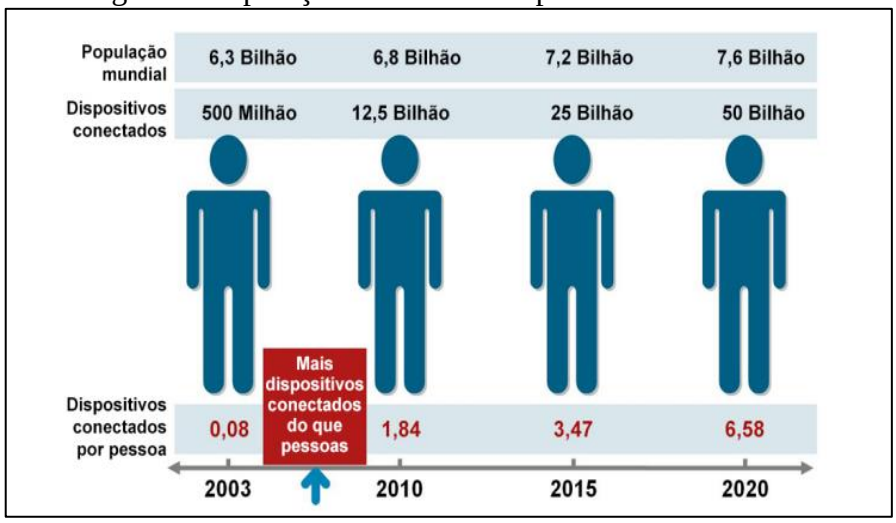
Ashton (2009), acredita que IoT tem o potencial de mudar o mundo se manuseada da maneira correta, assim como a *Internet* mudou e ainda continua trazendo inúmeras outras mudanças com sua evolução.

A proposta da IoT é concernente a que os objetos estão e serão cada vez mais conectados entre si, tendo à disposição uma estrutura global de objetos interconectados, chamados “inteligentes”, que provém de uma *interface* de conexão em forma de serviços facilitados pela interação através da *Internet* (YOUSIF et al., 2021).

A Cisco em 2011, *Internet Business Solutions Group*, Grupo de Soluções de Negócios da *Internet* (IBSG), estima que o ano de 2009, foi considerado como o surgimento da IoT. Desde então, havia mais objetos conectados que a própria população mundial, considerando os produtos eletrônicos, como computadores pessoais, *smartphones* e *tablets*.

Ilustrado na Figura 1, observa-se o número de dispositivos por habitante no mundo em função do tempo.

Figura 1: População Mundial x Dispositivos Conectados



Fonte: CISCO IBSG, 2011

A importância do investimento em IoT decorre das perspectivas de lucro positivo no setor. Ainda segundo a Cisco (2011), estima-se que a IoT pode adicionar 352 bilhões de

dólares à economia brasileira até o final de 2022. O grande avanço e previsões como essa, fazem com que diversos setores nas empresas possam se beneficiar dessa tecnologia.

A consultoria internacional McKinsey Global, prevê um impacto econômico da IoT consideravelmente grande, podendo chegar de USD 3,9 trilhões a USD 11,1 trilhões por ano em 2025, tendo em consideração 11% da economia mundial (MANYIKA. J. Et al. 2015).

Essa previsão depende de uma série de fatores, como a diminuição dos custos da tecnologia e o aumento do nível de aceitação em ambas as partes consumidores e trabalhadores.

O presente trabalho visa abordar a utilização de IoT e objetos da vida cotidiana quando conectados à *Internet* possam agir de modo inteligente e sensorial, como por exemplo, na automação residencial. Essa área vem revolucionando e passando por mudanças marcadas pela convergência de tecnologias avançadas digitais e físicas com grande escala em todo segmento nacional e internacional.

1.1 Justificativa

Justifica-se estudar este assusto pela necessidade de proporcionar facilidade e praticidade ao usuário que busca uma rotina do dia a dia acessível, confortável e segura. Em relação à segurança da residência, a possibilidade de controlar de forma remota o portão eletrônico de uma residência.

Os porteiros eletrônicos já representaram grande avanço no conforto e na segurança das residências da população, com capacidade de identificar e até atender à porta da residência sem a necessidade de abri-la. Tendo em vista a sociedade tem necessidade de mais segurança e conforto, é possível melhorar o processo, possibilitando o monitoramento direto por uma plataforma *web*.

Uma série de vantagens pode ser observada como identificar pessoas desconhecidas, a necessidade da presença física do morador para abrir o portão, facilidade no agendamento de visita para outra pessoa que está presente na residência, agendar horário com entregador para deixar encomendas a um terceiro quando não estiver na residência. Uma vez tendo o acesso a plataforma *Web*.

Embora existam algumas automações residenciais voltadas para aplicação remota, o modo de realizar tarefa com a presença da pessoa física no local, proporciona a necessidade de realizar este trabalho com uso do microcontrolador ESP32 integrado aos dispositivos de comunicação e acionamento do portão eletrônico de forma presencial e remoto.

1.2 Questão de pesquisa

Tendo em vista a importância de controlar um portão de forma remoto, facilita a troca de dados de maneira rápida e acessível, este projeto de pesquisa propõe responder a seguinte questão:

Como proporcionar acessibilidade ao usuário na realização de monitorar e controlar abertura e fechamento remotamente do portão eletrônico, através de uma plataforma *web*?

1.3 *Objetivo geral*

Este projeto tem como objetivo realizar o controle de abertura e fechamento do portão eletrônico remotamente através da utilização de IoT.

1.4 *Objetivos específicos*

- Apresentar os fundamentos e conceitos de IoT;
- Apresentar os conceitos e características do microcontrolador ESP32;
- Estudar e apresentar conceitos de automação residencial;
- Apresentar conceitos de protocolos de comunicação e de servidor *Web*;
- Construção de uma maquete como modelo representativo;
- Criação do protótipo do portão eletrônico em cima do modelo representativo;
- Desenvolvimento do sistema do microcontrolador ESP32, para gerenciar e controlar abertura e fechamento do portão eletrônico;

1.5 Metodologia

Para alcançar os objetivos do presente trabalho, foi efetuada a pesquisa bibliográfica, que se refere ao estudo de artigos, livros, teses e outras publicações que estão comumente disponibilizadas por editoras e indexadas (WAZLAWICK, 2014).

Quanto a natureza deste trabalho, trata-se de um resumo de assunto, pois apenas trabalha e expande assuntos de trabalhos feitos e publicados, que se tornaram a motivação para a criação deste trabalho (WAZLAWICK, 2014).

Este resumo de assunto visa sistematizar uma área de conhecimento específico, mostrando sua evolução e sua importância.

Quanto aos seus objetivos, trata-se de uma pesquisa descritiva, pois com este trabalho buscar-se obter dados mais consistentes sobre Automação de Porteiro Eletrônico,

descrevendo os fatos como são. A pesquisa é feita através de um estudo de levantamento de dados para análise.

Quanto aos seus procedimentos técnicos, trata-se de uma pesquisa experimental, caracterizada pela manipulação e obtenção de variáveis de um ambiente, já no caso o ambiente físico simulado de uma automação do porteiro eletrônico.

Nesta pesquisa experimental, as variáveis do porteiro eletrônico, relés, micro servo motor, fechadura eletrônica e sensor de movimento são observados, analisadas e controladas através de testes pelo pesquisador, cuja medição de controle, monitoramento leva a uma determinada conclusão através da análise dos resultados estatísticos e generalizáveis.

1.6 Estrutura da monografia

A estrutura do presente trabalho consiste em sete capítulos que tratam os seguintes assuntos:

No capítulo 1 deste trabalho apresenta-se a introdução sobre os assuntos abordados, sendo justificativa, objetivos gerais, específicos e metodologia.

No capítulo 2 aborda os fundamentos teóricos sobre IoT, o uso da tecnologia 5G, dados em nuvem, conectividade de periféricos ligados à IoT e automação residencial.

No capítulo 3 é abordada a introdução sobre microcontroladores a sua importância na automação residencial, protocolos de comunicação, protocolos MQTT, conexão *Wi-fi* e controle PWM.

No capítulo 4 apresenta a descrição e funcionalidade dos componentes elétricos, e orçamento dos componentes.

No capítulo 5 descreve a implementação de um porteiro eletrônico de forma geral, começando pelo planejamento até a construção, descrição da maquete do portão eletrônico, definição do microcontrolador esp32, componentes eletrônicos com respectiva ligação e substituição de componentes.

A presenta o desenvolvimento do sistema do microcontrolador com servidor *Web* onde serão armazenados os dados coletados a criação de uma plataforma *Web* de controle e monitoramento para visualizar os históricos de acesso, e desenvolvimento do servidor *Web*.

No capítulo 6 descreve as dificuldades encontradas, a realização dos testes, o funcionamento de todo o sistema para validar a segurança do porteiro eletrônico.

O capítulo 7 apresenta as conclusões obtidas através do desenvolvimento do trabalho prático e teórico, e as sugestões de trabalhos futuros para quem for da continuidade desenvolvimento do trabalho.

2 IoT

A IoT popularmente conhecida como *Internet* das coisas é um conceito em que o mundo real e o mundo virtual se conectam entre si, criando a possibilidade de conexão ao mundo inteligente nos diferentes segmentos da sociedade (DIAS, 2016).

A IoT é classificada como um conjunto de tecnologias e protocolos associados, que permitem a conexão com objetos através de uma rede de comunicação que são identificados e controlados pela à rede, (CAVALLI, 2016).

Para transformar e enriquecer a maneira como percebemos e interagimos com a realidade, se percebe que a visão da IoT muitas vezes pressupõe dispositivos distribuídos capilarmente para os quais as *Wireless sensor network*, Redes de Sensores sem Fio (RSSF) continuam a desempenhar um papel importante como uma das principais tecnologias habilitadoras desde o início do paradigma da IoT, embora seu avanço e escalabilidade percorra de forma infinita.

A IoT foi associada ao uso da tecnologia RFID. Desde então, por volta de 2005, o termo passou a popularizar no mercado, em seguida a IoT apresentou relação com *Wireless* Sensor Networks, Redes de Sensores Sem Fio (RSSF).

Ashton (2009) cofundador e diretor executivo do Auto-ID Center, apresentou numa palestra uma nova ideia do sistema RFID, identificação por radiofrequência, para a rastreabilidade do produto industrial na cadeia de suprimentos, com o objetivo de expressar o avanço da tecnologia para evoluir de forma inteligente no processo produtivo.

Um sistema de RFID é composto, por uma antena, um transceptor, que faz a leitura do sinal e transfere a informação para um dispositivo leitor, que oferece uma combinação de vários fatores, como ondas eletromagnéticas para acessar dados armazenados, identidade do objeto e outras informações contidas na etiqueta RFID.

A etiqueta RFID é composta de um *microchip*, que armazena os dados e garante uma transmissão e recebimento de ondas eletromagnéticas (DIAS, 2016).

Segundo minerva, B. R. et al. (2015) a etiqueta RFID responde a um sinal de frequência determinada com resposta “0” ou “1”. A realização da aplicação em uma loja comercial, com objetivo de identificar um possível roubo, foi descrita da seguinte maneira.

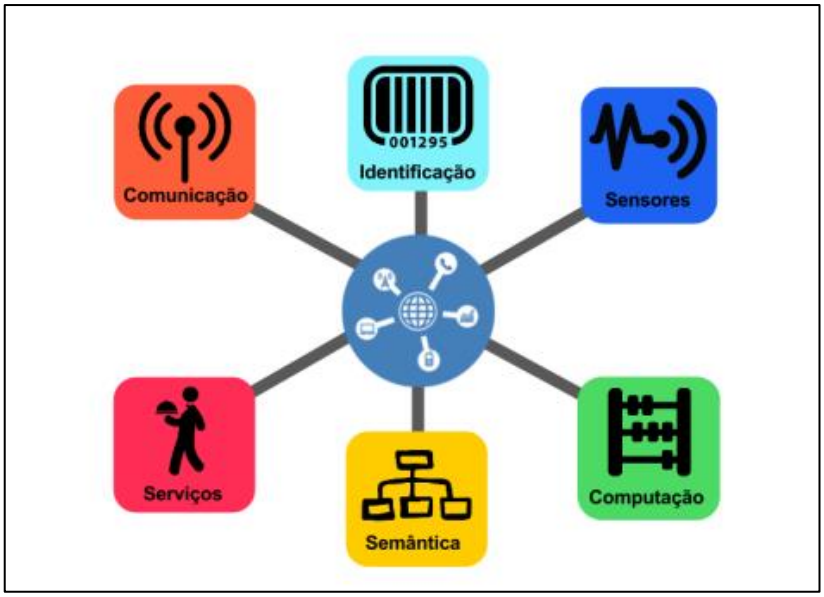
O cliente ao aproximar do caixa a etiqueta RFID do produto é identificada, com isso possibilita a passagem pelas portas sem acionar o alarme. Caso a etiqueta do produto não

seja identificada no caixa, o alarme será disparado pelo sistema antirroubo, quando for passar pelas portas de saída da loja.

2.1 Fundamentos Básicos que Compõem IoT

A IoT pode ser vista como características e combinação de diversas tecnologias associadas, as quais são complementares no sentido de viabilizar a integração dos objetos em cenários diferentes, físico e virtual, como é visto na Figura 2.

Figura 2: Fundamentos que Compõem IoT



Fonte: SANTOS et al., 2016

Santos et al., (2016) comenta cada um dos fundamentos que compõem IoT, que são relacionados a seguir:

- **Identificação:** é uma das partes mais importantes, visto que é primordial identificar os objetos unicamente para conectá-los à rede via *Internet*. Por exemplo, tecnologias como *Radio Frequency Identification* - Identificação por Radiofrequência (RFID), *Near Field Communication*, Comunicação de Campo Próximo (NFC) e endereçamento *Internet Protocol*, Protocolo de *Internet* (IP) podem ser empregados para realizar a identificação dos objetos em determinado local;
- **Sensores/Atuadores:** o objetivo dos sensores é coletar informações sobre o contexto em que os objetos se conectam e, em seguida, armazena e encaminha esses dados para *data warehouse, clouds* ou centros de armazenamento. Já os

atuadores podem manipular o ambiente ou reagir de acordo com os dados lidos e tratados;

- **Comunicação:** responsável pela transferência de dados dos dispositivos para as aplicações e para outros dispositivos, ou seja, receber e/ou enviar mensagens, bem como trazer instruções das aplicações para os dispositivos, de maneira eficiente e segura, segundo a *(International Telecommunication Union* (ITU-T). As diversas técnicas usadas para conectar objetos inteligentes desempenha papel importante baseando no consumo de energia dos objetos, por outro lado pode ser um fator crítico, com base das tecnologias usadas o Wi-Fi, o *Bluetooth* ou o RFID;
- **Computação:** inclui a unidade de processamento como, por exemplo, microcontroladores, processadores e *Field Programmable Gate Array,* Matriz de Portas Programáveis em Campo (FPGAs), responsáveis por executar algoritmos locais nos objetos inteligentes;
- **Serviços:** diversas classes de serviços a IoT pode prover, dentre elas, destacam-se os "Serviços de Identificação", responsáveis por mapear *Physical Entities*, Entidades Físicas (EF) de interesse do usuário em *Virtual Entities*, Entidades Virtuais (EV) como, por exemplo, a temperatura de um local físico em seu valor, coordenadas geográficas do sensor e instante da coleta;
    - Serviços de Colaboração e Inteligência - que agem sobre os serviços de agregação de dados para tomar decisões e reagir de modo adequado a um determinado cenário;
    - Serviços de Ubiquidade - que visam prover serviços de colaboração e inteligência em qualquer momento e qualquer lugar em que eles sejam necessários;
    - Serviços de Agregação de Dados - que coletam e sumarizam dados homogêneos/heterogêneos obtidos dos objetos inteligentes;
- **Semântica:** inclui a habilidade de extração de conhecimento dos objetos na IoT. Trata da descoberta de conhecimento e uso eficiente dos recursos existentes na IoT, a partir dos dados existentes, com o objetivo de prover Resource Description Framework (RDF), *Web* Ontology Language (OWL) e Efficient XML Interchange (EXI), de determinado serviço. Algumas técnicas podem ser usadas como *Resource Description.*

2.2 Conexão Wireless

A conectividade na área da computação, refere-se à disponibilidade de um dispositivo se interligar a outro (ou a uma rede). Portando a conectividade de um computador por exemplo é dada pela capacidade de se conectar a outros equipamentos seja ele smartfone, servidor ou até mesmo uma rede.

O padrão de conectividade em IoT é considerado favorável, uma vez que o número de dispositivos conectado possa elevar a banda de conexão, com tudo isso passa confiança, aquele que consegue unir velocidade na transmissão de dados (independentemente da distância), elevados padrões de segurança, baixo consumo de energia e, claro, projetos financeiramente viáveis (ODATA, 2022).

A ideia é verificar o cenário real, muitas vezes é necessário dar prioridade para alguns desses aspectos em detrimento de outros, e apenas a análise caso a caso dos resultados com a IoT poderá dizer qual opção de conectividade se encaixa melhor naquele cenário específico, segui o padrão pode definir o comportamento de conexão que deseja.

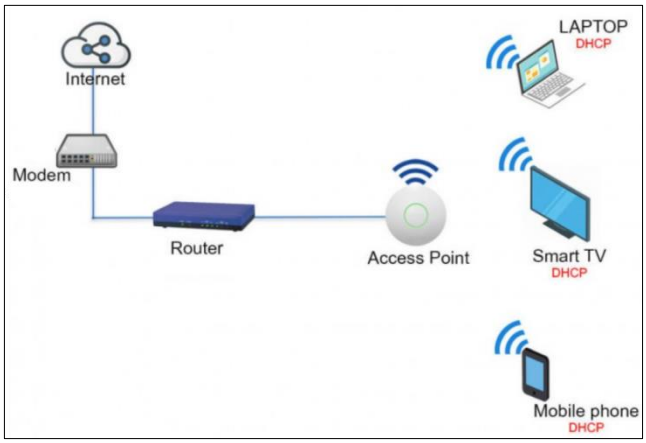
2.3 Access Point (Ponto de Acesso)

É um dispositivo de rede que permite aos dispositivos sem fio se conectarem a uma rede cabeada.

A ideia do ponto de acesso é permite que o usuário conecte vários dispositivos na mesma rede devido à necessidade de múltiplas conexões simultâneas, sem perder o alcance da comunicação sem fio. O número de dispositivos conectados na rede ao mesmo tempo pode chegar a 80 usuários, utilizados muitos em ambientes corporativos, empresas e estabelecimento comercial. Uma vez que o *Wi-Fi* não suportaria um volume grande de usuários conectados ao mesmo tempo (CISCO, 2022).

A Figura 3, mostra como funciona o access point.

Figura 3: Distribuição de *Internet* pelo Access Point.



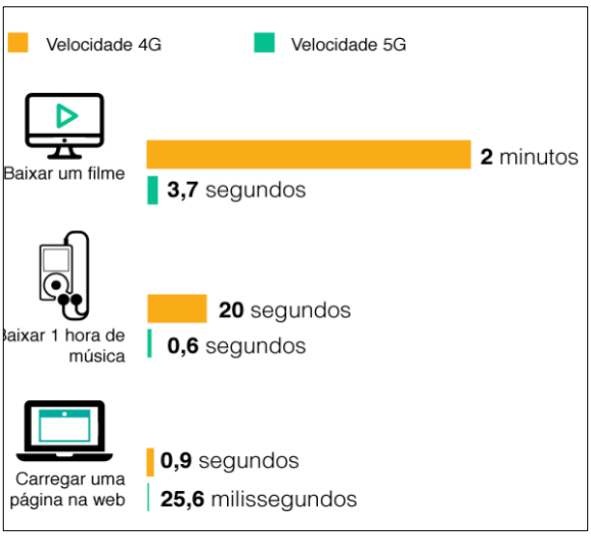
Fonte: CISCO, 2022.

2.4 Tecnologia 5G

A tecnologia 5G é a quinta geração de *Internet* móvel, uma versão evoluída e superior à conexão 4G. Espera-se que a tecnologia 5G traga mais velocidade para baixar e enviar pacotes de dados, reduzindo o tempo de resposta entre diferentes dispositivos e torne as conexões mais estáveis par todos (BBC NEWS, 2019).

A Figura 4 exibe algumas vantagens que a tecnologia 5G fornece. É interessante observar alguns pontos como cobertura de sinal e tamanho de dados.

Figura 4: Velocidade da tecnologia 5G.



Fonte: BBC NEWS, 2019.

A 5G é uma rede de conexão capaz de absorver conexões simultâneas de forma bastante eficiente, elevando um número de dispositivos conectados que pode chegar até 100

vezes mais que o 4G por unidade de área, todos com a mesma velocidade e latência. Um ponto que se destaca é o baixo consumo de energia.

Segundo a Mercados e Mercados MM, *Marketsand Markets* (2020), empresa internacional, projeta-se uma redução de 90% no consumo, o que se traduz em dispositivos de baixa potência em ambientes industriais e baterias com 10 anos de vida útil. Além disso, por ser programável, a 5G é capaz de se desconectar automaticamente quando não estiver em funcionamento (PINTO, 2020).

A conectividade 5G, em particular representa mais um passo na evolução de rede sem fio, abrindo diversos caminhos de possibilidades nos negócios para patamares nunca alcançados. A mesma pesquisa divulgada pela empresa MM relata anteriormente, destaca que o mercado de infraestrutura para 5G chegou à quase 3 bilhões de dólares em 2020, podendo alcançar mais de 33 bilhões de dólares, até 2026.

A conexão de tráfego cada vez mais carregado de dados que partirão desses dispositivos acaba exigindo demanda por interações praticamente instantâneas com tempo de resposta baixa tendendo a zero (0), faz da tecnologia 5G uma peça fundamental, sendo imprescindível a sua importância para efetividade de futuras soluções de IoT. A tecnologia 5G projeta alcançar latência de no máximo 1 milissegundo (PINTO, 2020).

2.5 Computação em Nuvem

É um modelo de entrega de serviços digitais sob demanda por meio da *Internet*. Esses serviços podem ser armazenamento de arquivos, redes, *software*s, bancos de dados e servidores.

A computação em nuvem aborda um papel essencial para o funcionamento efetivo e de baixo custo dos dispositivos de IoT, garantindo confiabilidade, flexibilidade, escalabilidade, segurança e privacidade dos sistemas a baixo custo. O legal disso é que esses dispositivos podem permanecer em ambientes hostis e de baixa confiabilidade, desde que seus dados sejam enviados periodicamente para a nuvem. (OLIVEIRA, 2017).

O diferencial desse modelo é a economia, pois dispensa a presença de servidores e possibilita a utilização de canais de comunicação de baixa confiabilidade e flexibilidade de acesso. O paradigma de computação em nuvem fornece funcionalidade para gerenciar, processar e armazenar a quantidade crescente de dados gerados por sistemas autônomos de IoT por meio de acesso conveniente e sob demanda a um conjunto de recursos (OLIVEIRA, 2017).

2.6 Automação Residencial

A automação residencial e os projetos têm como finalidade trazer mais conforto para os moradores, praticidade e facilidade para o cumprimento de tarefas rotineiras do dia a dia de maneira prática e segura.

Paulus et al. (2017, p. 6) comenta sua opinião sobre a automação residencial:

> A automação residencial é um ramo derivado da Automação Industrial, com foco nas operações domésticas. É responsável pelo controle de equipamentos elétricos e eletrônicos sem que haja necessidade de intervenção humana com os mesmos, mas sim, através de sistemas de controle. O grande objetivo é facilitar as tarefas do cotidiano, de modo a atender às necessidades das pessoas no que se refere à autonomia, segurança e conforto.

Desta forma, percebe-se que o fator que define a automação residencial não está ligado somente à modernização dos ambientes, mas ao conforto dos usuários, que pode facilitar a vida das pessoas em qualquer ambiente, especialmente no que se refere de condições necessárias para que indivíduos com deficiência possam acionar remotamente determinados dispositivos.

O fator por trás disso está associado à integração de sistemas, junto à capacidade de executar ações, mediante as instruções estabelecidas pelo usuário (MURATONI. B. et al. 2011).

A automação residencial proporciona diversos benefícios como:

- Acessibilidade: as pessoas com deficiência limitada, proporcionando maior conforto e comodidade;
- Conforto: é o benefício mais agradável na automação residencial. A experiencia única ao usuário permite o controle exclusivo de forma flexível e prática;
- Segurança: o sistema de segurança é um fator fundamental em qualquer residência, proporcionando ao usuário mais tranquilidade para realizar suas atividades com segurança. O sistema de integração com a automação residencial, permite que o usuário tenha o controle de sua residência na palma da mão, controlando e monitorando as câmeras de segurança e os alarmes, por exemplo.

3 MICROCONTROLADORES

Este capítulo tem como objetivo documentar um estudo sobre microcontroladores e sua importância na automação residencial.

O microcontrolador é um circuito integrado programável com funções de leitura e escrita que contém todos os componentes de um computador tais como *Central Processor Unit*, Unidade Central de Processamento (CPU), memória, portas de entrada e saída, entre outras funções, com capacidade de desempenhar diversas funções de controle, como controlar um sistema de monitoramento, movimentos de robôs, acionamento de portas, janelas, e entre outros dispositivos periféricos (MIYADAIRA, 2009).

Sobre o poder dos microcontroladores Chase (2007, p. 6) afirma que:

> Os microcontroladores são pequenos sistemas computacionais bastante poderosos que englobam em um único chip: interfaces de entrada/saída digitais e analógicas, periféricos importantes como a memória RAM, memória flash, interfaces de comunicação serial, conversores analógicos/digitais e temporizadores/contadores. A vantagem dos microcontroladores é que além de possuir os periféricos integrados a um único chip, são responsáveis por executar e armazenar os programas escritos para eles (firmware), assim como a capacidade de absorver mais funções com o incremento de periféricos, através de CI´s "driver´s", como comunicação USB [...]. Com o advento dos microcontroladores de 16 e 32 bits (atualmente o padrão é de 8 bits) a capacidade de gerenciar soluções mais complexas e maior velocidade de processamento se iguala ao do microprocessador. O crescimento dos sistemas embarcados muito se deve a este componente.

Os microcontroladores não possuem nenhuma função específica de fábrica, a ideia das empresas que fabrica, é que a programação da placa fica de acordo com a necessidade de cada usuário, especificando a utilidade para cada função que deseja usar (MIYADAIRA, 2009).

Os microcontroladores necessitam ser programados utilizado a linguagem específica para que possam operar corretamente. Isso pode ser feito fora da placa de circuito ou na própria placa onde o microcontrolador vai operar.

3.1 Microcontrolador na Automação Residencial

O grande avanço na área de automação residencial fez com que a procura por dispositivos eletrônicos aumentasse, uma vez que o uso de microcontrolador seria ideal para automatizar determinado objeto na residência, levando mais praticidade, conforto e segurança as pessoas.

A importância disso se destaca atualmente pelo número de benefícios que pode trazer na vida das pessoas. Observa-se que algumas residências já têm alguns tipos de equipamento automatizado, por exemplo persianas via controle remoto.

3.2 Protocolos de Comunicação sem Fio

Os protocolos de comunicação são baseados em um conjunto de regras e procedimentos que efetuam o processo de controle e a permissão da troca de dados entre máquinas e seus periféricos. Portanto, eles podem garantir que a troca de informações seja realizada de forma eficiente e segura.

Silva (2021) destaca alguns protocolos de comunicação sem fio: o Z-wave, o Zigbee, o Wi-Fi e o *Bluetooth*.

O Z-wave é um protocolo de rede sem fio focado em acesso remoto, segurança e economia de energia. Destaca se o uso em área residencial, comercial e prédios de grande porte. No Brasil a frequência é a AU 921,4 MHz, podendo mudar dependendo do país, o Z-wave não sofre interferência de redes que usam frequência de 2.4 GHz, como roteadores e telefone sem fios.

Algumas vantagens de usar o Z-wave são:

- Focado em gestão de economia de energia e segurança;
- Facilidade de instalação por ser sem fio;
- Funciona em rede *mesh*;
- Funciona com 2 ou 200 equipamentos ao mesmo tempo;
- Qualquer equipamento elétrico é passível de ser controlado;
- Monitoramento em tempo real dos dispositivos;

O Zigbee é um protocolo de comunicação que cria sua própria rede entre os componentes compatíveis em uma residência, por exemplo. É focado para trabalhar em automação residencial, com superioridade muito acima do protocolo Wi-Fi, em relação à velocidade. É também focado em periféricos, dispositivos como sensores de presença, sensores de movimento que podem ser usados em portas, portões e janelas.

É considerado um protocolo rápido, seguro, robusto e seu uso depende de um concentrador, conhecido como *Hub* que é um equipamento responsável pela criação da rede.

Algumas vantagens de se usar o Zigbee são:

- Zigbee funciona em sua própria rede, não depende do Wi-Fi;

- Zigbee é um protocolo sem fio;
- Caso um dispositivo apresente problema, a rede procura outro caminho;
- Uma única rede pode suportar 65 mil dispositivos;
- O maior foco é em dispositivos de baixa potência.

O *Bluetooth* foi projetado para permitir a comunicação sem fio de baixo custo e baixo consumo, em distâncias curtas. Ele possui padrão aberto, porém é limitado, permite que vários dispositivos se comuniquem no limite de até oito dispositivos dentro de uma faixa de 10 até 100 metros, dependendo da classe de potência apresentada na Tabela 1.

Tabela 1 - Classe de Potência do *Bluetooth*

Classe	Potência [mW]	Alcance [m]
1	100	100
2	2,5	10
3	1	1
4	0,5	0,5

Fonte: DIAS, 2016.

Dispositivos *Bluetooth* usam ondas de rádio para se conectar entre si. A comunicação acontece em um curto alcance, em redes *ad-hoc* conhecidas como *piconets*. Uma *piconet* é formada por um dispositivo *Bluetooth* servindo como mestre e um ou mais dispositivos servindo como escravos. Os escravos podem se comunicar apenas com seu mestre em um ponto-a-ponto mediante o controle do mestre. Um dispositivo escravo pode estar em modo ativo ou em espera, sendo o último utilizado para reduzir o consumo de energia.

3.3 Wireless Fidelity (Wi-Fi)

O *Wi-Fi* conhecido como *Wireless Fidelity* (fidelidade sem fio) é um protocolo de comunicação baseado no padrão IEEE 802.11 que possui uma diversidade de capacidade e cobertura de dispositivos além de ter um custo baixo. Muito utilizado no dia a dia, este padrão de comunicação sem fio é flexível, suporta diversos dispositivos conectados, possui segurança e controle de acesso (OLIVEIRA, 2017).

O uso do Wi-Fi apresenta algumas vantagens tais como:

- Flexibilidade;
- Facilidade de uso;
- Variedade de dispositivos compatíveis;

- Facilidade de configuração dos dispositivos;
- Custo baixo para implementar projetos *smart*.

A vasta utilização do *Wi-Fi* está associada também à automação residencial, embora vários dispositivos usados na automação já vêm com essa tecnologia integrada, no caso do microcontrolador ESP32 possui conectividade tanto *Bluetooth* quanto *Wi-Fi*.

A proposta é conectar um conjunto de dispositivos em redes locais sem fio. A rede *Wi-Fi* considera um número finito de elementos de rede ligados em um mesmo canal, porém esse número é ilimitado dentro da teoria.

3.4 Protocolo MQTT

O *Message Queuing Telemetry Transport*, Transporte de Filas de Mensagem de Telemetria (MQTT) é um protocolo de sintaxe simples e interface leve e foi projetado para dispositivos que utilizam largura de banda baixa, com alta latência e com requisitos de hardware simples. Ele foi criado pela IBM em 1999, para comunicação *Machine to Machine*, Máquina a Máquina (M2M) e permite a comunicação bidirecional, diferente do *Hypertext Transfer Protocol*, Protocolo de Transferência de Hiper Texto (HTTP) mas, a seu modo específico coleta dados e gerência dispositivos da IoT (JUCA, 2018).

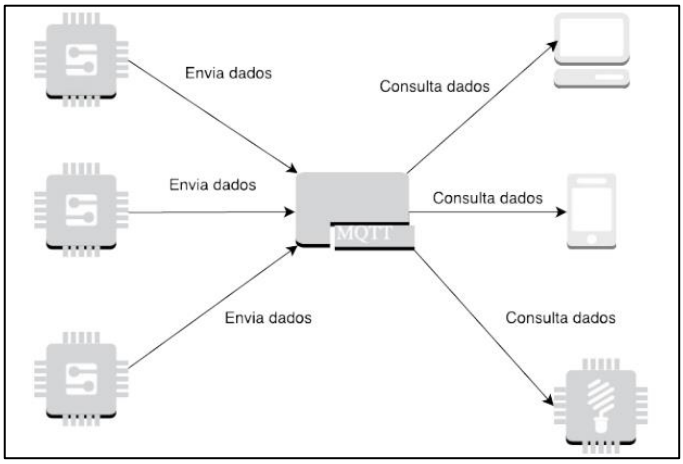
O MQTT usa o conceito de *broker*, que é um *software* servidor que recebe solicitações e as repassa, quando solicitado. O *broker* atua como servidor nos dois sentidos, tanto recebendo, quanto enviando os dados. Assim, evita a necessidade de manter um servidor na rede, atuando sempre nessa função (OLIVEIRA, 2017).

Segundo Juca (2018) o protocolo MQTT é baseado no *Transmission Control Protocol/Internet Protocol*, Protocolo de Controle (TCP/IP) e ambos, cliente e *broker*, necessitam da pilha TCP/IP para o seu funcionamento, ou seja, é possível interagir via MQTT com um microcontrolador ESP32 ou Raspberry Pi de qualquer lugar do planeta que possua acesso à *Internet*, usando a porta de transporte TCP 80. Outro ponto é que este protocolo possui implementações nos mais diversos sistemas operacionais, permitindo assim que dispositivos totalmente diferentes "conversem" pela *Internet*.

O MQTT é divido em três partes: *Publishers*, *Pubscribers*, *Broker***.**

Os *publishers* disponibilizam as informações, os *subscribers* recebem as informações e por último, o *broker* é o servidor MQTT na nuvem, acessível de qualquer lugar que contenha conexão com a *Internet* e é responsável por receber e redirecionar todas as mensagens dos clientes.

Figura 5: Funcionamento básico do MQTT.



Fonte: OLIVEIRA, 2017.

3.5 Pulse Width Modulation (PWM)

O Pulse Width Modulation, Modulação Por Largura de Pulso (PWM) é considerada uma técnica com objetivo de controlar a potência de dispositivos ligados à saída de sistemas microcontrolados. O princípio é modular um sinal de onda quadrada sobre a alimentação do dispositivo (OLIVEIRA, 2017).

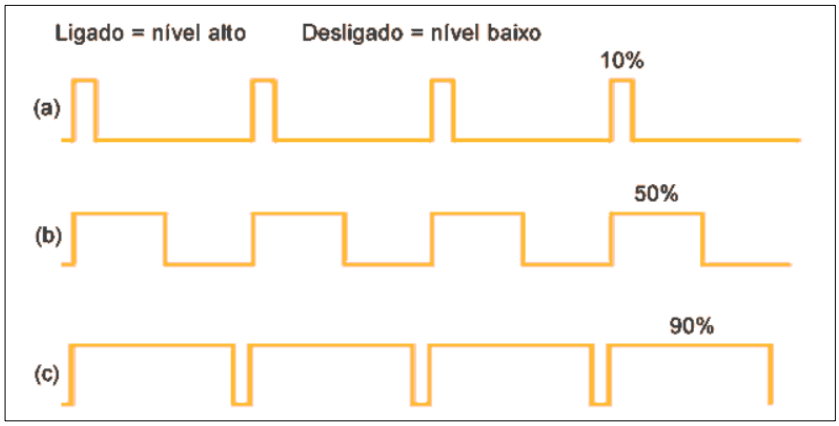
A técnica do módulo PWM apresenta movimentos proporcionais aos comandos indicados, controlando o giro e a posição, diferente da maioria dos motores convencionais.

Levando em consideração um dispositivo de malha fechada, seu funcionamento é feito por meio do recebimento de um sinal de controle, que verifica a posição atual, e atua no sistema indo para a posição desejada, oferecendo um torque máximo de ~0,80, quilogramas, posição de 360º e com peso de 25 gramas.

O eixo do motor possui liberdade de 360º e corrente contínua, com o ângulo proporcional ao sinal positivo do PWM (FILIPEFLOP, 2021).

A Figura 6, apresenta três saídas PWM que codificam três tipo de diferentes valores de sinal analógico tais como 10%, 50% e 90% da energia de entrada. Para entender melhor quando alimentação for 9V e o ciclo de trabalho for 10%, tem-se um sinal analógico 0,9V, por exemplo.

Figura 6: Controle de Sinal PWM.



Fonte: USINAINFO, 2022.

Neste trabalho é utilizada a ponte H, visto na seção 4.4.1.2 que usa esta técnica, para controlar os dispositivos eletrônicos, como o motor usado nesta monografia.

4 REFERENCIAL TEÓRICO

Este capítulo aborda a descrição de todos os componentes eletrônicos na parte *hardware* utilizado nesta monografia, como pode-se observar nas seções seguintes.

4.1 Orçamento dos Componentes

Para a realização deste projeto, foram utilizados os seguintes componentes eletrônicos, como mostra a Tabela 2.

Tabela 2 - Orçamento dos Componentes Usados no Projeto – 2022.

Componentes	Quantidade	Valor Unitário(R$)	Total(R$)
NodeMCU ESP32 Iot com WiFi e Bluetooth -30 Pinos	1	72,5	72,50
Placa Protoboard 640 furos	1	15,9	15,90
Modulo Reles 5V 2 Canais	1	46,9	46,90
Driver Ponte H – L298N	1	28	28,00
Sensor de Movimento PIR HC-SR501 100°	1	13	13,00
Motor DC 3V-6V 45RPM	1	35,6	35,60
Jumpers - Macho/Fêmea - 20 Unidades de 20cm /40cm	1	8,9	8,90
Fechadura Elétrica Solenoide 12V NF FEC-91	1	45,9	45,90
Total (R$):	8	1	266,70

Fonte: Elaborada pelo autor.

Para a construção desse trabalho, os componentes utilizados têm uma relação custo-benefício consideravelmente regular, pois se mostra viável para esse tipo de aplicação. O poder dos componentes é considerado altamente tecnológico, como o microcontrolador ESP32, capaz de desempenhar funções de controle no recebimento de informações e acionar dispositivos em longa distância, lembrando que a utilização desses componentes se aplica em amplos cenários de projetos de diferente segmento, e até mesmo em projetos futuros.

4.2 Microcontrolador ESP32 NodeMCU

Um microcontrolador é considerado um pequeno dispositivo inteligente e poderoso, constituído de uma *Central Processor Unit*, Unidade Central de Processamento (CPU), memória de dados e periféricos como portas de entrada e saída *In/Out*, Entrada/Saída (I/O), com capacidade de desempenhar várias funções de controle (MIYADAIRA, 2009).

O microcontrolador é o principal componente da aplicação, visto que é responsável por receber informações e acionar dispositivos. Assim, se torna econômico e viável, empregar esses equipamentos para controlar outros dispositivos e possuem as seguintes características (TARGA, 2019).

- Quantidade de Memória Interna;
- Baixo Consumo de Energia;
- Baixo Custo;
- Tamanho;
- Eficiência de Processamento.

Existe no mercado a família dos microcontroladores com diversas opções para aplicações de projeto. No caso deste projeto foi escolhido o microcontrolador ESP32 NodeMCU.

O módulo ESP32 NodeMCU consiste em uma placa de desenvolvimento baseada no módulo ESP32S Wi-Fi, o qual é um componente eletrônico altamente tecnológico desenvolvido especialmente para conectar projetos robóticos ou de automação residencial a *Internet* com grande facilidade e baixo custo.

Um diferencial desse módulo é que, além de possuir conexão Wi-Fi nativa e *Bluetooth* embutido, a capacidade de processamento é muito rápida, tornando seu projeto ainda mais prático, aumentando as possibilidades de uso. Ele é um *firmware* e *kit* de desenvolvimento *Open Source* que auxilia na prototipagem do produto voltado à IoT, necessitando de apenas de um *script* escrito por poucas linhas. Isso significa que ele não precisa de um Arduino para colocá-lo em funcionamento, mostrando ser um sistema microcontrolador com possibilidade de *interface* micro *Universal Serial Bus*, Barramento Serial Universal (USB), além da comunicação sem fio Wi-Fi (ESPRESSIF, 2021).

O microcontrolador módulo ESP32 NodeMCU possui 30 pinos, sendo 25 pinos *General Purpose Input/Output*, Entrada/Saída de Uso Geral (GPIOs), com função *Pulse Width Modulation*, Modulação de Largura de Pulso (PWM), *Inter-Integrated Circuit* - Circuito Inter-Integrado (I2C) e *Serial Peripheral Interface*, Interface Periférica Serial (SPI), sendo 18 entradas *Analog to Digital*, Analógico para Digital (A/D), e duas saídas *Digital to Analog*, Digital para Analógico (D/A).

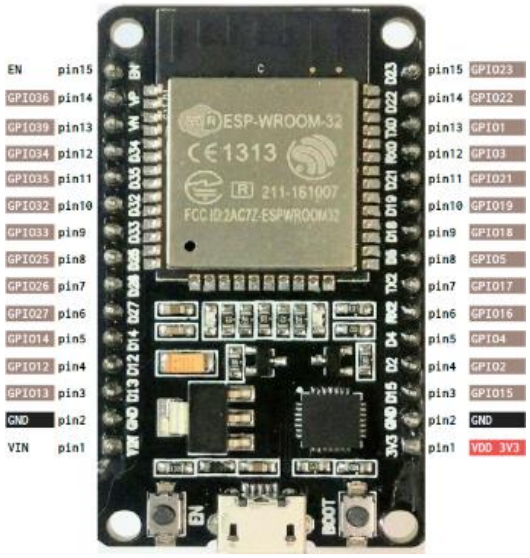
Especificações:

- Microcontrolador: Tensilica Xtensa 32-bit LX6;
- Controlador: ESP-WROOM-32;

- RAM: 520 K*byte*s;
- ROM: 448 K*Byte*s;
- Tensão de operação: 4,5 ~ 9V DC;
- Modo de operação: STA/AP/STA+AP;
- Nível lógico: 3.3V;
- Clock máximo: 240MHz;
- Taxa de transferência: 110-460800bps;
- Suporta Upgrade remoto de firmware;
- Flash: 4 MB;
- Portas GPIO: 25;
- GPIO: funções PWM, I2C, SPI;
- Conversor analógico digital (ADC);
- Memória flash externa: 32Mb (4 mega*byte*s);
- Antena embutida;
- Conector micro-USB;
- Bluetooth BLE 4.2;
- Wireless padrão 802.11 b/g/n;
- Conexão *Wifi* 2.4GHz (máximo de 150 Mbps);
- Suporte a redes WiFi: 802.11 b/g/n (2,4 a 2,5GHz);
- Segurança: WPA/WPA2/WPA2-Enterprise/WPS;
- Criptografia: AES/RSA/ECC/SHA;
- Distância entre pinos: 2,54 mm;
- Dimensões: 49 x 25,5 x 7 mm;
- Peso: 9g.

A Figura 7, mostra a disponibilidade das portas representada por pino numerado do módulo ESP32 NodeMCU.

Figura 7: Módulo ESP32 NodeMCU



Fonte: Adaptado (FERNANDO K, 2018)

4.2.1 Módulo Relé 5V 10A 2 canais com Opto-acopladores

O relé 5V 10A é um módulo de acionamento que permite integração com outros sistemas de microcontroladores, com finalidade de controlar cargas de variadas tensões de corrente alternada de até 10A (Amperes), servindo como um interruptor que abre e fecha de acordo com a configuração que foi predefinida na sua fabricação.

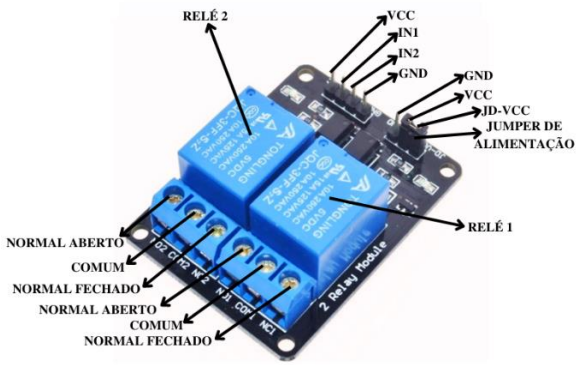
O uso de opto-acopladores deste módulo da possibilidade para que o componente seja capaz de isolar uma região da outra, em casos de descargas elétricas ou mal contato, ou seja trabalhar como um sistema de segurança que tem as especificações: (USINAINFO, 2022).

- Tensão de operação: 5VDC (VCC e GND);
- Tensão de sinal: 5VCD (IN1 e IN2);
- Tensão de saída: (30VDC a 10ª) ou (250VAC a 10A);
- Corrente típica de operação: 15~20mA;
- Controlar cargas de: 220V AC;
- Tempo de Resposta: 5~10ms;
- Relé possui (Cada): 3 terminais proporcionando 1 contato NA, 1 NF e o CM;
- LED: Indicador de status;
- Indicador: 4 furos de 3mm para fixação, nas extremidades da placa;

- Dimensões: 51 x 38 x 20mm;

A Figura 8 apresenta a pinagem do módulo relé com suas características na estrutura.

Figura 8: Módulo Relé 5V 10A 2 canais



Fonte: USINAINFO, 2022 (imagem original modificada).

Sua alimentação é de 5V - VCC (Voltagem em Corrente Contínua), os pinos IN1 e IN2 são entradas de controle de relé 1 e relé 2, alimentado pelo nível lógico baixo 0V, já os pinos neutros *Graduated Neutral Density*, Filtro Graduado de Densidade Neutra (GND) de 0V, o pino JD-VCC é uma alimentação externa caso precise ativar os relés, também vem acompanhado de um jumper de alimentação para escolher qual o relé desejar alimentar (USINAINFO, 2022). É composto por um contato NA - normalmente aberto e um NF - normalmente fechado e o CM - comum para cada relé, como pode-se visualizar na Figura 08.

4.2.2 Módulo Driver Ponte H – L298N

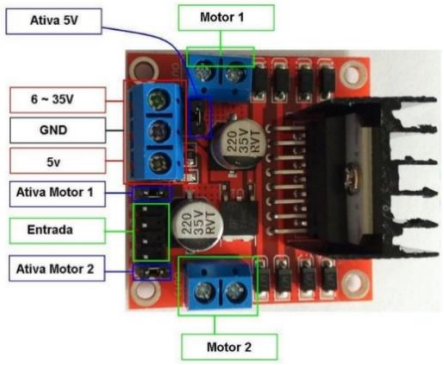
O Módulo Driver Ponte H L298N é um módulo de controle para cargas indutivas, baseado de um chip, a ponte H é usada para controlar velocidade e sentido de rotação de um ou até dois motores ao mesmo tempo, por exemplo o motor DC ou motor de passo e tem as especificações:

- Chip L298N;
- Tensão de operação: 5V~35V;
- Tensão lógica: 5V;
- Corrente lógica: 0mA ~ 36mA;
- Corrente de operação máxima: 2A por canal / 4A max;
- Limites de temperatura: - 20 a + 135ºC;

- Potência máxima: 25W;
- Dimensões: 43 x 43 x 27mm;
- Peso: 30g;

Na Figura 9 se observa a pinagem da ponte H, composta por um regulador de tensão integrado de (5V), que disponibiliza uma saída regulada de mais +5V no pino (5V), para uso externo de um segundo componente (USINAINFO, 2022).

Figura 9: Pinagem do Módulo Driver Ponte H L-298N



Fonte: MUNDOELETRÔNICO 2022.

A entrada de dados é composta por 4 pinos IN1, IN2, IN3 e IN4, responsável pela rotação do motor A (IN1 e IN2) e Motor B (IN3 e IN4), os 4 pinos de saída se divide em dois pinos para cada motor 1 e 2, já os pinos Ativa M1/Ativa M2 é responsável pelo controle PWM do motor. O pino de alimentação externa 6~35V é responsável por controlar componente que opera acima de 12V, e um pino GND.

4.3 Sensor de Movimento PIR HC-SR501 100°

O sensor HC-SR501 100º é um dispositivo que permite a detecção de movimentos e responde a um estímulo físico baseado na variação da luz infravermelha emitida pela radiação do corpo humano. (SILVEIRA, 2016).

O funcionamento se dá pela capacidade de detectar movimento em uma área de até 7 metros trabalhando com um ângulo de até 100°, caso alguém se movimentar nesta área o pino de alarme é ativo. Lembrando que o sensor é capaz de captar somente objetos que estão em movimento e que exalam calor, como o corpo humano.

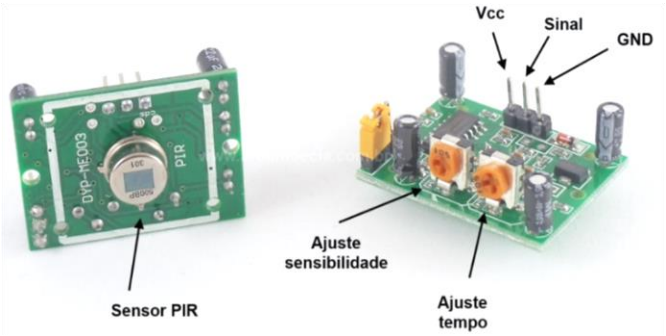
O sensor de movimento *Passive Infrared*, Infravermelho Passivo (PIR), é composto de uma placa com pino de alimentação de 5V um pino GND, pino de sinal de saída, constituído por um piroelétrico e uma lente Fresnel que são responsáveis por ampliar a

radiação infravermelha e 2 pinos de ajuste de sensibilidade e tempo para detecção de movimento e tem as especificações: (USINAINFO, 2022).

- Modelo: HCSR501;
- Chip: BISS0001;
- Alimentação: 5V – 20VDC;
- Tensão de dados: 3,3V (Alto), 0V (Baixo);
- Temperatura de trabalho: - 20 ~ +80ºC;
- Distância detectável: 3 - 7m (Ajustável);
- Tempo de bloqueio: 2,5seg (Default);
- Tempo de delay: 5-200seg (default: 5seg);
- Dimensões: 3,2 x 2,4 x 1,8cm;
- Peso: 7g;

Na Figura 10 pode-se visualizar a pinagem e características na estrutura.

Figura 10: Sensor de movimento *PIR* HC-SR501 100º



Fonte: ARDUINO&CIA 2014;

4.3.1 Motor DC 3V- 6V45RPM

O motor DC45RPM é um dispositivo que produz movimento, que atua de acordo com sua função e configuração. É uma opção barata e fácil de usar, bastante utilizado em aplicações na área de robótica e nos sistemas microcontroladores. O motor requer uma alimentação de 4,5V, e 2 terminais positivo/negativo, através dos dois terminais existentes na ponta do Motor DC é possível fazer a inversão da polaridade, o que possibilita ao motor girar tanto no sentido horário quanto anti-horário sendo capaz de adicionar movimento no eixo do motor para determinada funcionalidade (SILVA, 2021).

A força de trabalho (torque do motor) pode atingir uma velocidade de aproximadamente 22,5 rotações por minuto em 3V e até 45 rotações por minuto em 6VDC.

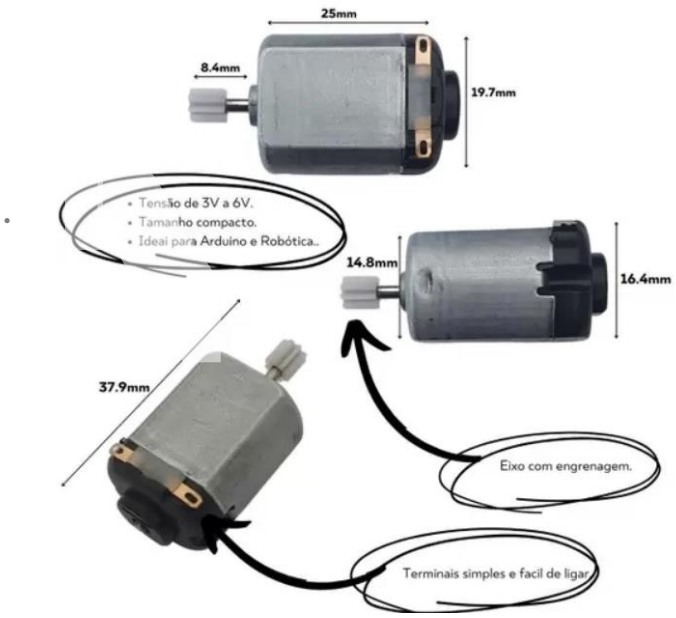
A rotação pode aumentar dependendo o quanto é atribuído na alimentação do motor. (USINAINFO, 2022).

As especificações do motor são apresentadas a seguir:

- Modelo: MDC45R;
- Alimentação: 3 – 6VDC;
- Terminais: 02 polos;
- Rotação: 22,5RPM (3V) / 45RPM (6);
- Dimensões: 20MM(L) X 15MM(A) X 38MM(C);
- Peso: 29g;

Na Figura 11 pode-se observar que o motor DC45RPM possui um eixo com engrenagem que serve para deslisar na cremalheira do portão.

Figura 11: Motor DC45RPM



Fonte: FILIPEFLOP, 2021.

4.3.2 Fechadura Elétrica Solenoide 12V NF FEC-91 – Lingueta Superior

A fechadura elétrica solenoide 12V NF FEC-91 foi projetada com intuito de auxiliar e tornar mais prático e tecnológico o acesso de locais. A trava elétrica solenoide é alimentada por uma tensão de 12V em seus terminais.

Quando a tensão da fechadura for liberada, ela atrai o pino para sua parte interna e quando a tensão for cortada o pino retorna à sua posição original (USINAINFO, 2022).

A fechadura elétrica é considerada um meio de segurança inteligente por proporcionar um controle de entrada e saída de pessoas. Essa permissão é dada através de um único botão de acionamento para a sua abertura ou fechamento. Geralmente, a porta é aberta à distância, seja por um porteiro ou pelo próprio morador, após a identificação do visitante ou pessoa conhecida via interfone.

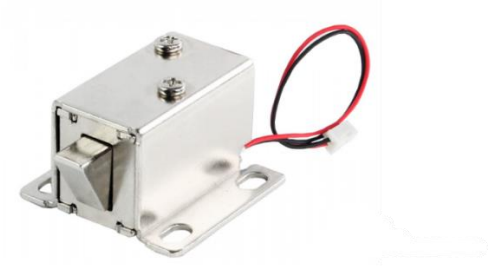
O funcionamento é constituído por uma bobina elétrica, também conhecida como solenoide. Dessa maneira, ao ser percorrido por uma corrente elétrica, o elemento se transforma em um eletroímã, atraindo uma lingueta de ferro e abrindo a porta. Quando isso acontece, a fechadura elétrica emite um som característico, que indica a permissão para abrir a porta. Enquanto isso, o sistema de fechamento consiste em uma mola que empurra a lingueta para travar a porta (SOPRANO, 2020).

As especificações da fechadura elétrica são:

- Modelo: FEC-91;
- Tensão: 12VDC;
- Corrente máxima: 0,5A;
- Pulso máximo: 10seg;
- Lingueta: superior
- Comprimento do cabo: 11cm;
- Comprimento do pino: 6mm;
- Dimensões: 33x27, 5x17,5mm;
- Material: Aço inox;
- Peso: 39g;

A Figura 12 apresenta a fechadura elétrica com 4 furos para fixação.

Figura 12: Fechadura Elétrica Solenoide.



Fonte: USINAINFO, 2022.

4.3.3 Placa Protoboard 640 furos e Jumpers Macho/Fêmea

O *protoboard*, mais conhecido como placa de ensaio, é usado para criação e montagem de circuitos eletrônicos experimentais sem a necessidades de soldagem. A placa de 640 furos proporciona agilidade na conexão dos componentes que deseja usar no desenvolvimento de projeto, como mostra a seguinte Figura 13.

Figura 13: Placa Protoboard 640 furos.

Fonte: USINAINFO, 2022.

O *jumper* é conhecido como fio chamado de macho ou fêmea por ter a ponta com conectores relativamente ideal para realizar conexão de dispositivo para outro na placa *protoboard*. O seu uso é indispensável quando se refere desenvolvimento de circuitos eletrônicos, com várias cores e facilita a identificação e a ligação de uma ponta a outra, evitando a perda de tempo durante a montagem, como pode-se observar na Figura 14 (FILIPEFLOP, 2021).

Figura 14: Jumpers Macho/Fêmea, Macho/Macho, Fêmea/Fêmea.

Fonte: FILIPEFLOP, 2021.

4.3.4 Cremalheira

A cremalheira é uma peça produzida de liga plástica reforçada, tem como objetivo o uso em diversas finalidades de aplicação na área da robótica.

A alta durabilidade e resistência na estrutura proporcionam segurança no projeto que tem como objetivo usar, com conjunto de 81 dentes a cremalheira é compatível com diversas engrenagens dentadas, facilitando assim o encaixe simplificado da mesma (USINAINFO, 2022).

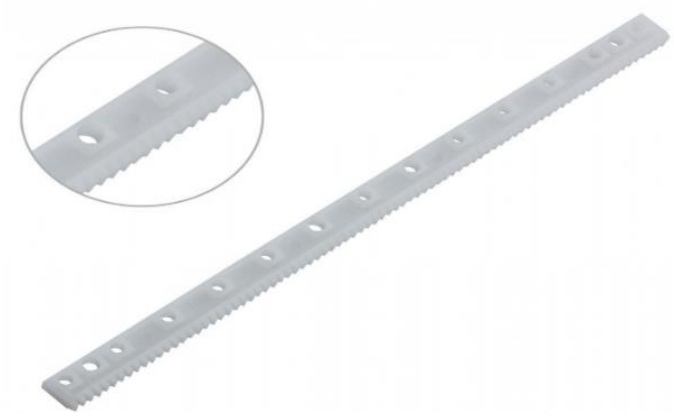
As especificações da cremalheira são:

- Modelo: CR15;
- Comprimento: 125mm;

- Altura: 7mm;
- Largura: 4mm;
- Peso: 3g;

A Figura 15, aborda a estrutura da cremalheira.

Figura 15: Cremalheira



Fonte: USINAINFO, 2022.

4.3.5 Fonte de Alimentação DSA-36W-12 1

A fonte de alimentação DAS-36W- 12 1 é um periférico que vem acompanhado de aparelhos eletrônicos com objetivo de transformar a corrente alternada que chega pelas tomadas em uma corrente elétrica contínua**.**

A fonte recebe energia em 110V ou 220V de corrente alternada que se transforma na tensão adequada, para alimentar o aparelho de acordo com sua necessidade de alimentação, que na maioria é de 12V.

O uso de fonte de alimentação é indispensável, quando se refere ao funcionamento de equipamento que necessita de energia elétrica. Algumas fontes podem elevar um nível de tensão alta ou baixa, também tem a que isola o circuito da rede de energia, capaz trabalhar como um sistema de segurança para proteger os aparelhos contra estabilidade na rede e picos de energia e é apresentada na Figura 16 (ILUMINIM, 2022).

As especificações da fonte de alimentação são:

- Modelo: DSA-36W-12 1;
- Entrada: 100-240V 50~60Hz 1,0A (Bivolt);
- Saída: DC 12V 3,0A;
- Pino: 5,5mm x 2,0mm;
- Dimensões: 11,5 x 4,5 x 3,3 cm;

Figura 16: Fonte de Alimentação DSA-36W-12 1.

Fonte: Elaborada pelo autor.

A utilização desta Fonte de Alimentação DSA-36W-12 1 foi fundamental para o desenvolvimento do projeto, com objetivo de alimentar a fechadura elétrica solenoide, uma vez que sua alimentação necessita de 12V para funcionamento. É uma fonte usada que pertencia a um notebook antigo, mas o uso dessa fonte foi viável ao projeto, evitando o gasto de comprar uma fonte nova.

5 IMPLEMENTAÇÃO DE UM PORTEIRO ELETRÔNICO

Este capítulo tem por objetivo documentar a implementação de um porteiro eletrônico dotado de controle e monitoramento de abertura e fechamento via *web* em tempo real.

5.1 Introdução

São apresentados as características e o detalhamento do projeto, os materiais usados, as suas dimensões, as funcionalidades, as dificuldades encontradas na implementação e ajustes no projeto, e os resultados dos testes.

A construção desse projeto foi feita em 4 etapas:

1. Construção de um Modelo Representativo de um portão eletrônico (maquete): parte que completa todos os componentes como sensores, relés, fechadura e motor posicionados, para realizar os testes necessários no ambiente físico;
2. Construção de um projeto de microcontrolador: agente responsável por monitorar e controlar os dispositivos que estão integrados à sua estrutura, sendo a recepção dos sinais de processamento enviados pelos periféricos e acionamentos aos dispositivos para que executam determinadas operações de controle;
3. Construção de um Servidor *Web*: responsável por processar as requisições, através de solicitações HTTP enviadas pelo usuário, retornar e salvar dados como histórico de eventos, arquivo *Comma Separated Values*, Valores Separados Por Vírgulas (CSV), que trabalha também como banco de dados;
4. Construção de uma Plataforma *Web*: visualizar os dados obtidos de monitoramento do portão eletrônico, fornecidos por requisições do servidor *Web*.

5.2 Estrutura física da maquete

Nessa seção são descritas as etapas e procedimentos metodológicos para o desenvolvimento do projeto da maquete do portão.

5.2.1 Maquete modelo 3D

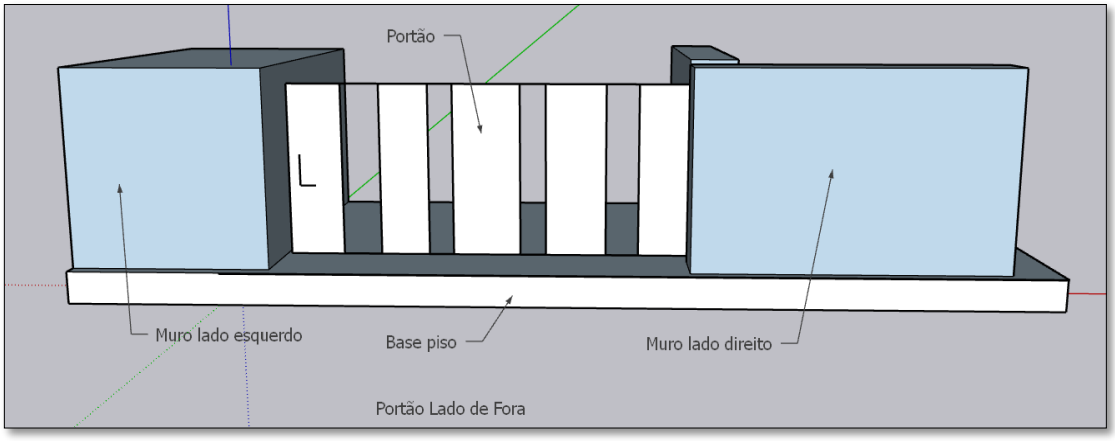
Foi criado um modelo 3D representativo em forma de maquete, um portão eletrônico de uma residência. Para isso, foi utilizado o SketchUP, é um *software* que permite o

desenvolvimento de projetos 3D de excelência, prometendo facilidade aos usuários na criação de modelos deste tipo, além de móveis e interiores (SKETCHUP, 2022).

É importante ressaltar que o *software* dá todas as medidas necessárias como dimensões em escala real, altura, largura e comprimento, para facilitar os cortes das peças e posterior montagem da maquete.

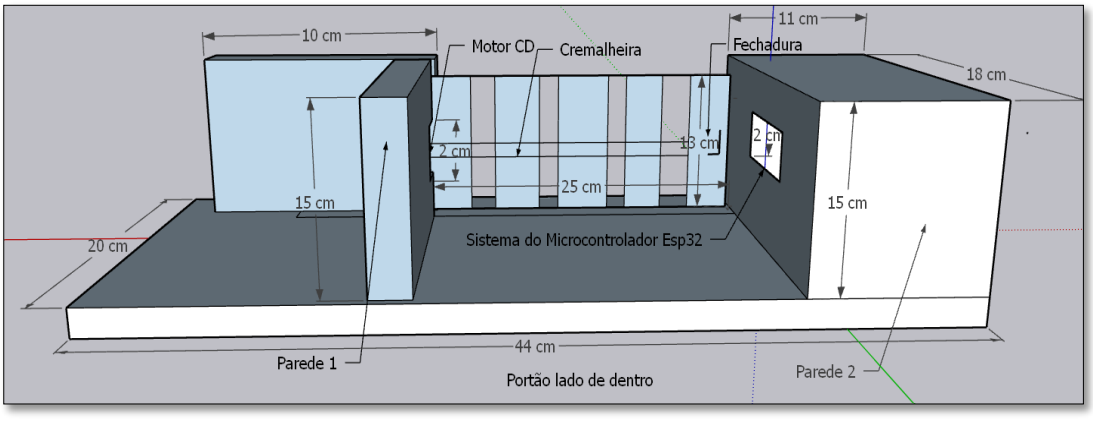
O uso desse *software* foi de grande utilidade no desenvolvimento deste trabalho pois permite obter uma projeção para as dimensões da estrutura da maquete em tamanho real, como mostra nas Figuras 17 e 18.

Figura 17: Modelo Representativo do Portão Eletrônico (Lado de Fora)



Fonte: Elaborado pelo autor

Figura 18: Modelo Representativo do Portão Eletrônico (Lado de Dentro)



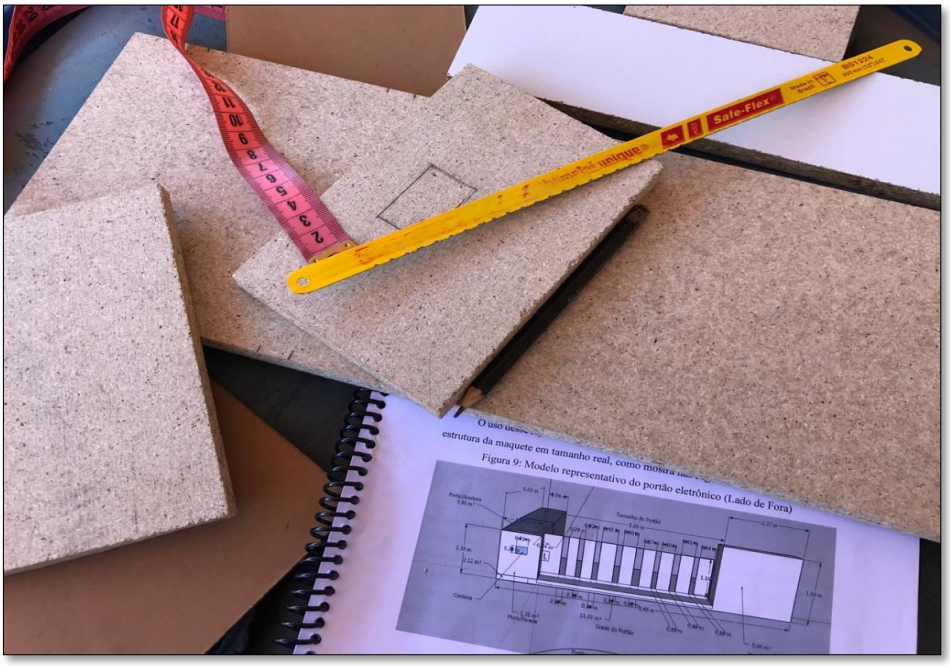
Fonte: Elaborado pelo autor

5.2.2 Execução da Maquete

Nessa etapa foi feita a construção do modelo representativo do portão eletrônico de acordo com o modelo 3D proposto. Foi feita a confecção das peças cortadas e a montagem

do portão. Os cortes foram feitos de um guarda-roupa antigo feito de compensado de madeira e *Polyvinyl Chloride*, Policloreto de Vinila (PVC). Após os cortes das peças foi realizada a montagem da maquete, com mostra nas Figuras 19 e 20.

Figura 19: Cortes das peças – parte 1



Fonte: Elaborada pelo autor

A montagem da maquete foi dividida em mais de uma etapa, como visto na Figura 20.

Figura 20: Cortes das peças – parte 2

Fonte: Elaborada pelo autor

As Figuras 19 e 20 apresentam as peças de compensado de madeira, antes da montagem. Foram usadas algumas ferramentas como auxílio para realizar a montagem do portão, como parafusos, lâmina de serra, fita métrica, cola de madeira, e cola quente. A montagem foi realizada em seguida, como mostra a Figura 21.

Figura 21: Montagem da Maquete

Fonte: Elaborada pelo autor

Na Figura 22 se visualiza a maquete do portão montada e finalizada como proposto na seção 5.2.1 desse trabalho.

Figura 22: Maquete do portão finalizada

Fonte: Elaborada pelo autor

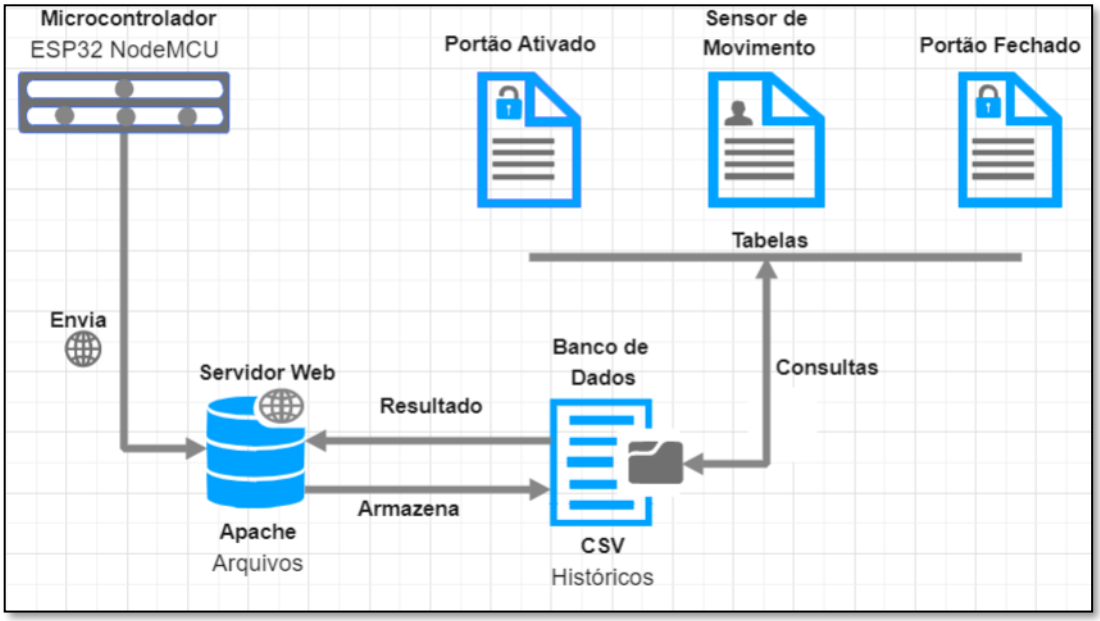
Como mostra a Figura 22, pode-se visualizar a estrutura da maquete finalizada. Em seguida foi feito um acabamento na parte interna da estrutura da maquete do portão, para alinhar os componentes da parte *hardware* alocados. No decorrer deste trabalho são mostrados todos os componentes usados no projeto.

5.3 Desenvolvimento do Sistema do Microcontrolador

Nesse tópico são detalhadas as etapas de desenvolvimento do sistema do microcontrolador, que é a parte do circuito responsável por realizar o controle e o monitoramento do portão. O microcontrolador gerencia e captura os dados que são fornecidos pelos dispositivos sensor, fechadura e motor que posteriormente repassa para o servidor *Web e*, como resposta o servidor retorna a interface *Web* para visualização.

A Figura 23, apresenta o diagrama do sistema do microcontrolador com o servidor e o banco de dados. Maiores detalhes são mostrados nas seções seguintes.

Figura 23: Diagrama do Sistema do Microcontrolador Servidor e Banco de Dados



Fonte: Elaborada pelo autor

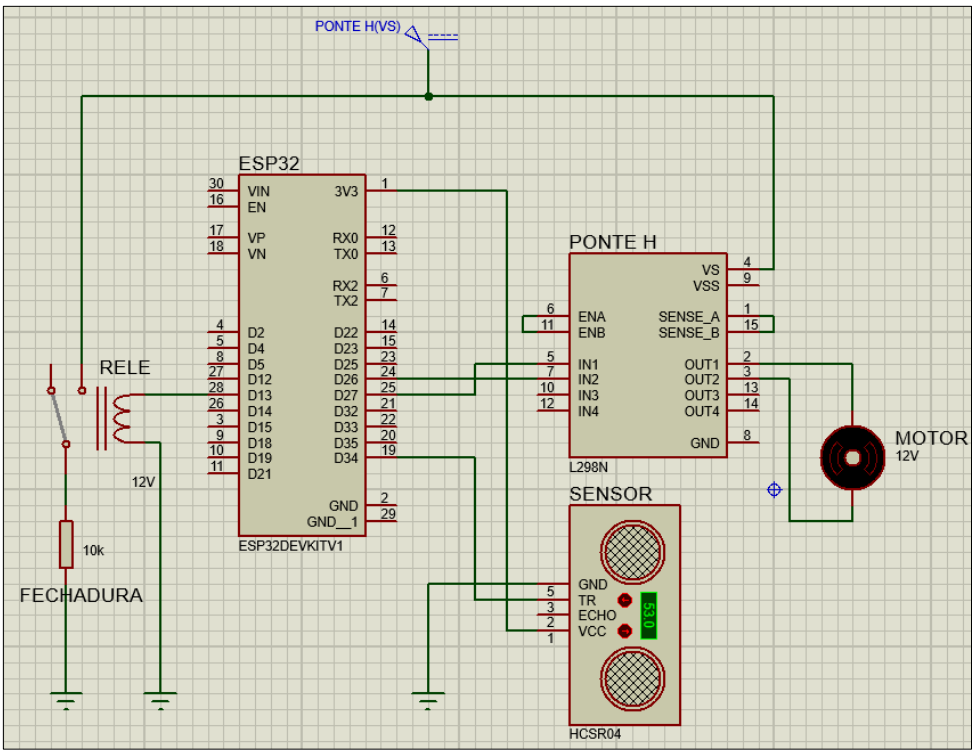
5.3.1 Simulação do Circuito Elétrico

Para simplificar o sistema de funcionamento do portão, foi realizado uma simulação do microcontrolador integrado aos periféricos. A simulação foi feita através do *software Proteus Design Suite*, que é próprio para criação de projetos eletrônicos, composto por

ferramentas que auxiliam no desenvolvimento de esquemático, projeto de circuitos integrados e placas de circuito impresso (MADEIRA, 2015).

O *software* é de uso pago, porém existe a versão de avaliação gratuita, que foi usada nesse trabalho. Essa versão tem uso limitada para algumas ferramentas de auxílio. Com isso, não foi possível usar todos os componentes descritos nesse trabalho. Então, foram usados alguns componentes semelhantes ao original, para representar a ligação do circuito, como no caso da fechadura, que foi usado um resistor para representá-la. O outro componente foi o sensor de movimento substituído por outro de modelo diferente. A Figura 24, apresenta o circuito do microcontrolador e componentes.

Figura 24: Circuito Integrado do Microcontrolador



Fonte: Elaborada pelo autor.

O microcontrolador ESP32 NodeMCU mencionado na seção 4.4.1 possui 25 pinos *General Purpose Input/Output* (GPIOs). Para esta simulação foi utilizado 4 pinos para a comunicação com os dispositivos periféricos.

Como pode-se observar na Figura 24, o primeiro pino D13 na esquerda é responsável pela conexão do microcontrolador ao relé que uma vez alimentado, é acionada sua chave de abertura e ativa o acionamento da fechadura. Já o pino 3,3V é provida pelo próprio microcontrolador que serve para alimentar dispositivos periféricos com necessidade de

alimentação de até 3V, como é o caso do sensor pino VCC, que está recebendo uma alimentação do microcontrolador.

O Segundo pino é o D34, responsável pela troca de informações entre o sensor e o microcontrolador. Quando o sensor detecta o movimento, ele envia a informação para o microcontrolador através da porta TR do sensor ligada no pino D34, responsável por receber os dados.

Os pinos D26 e D27 são responsáveis pelo controle PWM da ponte H que, consequentemente recebe um sinal para controlar a rotação do motor.

O pino é o GND do sensor, é o pino terra, ou seja, negativo 0V do próprio periférico.

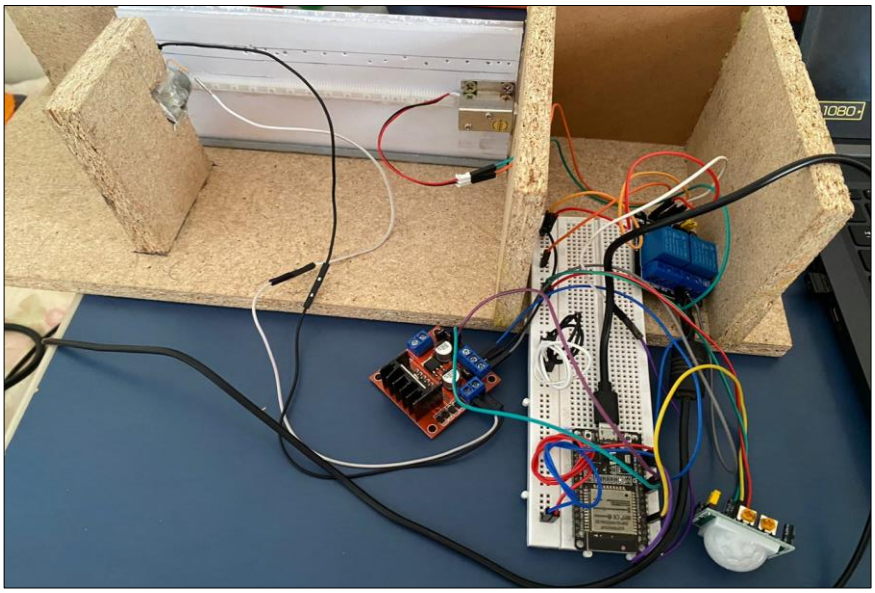
No momento que o pino IN1 recebe um sinal de nível lógico baixo (0), e o pino IN2 recebe um sinal de nível lógico alto (1), o motor tende a girar no sentido horário. Porém, se os níveis lógicos são invertidos o motor gira no sentido anti-horário.

5.3.2 Montagem do Circuito Elétrico

Esta seção apresenta o circuito elétrico do microcontrolador montado na maquete, como pode ser visto na Figura 25.

As ligações do microcontrolador com os periféricos consistem no uso de *jumper*, um fio macho ou fêmea como, para respectivo contato entre dois dispositivos. Para fins didáticos as ligações dos componentes eletrônicos foram feitas utilizando uma *protoboard* conhecida como (placa de ensaio ou matriz de contatos). A sua utilização facilita na montagem de circuitos eletrônicos, realização de testes e representa uma visualização melhor nas ligações dos fios.

Figura 25: Circuito do microcontrolador e Periféricos.



Fonte: Elaborada pelo autor.

Na simulação não houve necessidade de fornecer alimentação para o dispositivo microcontrolador. Contudo, na montagem do circuito, essa situação foi bem diferente. Alguns dos componentes utilizados neste trabalho precisam de alimentação externa de 12V, como é o caso da fechadura eletrônica. Logo, entra o uso da fonte de alimentação externa já citada na seção 4.4.2.3 desse trabalho, A sua utilização se encaixa de forma viável para alimentar os componentes que necessitam de corrente contínua superior a 5V, capaz de fornecer a corrente necessária para o circuito como um todo.

Durante o funcionamento do circuito, de início foi necessário realizar o carregamento do programa para a memória interna do microcontrolador, conectando o microcontrolador via USB no computador ao qual a *Integrated Development Environment*, Ambiente de Desenvolvimento Integrado (IDE) estar instalada. Depois foi feita a compilação e o carregamento do programa para o dispositivo. Feito isso, o programa ficou gravado na memória do microcontrolador. Uma vez que o programa é gravado na memória, não há perigo de perdê-lo, exceto ser for feito o *reset*. Depois disso, o microcontrolador pode funcionar de forma independente, desde que receba alimentação externa.

5.3.3 VS Code (Visual Studio Code)

O *Visual Studio Code* mais conhecido com *VS* Code é um editor de código-fonte leve, poderoso, com bastante recursos para quem trabalha com desenvolvimento em diferentes tipos de linguagem de programação. A plataforma disponibiliza suporte integrado para

JavaScript, TypeScript, Node.js dentre outras linguagens, e uma gama de extensões para outras linguagens e ambientes de execução (VS CODE, 2022).

A plataforma é bem completa, fornece bastante recursos, como fermentas de desenvolvimento, extensões e pacote de bibliotecas, não tendo a necessidade de buscar em outro lugar. O uso desta plataforma foi de grande importância para o desenvolvimento da programação do microcontrolador e da plataforma *web* deste trabalho, sem a necessidade de usar outra *Integrated Development Environment*, Ambiente de Desenvolvimento Integrado (IDE) (VS CODE, 2022).

O PlatformIO IDE é um ambiente de desenvolvimento de *software* integrado, destinado a várias arquiteturas, como por exemplo, arquitetura do microcontrolador ESP32. Trata sobre como gerenciar bibliotecas por estar inserida dentro do *VS* Code, e possibilita usar as mesmas funções de bibliotecas da IDE do Arduino, que são as mesmas usadas na programação de placas de microcontroladores. Uma vez feita a instalação da extensão na plataforma do *VS* Code, esta já está pronta para programar a placa do microcontrolador, como pode-se observar na Figura 26 (VS CODE, 2022).

Figura 26: PlatformIO IDE.

Fonte: VS CODE, 2022.

No caso desse trabalho, como é visto na Figura 26, a extensão já está instalada, e foi de grande importância para poder programar o microcontrolador deste projeto.

A programação desta monografia foi realizada em duas etapas, visto na seção 5.3.4, que descreve toda programação que está gravada na placa da memória do microcontrolador e na seção 5.4, é a parte de todo o código da plataforma *web* e o seu desenvolvimento de forma geral.

5.3.4 Programação

A programação que está gravada na memória do microcontrolador está dividida em três partes. Essa separação facilita o entendimento da composição de cada elemento, descrito na seguinte ordem:

1. Pacote de Bibliotecas;
2. Constantes e Variáveis;

3. Funções;

5.3.5 Pacote de Bibliotecas

As bibliotecas no desenvolvimento de *software* tratam de uma coleção de recursos, como o conjunto de códigos pré-configurados e os dados auxiliares. A utilização de biblioteca específica para cada tipo de função possibilita o funcionamento do programa. Algumas bibliotecas, em particular "*#include <wifi.h>*", foram usadas neste trabalho, como mostra a Figura 27.

Figura 27: Bibliotecas Parte 01.

Fonte: Elaborada pelo autor.

Além da biblioteca "*#include <Wifi.h>*", já citada, foram utilizadas mais quatro bibliotecas, uma biblioteca de controle do motor CD com nome "*#include <ESPS32Servo.h>*", outra do próprio Arduino para comandos básicos, além de mais duas do servidor *web*, responsável por todo o funcionamento do servidor *web* HTTP e *websocket* e por último TCP.

O *webSocket* é um protocolo de comunicação sobre TCP que permite a comunicação bidirecional em uma única conexão entre o cliente e o servidor.

Na comunicação *full-duplex*, o servidor e o cliente podem transmitir e receber dados simultaneamente em tempo real sem nenhum tipo de bloqueio, quando a conexão for estabelecida. O servidor pode enviar quantas solicitações quiser, assim reduzindo a sobrecarga em comparação comunicação *pull* (puxar) do HTTP (BANDURSKI P., 2022).

O HTTP é um protocolo de transferência que possibilita a comunicação entre um navegador e um servidor *web*, ou seja, permite que usuário possa inserir a *Uniform Resource Locator*, Localizador Uniforme de Recursos (URL) no navegador e em seguida exibir conteúdo e dados que nele existem (SOUZA, 2019).

A Figura 28, mostra as credenciais de rede, SSID = "usuário" e PASS = "senha", e servidor.

Figura 28: Credencias de Rede e Servidor.

```
// Credenciais de rede local
const char* ssid = "CLAUDIO_NET";
const char* pass = "[redacted]";

// Criando servidor e socket
AsyncWebServer server(8081);
AsyncWebSocket ws("/ws");
```

Fonte: Elaborada pelo autor.

Para conectar o microcontrolador ESP32 ao *Wi-fi* é necessário colocar as credenciais de rede usuário e senha que são salvas em variáveis globais.

O construtor da classe recebe como entrada o número da porta onde o servidor está escutando as requisições HTTP recebidas. Para este trabalho foi utilizada porta (8081) do servidor *Web*, função "<*asnycWebServer server (8081)*>", e a "<*asnycWebSocket ws("/ws"))*>", utilizada para conexão não criptografada.

5.3.6 Constantes e Variáveis

As constantes são utilizadas para armazenar valores fixos que não podem ser alterados durante a execução do programa. Quando há dados que obrigatoriamente não podem ser modificados ao longo da execução do programa, então os dados devem ser declarados como uma constante (MADHAVI PINGILI, 2019).

A definição de constantes da Figura 29, apresenta a identificação de pinagem do microcontrolador para cada função desejada. Toda a pinagem do microcontrolador pode ser vista na seção 4.4.1 desse trabalho. A identificação é importante porque uma vez que se conhece a função de cada pino, ganha-se tempo e agilidade no decorrer do desenvolvimento. A pinagem do microcontrolador utilizada como pode ver na Figura 29.

Figura 29: Declaração de Constantes.

```
#define BUILTIN_LED 2
#define FECHADURA_GPIO 13
#define MOTOR_GPIO 12
#define SENSOR_GPIO 34
#define LED_GPIO 2
#define MOTOR_DIRECIONAL_CONTROL1 27
#define MOTOR_DIRECIONAL_CONTROL2 26
#define MOTOR_DELAY 85
#define FECHADURA_DELAY 420
#define SENSOR_DELAY 4500

#define ABRIR false
#define FECHAR true
#define ATIVAR true
#define DESATIVAR false
#define MOTOR_ABRIR 0
#define MOTOR_FECHAR 120
```

Fonte: Elaborada pelo autor.

As variáveis são elementos da programação que definem um ou mais valores armazenados e manipulados pelos programas quando está em execução (DANIELA, 2019). A Figura 30, descreve os quartos variáveis usadas nesse projeto.

Figura 30: Variáveis

```
35    void (*outerWrite)(uint8_t, uint8_t);
36    void (*outerAnalogWrite)(uint8_t, int);
37    int (*outerRead)(uint8_t);
38    void (*outerDelay)(uint32_t);
```

Fonte: Elaborada pelo autor.

5.3.7 Funções

A função é um trecho de código que executa alguma tarefa específica, como o procedimento, porém retornando a qualquer momento um resultado ao solicitante. Para se definir uma função deve-se indicar o tipo do retorno da função, seu nome e os seus parâmetros. Dito isso, a função pode ou não retornar um valor (REIS F., 2015).

A Figura 31 apresenta a função do motor e fechadura com identificação de operação do motor "*LOW*" e "*HIGH*" que representa acionamento do motor em nível alto e nível baixo, aberto ou fechado com *delay* (tempo de atraso) no início quando é acionado. Logo, a fechadura é ativada e desativada a cada final de percurso.

Figura 31: Funções.

```
55  void motor(int direction){
56    if(direction > 0){
57      outerWrite(MOTOR_DIRECIONAL_CONTROL1, LOW);
58      outerWrite(MOTOR_DIRECIONAL_CONTROL2, HIGH);
59      outerDelay(MOTOR_DELAY);
60      outerWrite(MOTOR_DIRECIONAL_CONTROL1, LOW);
61      outerWrite(MOTOR_DIRECIONAL_CONTROL2, LOW);
62      outerDelay(FECHADURA_DELAY);
63      fechadura(DESATIVAR);
64    }
65    else {
66      fechadura(ATIVAR);
67      outerDelay(FECHADURA_DELAY);
68      outerWrite(MOTOR_DIRECIONAL_CONTROL1, HIGH);
69      outerWrite(MOTOR_DIRECIONAL_CONTROL2, LOW);
70      outerDelay(MOTOR_DELAY*0.89);
71      outerWrite(MOTOR_DIRECIONAL_CONTROL1, LOW);
72      outerWrite(MOTOR_DIRECIONAL_CONTROL2, LOW);
73    }
74  }
```

Fonte: Elabora pelo auto.

Na Figura 32 é definida a função "<*void setup*()>", vista na linha 143. Essa função é chamada uma única vez quando a tarefa é inicializada no programa e é classificada como a função principal dentro do programa, sem ela o programa não será executado.

A função "<*pinMode*>" especifica a configuração dos pinos, que serve como entrada ou saída para cada parâmetro "<*INPUT*>", e "<*OUTPUT*>".

Figura 32: Void Setup().

```
143  void setup() {
144    ESP32PWM::allocateTimer(0);
145    servoMotor.setPeriodHertz(50);
146    servoMotor.attach(MOTOR_GPIO, 1000, 2000);
147
148    outerWrite = digitalWrite;
149    outerAnalogWrite = analogWrite;
150    outerRead = digitalRead;
151    outerDelay = delay;
152
153    pinMode(FECHADURA_GPIO, OUTPUT);
154    digitalWrite(FECHADURA_GPIO, HIGH);
155    pinMode(SENSOR_GPIO, INPUT);
156    pinMode(LED_GPIO, OUTPUT);
157    pinMode(BUILTIN_LED, OUTPUT);
158    pinMode(MOTOR_DIRECIONAL_CONTROL1, OUTPUT);
159    pinMode(MOTOR_DIRECIONAL_CONTROL2, OUTPUT);
```

Fonte: Elaborada pelo autor.

A comunicação serial é o centro entre o computador e o microcontrolador, na fase teste do desenvolvimento do código. Diante disso, primeiramente deve-se definir a taxa de transferência em *bits* por segundo do microcontrolador. Para isso é necessário inicializar a

comunicação serial usando o comando *Serial.begin(),* que recebe como parâmetro a velocidade da conexão. Após esse comando, insere-se um "<(*delay*1000)>", para seu processamento.

Usou-se a taxa de transmissão do "*<Serial.begin(115200)>",* e como pode ser observado a Figura 33, pode verificar o valor atribuído. Feito isso, as mensagens são enviadas pela comunicação serial usando o comando *Serial.print((("Connecting to WiFi...").*

Figura 33: Configuração Serial – Begin.

```
165   // Configurando a serial
166   Serial.begin(115200);
167
168   // Connecting Wifi
169   WiFi.begin(ssid, pass);
170   while(WiFi.status() != WL_CONNECTED){
171     delay(1000);
172     Serial.println("Connecting to WiFi...");
173   }
```

Fonte: Elaborada pelo autor

A última etapa *"<void loop()>",* representa a conexão do sensor. Quando o microcontrolador é conectado ao Wifi, o sensor é ativado e *Light Emitting Diode*, Diodo Emissor de Luz (LED*)* do microcontrolador é aceso, assim mostra a Figura 34.

Figura 34: Void Loop().

```
188   void loop() {
189     ws.cleanupClients();
190     // Serial.printf("Sensor state: %u\n", digitalRead(SENSOR_GPIO));
191     if(digitalRead(SENSOR_GPIO)){
192       led(ATIVAR);
193       ws.textAll("SENSOR ATIVADO");
194       delay(SENSOR_DELAY);
195       led(DESATIVAR);
196     }
197   }
```

Fonte: Elaborada pelo autor.

5.4 Plataforma *Web*

Esta etapa apresenta código do desenvolvimento da plataforma *Web* e autenticação de usuário, que está dívida em duas partes:

- Código de desenvolvimento da plataforma *Web*;
- Autenticação de usuário;

Para o desenvolvimento da plataforma *Web* foi utiliza o *VS* Code mencionado na seção 5.3.3 desse trabalho. A utilização dessa plataforma foi de grande importância para a realização da *interface*, tendo a necessidade do uso de algumas linguagens de programação

como HTML, Java Scripts, CSS, além de algumas bibliotecas como o React, conforme mostra a Figura 35.

Figura 35: Pacotes de bibliotecas.

```
1  import React, { useState, useEffect, useCallback } from 'react'
2    import logo from './logo.svg';
3    import './App.css';
4    import { useAuth0 } from "@auth0/auth0-react";
```

Fonte: Elaborada pelo autor.

A linha 4 da Figura 35 apresenta a classe que permite usar com "*<react> "* a função da autenticação de usuário e, com isso atribui parâmetros para configurar o aplicativo.

Figura 36: Definição de Constantes.

```
45    const {
46      loginWithRedirect,
47      isAuthenticated,
48      user,
49      isLoading,
50      logout,
51      getAccessTokenSilently,
52    } = useAuth0();
53
54    const [ connLoading, setConnLoading ] = useState<boolean>(false);
55    const [ ip, setIp ] = useState('');
56    const [ ws, setWs ] = useState<null | WebSocket>(null);
57    const [ logs, setLogs ] = useState<Array<{ DATA: string, "CAIXA DE EVENTOS": string }>>([])
58    const [ autorizado, setAutorizado ] = useState<boolean>(false);
```

Fonte: Elaborada pelo autor.

A validação de usuário apresentada na Figura 37, mostra como é feita a permissão de usuário e funciona como uma chave única para cada usuário que tem permissão. O sistema faz a verificação do usuário e identifica como "*<true>*" ou "*<false>*" (verdadeiro ou falso) a permissão validada ou recusada.

Figura 37: Validação de Usuário.


```
{"google-oauth2|114251023303784563039":{"given_name":"Dougllas","family_name":"Silva","nickname":
"denidougllas17","name":"Dougllas Silva","picture":
"https://lh3.googleusercontent.com/a/ALm5wu2J60TkBIVvXE--w6Mj2FR7pnfg07y4oQP5CvnY=s96-c",
"locale":"pt-BR","updated_at":"2022-11-12T00:56:32.555Z","email":"denidougllas17@gmail.com",
"email_verified":true,"sub":"google-oauth2|114251023303784563039","portao":true},"undefined":{}}
```


Fonte: Elaborada pelo autor.

A Figura 38 ilustra-se quando o usuário não tem permissão para acessar a plataforma *Web*.

Figura 38: Permissão Recusada.

```
95          <div>
96            Você não tem permissão
97            <Button variant='contained' color='primary' onClick={ () => logout() }>Logout</Button>
98          </div>
```

Fonte: Elaborada pelo autor.

A Figura 39 mostra como o usuário é autenticado na plataforma *Web*.

Figura 39: Autenticação na Plataforma Web.

```
193                 Autenticado: { isAuthenticated ? 'Sim' : 'Não' }
194               </RegularText>
195               <Button variant='contained' color='primary' onClick={ () => logout() }>Logout</Button>
196
197               <div style={{ height: 380, width: '100%', backgroundColor: blue[400] }}>
198                 <DataGrid
199                   columns={columns}
200                   rows={logs}
201                   getRowId={ row => `${row['DATA']}${parseInt((Math.random()*1000000).toString())}` }
202                   pageSize={5}
203                   rowsPerPageOptions={[5]}
204                   checkboxSelection
205                 />
206               </div>
```

Fonte: Elaborada pelo autor.

5.4.1 Autenticação

A autenticação de usuário para se conectar na plataforma *Web*, teve como grande importância a utilização do *Auth0* é uma plataforma de entidade universal de solução flexível, com foco principal em autorização e autenticação de aplicativos web, móveis e IoT.

A *Auth0* é considerada a plataforma dos desenvolvedores por ter uma gama de recursos utilizável na criação de aplicativos, *Application Programming Interface*, Interface de Programação de Aplicação (APIs) entre outros, destaca-se alguns recursos como: (AUTH0, 2022).

- Autorização e Autenticação de Aplicativos;
- Segurança Adaptável Multicamadas;
- Autenticação Multifator;
- Autenticação sem Senha
- Gerenciamento de Usuários etc.

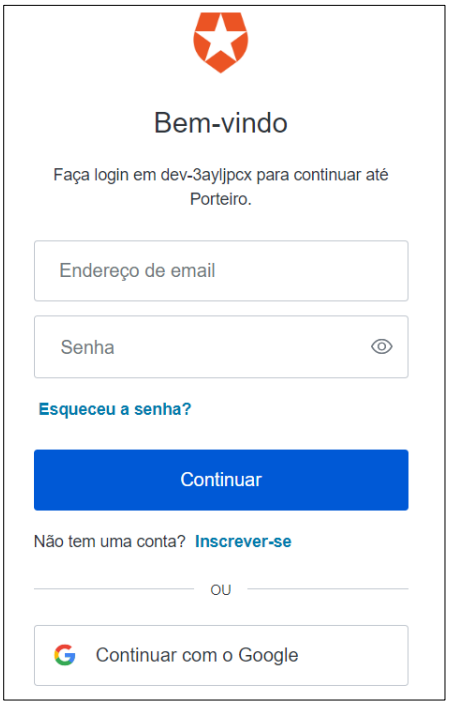
A utilização da plataforma *Auth0* foi muito importante pois permitiu usar esses recursos em poucas linhas de códigos, como é visto na Figura 41, e configurar na aplicação, a autenticação do usuário na plataforma *web* do porteiro eletrônico.

Uma vantagem da *Auth0* é a sincronização de contas de usuários. Uma vez que o usuário esteja logado em uma conta, como por exemplo o Gmail, o Hotmail e o LinkedIn, ela permite ao usuário se conectar na aplicação.

Na primeira tela da Figura 40, é necessário que o usuário realize o cadastro ou se autenticar com a sua conta do Google, desde que esteja logado.

A Figura 40, mostra o primeiro acesso, que é tela de cadastro do usuário

Figura 40: Tela de Cadastro.



Fonte: AUTH0, 2022.

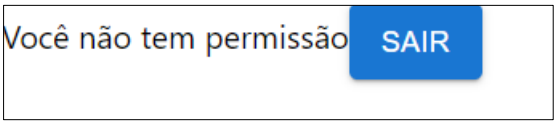
Após a autenticação, o usuário é direcionado para a próxima tela exibida na Figura 41, na qual é feita a autenticação de conta do usuário e determina se usuário tem permissão ou não para se conectar na plataforma.

Figura 41: Autenticação de Conta.

Fonte: AUTH0, 2022.

Caso a autenticação seja recusada, o usuário recebe uma mensagem notificando que não tem permissão para conectar na plataforma, como pode ser visto na Figura 42.

Figura 42: Permissão recusada.



Fonte: Elaborada pelo autor.

Caso a autenticação seja aceita, a permissão é liberada e o usuário está conectado na plataforma *web*, e pronto para monitorar o porteiro eletrônico. A Figura 43, apresenta uma tela com o usuário somente autenticado (sim), mas desconectado da plataforma *Web*.

Figura 43: Plataforma *Web* - Usuário Desconectado – Parte 01.

Fonte: Elaborada pelo autor.

Para realizar o controle de abertura e fechamento do portão, é obrigatório que o usuário digite um endereço IP: 192.168.x.xx e, caso o IP não seja válido, ele recebe uma notificação no canto superior da direita, como ilustra a Figura 44.

Figura 44: Plataforma *Web* – Parte 02.

Fonte: Elaborada pelo autor.

O gerenciamento da plataforma somente é permitido quando o usuário está conectado, o que permite controlar a abertura e fechamento do portão, além de monitorar a captura do sensor de movimento em tempo real.

A Figura 45, apresenta a tela com um usuário autenticado (sim), e conectado (sim) na plataforma *Web*.

Figura 45: Plataforma *W*eb – Usuário Conectado Parte 03.

Fonte: Elaborada pelo autor.

A plataforma desenvolvida permite que seja visualizado em tempo real o *status* do portão capturado pelo sensor e o horário que o portão está sendo gerenciado, através de um historio de eventos, a Figura 46 pode-se observar.

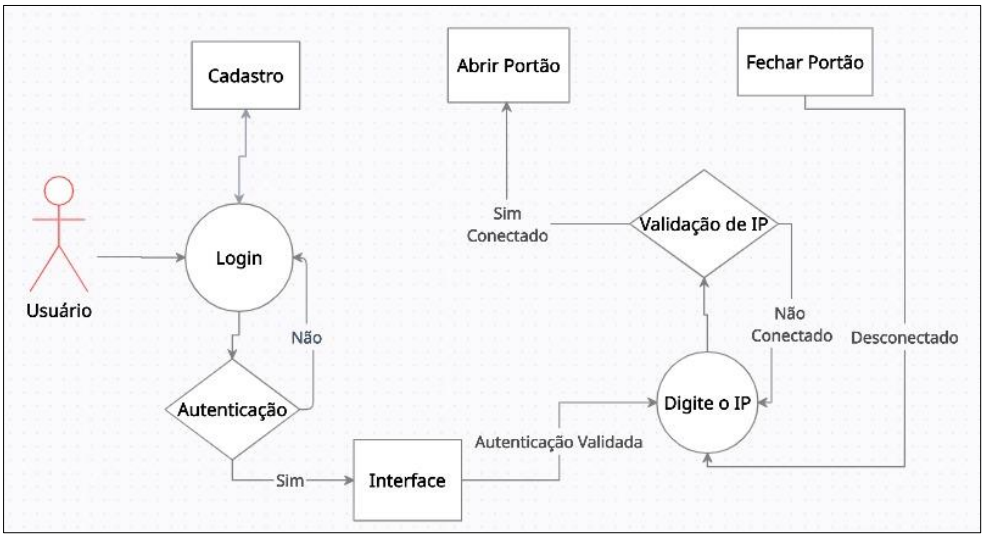
Figura 46: Plataforma *W*eb – Eventos

Fonte: Elaborada pelo autor.

A plataforma dá a possibilidade de se fazer a consulta de eventos anteriores, e classificá-los por categorias como a data, o dia, a hora, dentre outras.

A simulação como qualquer outro modo representativo é de suma importância para que possa entender qual é o objetivo que nela apresenta. Diante disso o diagrama da Figura 47, ilustra como é feita a validação de usuário na plataforma *web*.

Figura 47: Diagrama de Validação de Usuário.



Fonte: Elaborada pelo autor.

5.4.2 Servidor web (web server)

O servidor *web* (*web server*) pode se definir como uma combinação específica do *hardware* e do *software* trabalhando em conjunto.

Na parte do *hardware*, um servidor *web* é um computador ligado 24 horas por dia que armazena arquivos que compõem os *sites* tais como, o código HTML, *scripts* em *JavaScript*, imagens, folhas de estilo, que reporta a entrega para o dispositivo do usuário final (OLIVEIRA, 2017).

Na parte do *software*, o servidor *web* inclui várias partes que controlam como os usuários da *web* acessam os arquivos hospedados, tais como o armazenamento e a disponibilidade, que entende-se com um servidor HTTP. Um servidor HTTP é um *software* que interpreta URLs que são endereços *web* e o HTTP como protocolo que o navegador utiliza para visualizar páginas *web* (TAVARES, 2019).

O servidor *web* trabalha de duas formas: estático e dinâmico.

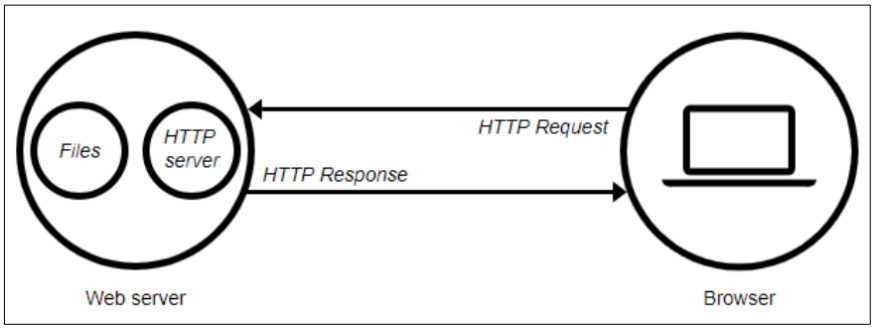
O funcionamento da parte estática consiste em um número de páginas fixo que envia informações ao usuário de forma HTTP. Conforme é modificado o conteúdo da página *web*,

de forma automática o navegador executa essas alterações, o que é exibido diretamente na página do navegador do usuário.

Na parte dinâmica do servidor *web*, os arquivos hospedados são atualizados antes de enviar o conteúdo para o seu navegador por meio do servidor HTTP/HTTPS. Isso possibilita que o servidor gere e entregue o conteúdo dinamicamente para o *browser* (TAVARES, 2019).

A Figura 48, representa como funciona a requisição e solicitação de um servidor *web* com HTTP, que de forma indireta pode ser usuário.

Figura 48: Requisição entre Servidor *Web* e Browser



Fonte: MOZILLA DEVELOPER, 2022.

Essa etapa descreve trechos de código do Servidor *Web*. Para o seu desenvolvimento foram usados alguns pacotes de bibliotecas específicas, como é visto na Figura 49.

O *import* das bibliotecas é de grande importância para a função que expressa um funcionamento do servidor. Por exemplo a biblioteca utilizada na linha 6, permite que o servidor execute qualquer arquivo que tem o formato, extensões CSV.

Figura 49: Bibliotecas Parte 02.

porteiro-server > src > TS app.ts > ...

```
1   // Bibliotecas
2   import express from 'express';
3   import fs from 'fs';
4   import path from 'path';
5   import moment from 'moment';
6   import csvtojson from 'csvtojson';
7   import cors from 'cors';
8
9   import httpProxy from 'http-proxy';
10  import { expressjwt as jwt } from 'express-jwt';
11  import * as jwks from 'jwks-rsa';
12  import dotenv from 'dotenv';
```

Fonte: Elaborada pelo autor.

A função *app.get()* fica responsável pela requisição e pela resposta entre o servidor e o cliente *"<(req, res)>"*, como pode ser observado na Figura 50.

É importante entender que o HTTP tem alguns métodos de comunicação, um deles é o GET usado na implementação do servidor que é implementado por meio da função *app.get()* vista nas linhas 51, 54 e 58 da Figura 50.

O método GET compõe-se de uma solicitação HTTP que é feita no servidor *web*, e responde como parâmetros, os dados que são exibidos na URL. O servidor faz a leitura da requisição e retorna como resposta somente o que foi pedido pelo método.

A constante definida com o nome *port* é o número da porta da camada de transporte que o servidor está usando, conforme pode ser visto na linha 71 da Figura 50.

Figura 50: Função app.get.

```
50  // app.use(jwtCheck);
51  app.get("/", (req, res) => {
52      res.send("Olá eu sou o servidor");
53  });
54  app.get("/pode_usar_portao", (req, res) => {
55      // @ts-ignore
56      return res.json(req.user.portao || false);
57  });
58  app.get("/register/:message", (req, res) => {
59      const { message } = req.params
60      fs.writeFileSync("./log.csv", `\n${moment().format("DD/MM/YYYY HH:mm:ss")};${message}`, { flag: "a+" });
61      res.status(200).send("OK")
62  });
63
64  app.get("/get_logs", (req, res) => {
65      csvtojson({ delimiter: ";" })
66          .fromFile("./log.csv")
67          .then(obj => {
68              res.json(obj);
69          })
70  });
71  const port = 8080;
72
73  app.listen(8080, () => {
74      console.log("Servidor rodando na porta", port);
75  });
```

Fonte: Elaborada pelo autor.

5.4.3 Banco de Dados

O *Database Management Systems*, Sistema de Gerenciamento de Bancos de Dados (SGBD) é um *software* que trabalha para gerenciar o armazenamento de uma base de dados, responsável por controlar, acessar, organizar e proteger as informações de uma aplicação. Os dados são organizados em tabelas, as quais mantêm relacionamentos de acordo com a lógica de organização dos dados (OLIVEIRA, 2017).

A princípio o objetivo do SGBD é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, na manipulação e persistência e organização de dados. O SGBD disponibiliza acesso restrito, somente usuários autorizados podem ter o acesso à *interface* para manipular os dados como incluir, alterar ou consultar dados armazenados.

O SGBD consiste de algumas linguagens especificas como o *Structured Query Language*, Linguagem de Consulta Estruturada (SQL), linguagem responsável por acessar os sistemas de gerenciamento de bancos de dados. Já o *Data Manipulation Language*, Linguagem de Manipulação de Dados (DML) é uma linguagem responsável por localizar, inserir ou atualizar os dados. A *Data Definition Language*, Linguagem de Definição de Dados (DDL) contém a parte da linguagem que define a estrutura dos dados e das tabelas (OLIVEIRA, 2017).

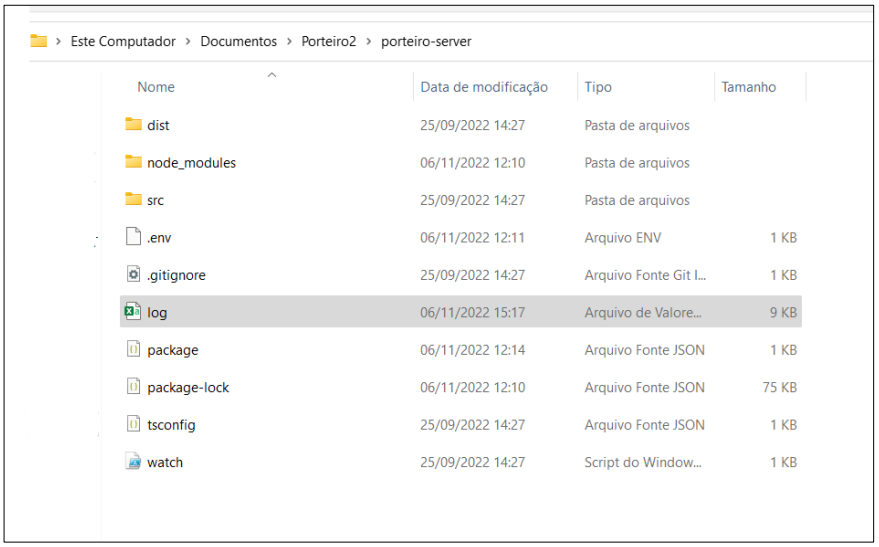
O armazenamento de dados deste trabalho, é feita no próprio servidor WEB, que armazena os dados no formato de arquivo *Comma Separated Values*, Valores Separados Por Vírgulas (CSV).

O arquivo CSV é um formato de arquivo de texto que armazena informações em forma de planilhas, tabelas que pode ter números e datas. Os arquivos CSV podem ser facilmente importados e exportados usando programas que armazenam dados em tabelas.

Os dados CSV são salvos no próprio computador que está rodando o servidor *web*, e pode ser visualizado de forma simples e, podendo abrir em qualquer *software* que executa arquivo com extensões CSV, por exemplo o Excel. Esses dados podem ser vistos também na plataforma WEB, como histórico de eventos do portão.

A Figura 51, apresenta o local onde o arquivo está salvo e seu formato.

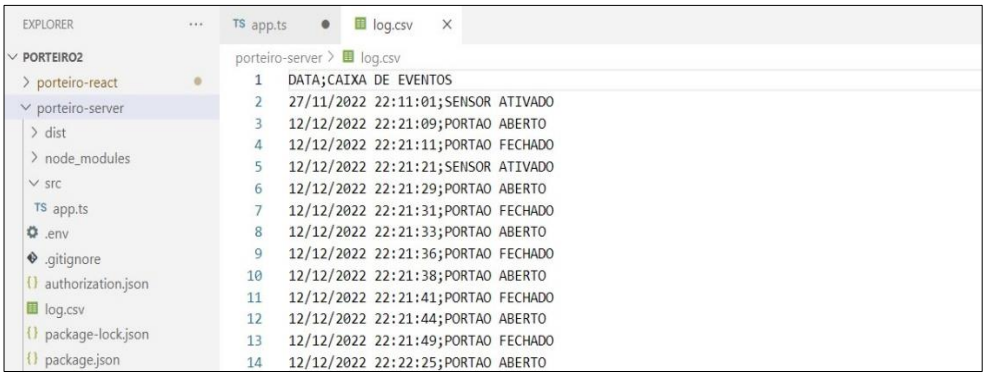
Figura 51: Pasta de Históricos.



Fonte: Elaborado pelo autor.

Na própria plataforma de desenvolvimento do VS Code, o servidor faz o armazenamento dos dados e disponibiliza para consulta as requisições que o cliente solicita, como é visto na Figura 52.

Figura 52: Históricos.



Fonte: Elaborada pelo autor.

6 RESULTADOS E TESTES

Neste capítulo são abordadas as substituições que foram feitas, as dificuldades encontradas durante a implementação do projeto, os testes e resultados alcançados.

Foi feita nesse trabalho toda a montagem do modelo representativo do porteiro eletrônico (maquete), implementação do microcontrolador, criação do servidor e da plataforma *web*, concluindo-se todas as etapas do porteiro eletrônico previstas nos objetivos iniciais. Com o projeto pronto foram realizados vários testes que permitiram apresentar os resultados alcançados.

6.1 Dificuldades encontradas

Na primeira fase do projeto houve várias dificuldades. A primeira delas foi a substituição do motor de rotação de abertura do portão. No planejamento inicial, a ideia seria abertura do portão na forma basculante, até se perceber que seriam necessários de mais recursos na parte do *hardware*, o que elevaria o custo financeiro maior, para se pudesse adaptar a estrutura, uma vez que o uso do servo motor não seria adequado para o tipo de operação. Após pesquisar em outros trabalhos feitos com características semelhantes, foi definida e alterada para um portão deslizante já que se possuía todos os materiais para sua implementação, exceto a cremalheira.

A segunda dificuldade encontrada foi a escolha de uma cremalheira que não estava no planejamento inicial. Essa peça foi indispensável para a estabilidade da abertura e fechamento do portão deslizante. Dependendo da necessidade pode-se adaptar a cremalheira em várias posições no portão. Nesse trabalho optou-se por colocar a cremalheira na posição central no meio do portão com ajuda de cola quente e assim, dar mais estabilidade na estrutura do portão.

Junto da cremalheira a ponte H – L298N foi outro periférico que surgiu durante a construção do projeto, com grande importância servindo como intermediário entre o motor e o microcontrolador.

6.2 Testes dos Periféricos Eletrônicos

Teste 1: Microcontrolador

O primeiro teste realizado foi o microcontrolador ESP32, em que se pode verificar o funcionamento das portas GPIOs e testar o Wi-Fi do ESP32.

Teste 2: Motor

A primeira substituição foi o micro servo motor SG90 para o motor DC 45RPM. O motivo da substituição se deu pelo fato do servo motor apresentar um giro curto na rotação do eixo, embora sua descrição apresentasse 180° de rotação. Em um dos testes feitos do servo motor chegou-se a alimentar com tensão até superior que sua capacidade, mesmo assim sem sucesso.

Teste 3: Sensor de Movimento

A segunda substituição realizada foi a troca do mini sensor de movimento PIR HC-SR505 para o sensor de movimento PIR HC-SR501. Sua substituição se deu por necessidades do sensor, que tinha ajustes de sensibilidade e tempo o que garantiu mais segurança e precisão na captura dos dados. Isso porque o mini sensor de movimento não tem essas características e apresentou oscilações nas coletas dos dados deixando vulnerável a segurança da residência caso o use, como apresentou nos testes feitos durante a implementação.

Teste 4: Fechadura Elétrica

O teste da fechadura elétrica deu mais trabalho por ser o componente que necessita de 12V de alimentação para funcionar. Então, teve-se a ideia de usar uma fonte de alimentação externa mencionada na seção 4.4.2.3 desse trabalho. Quanto ao uso desta fonte, esta foi viável e não houve custo. Foi feita a alimentação com sua energia, que em um primeiro momento não funcionou, até se percebeu mal contato do *jumper* que foi ajustado e, em seguida, foi realizada mais uma tentativa que apresentou sucesso.

Avaliação dos testes

Quanto aos testes, estes foram divididos para cada componente. Os testes individualizados, quebraram a possiblidade de alguns componentes apresentarem defeito ou até mesmo mal contato de funcionamento. Após a conclusão dos testes, foram integrados todos os componentes no microcontrolador ESP32, para a criação do sistema do circuito elétrico.

O código usado nos testes dos componentes foram os mesmo para o desenvolvimento do sistema, uma vez que sua adaptação é necessária para um melhor funcionamento de todos.

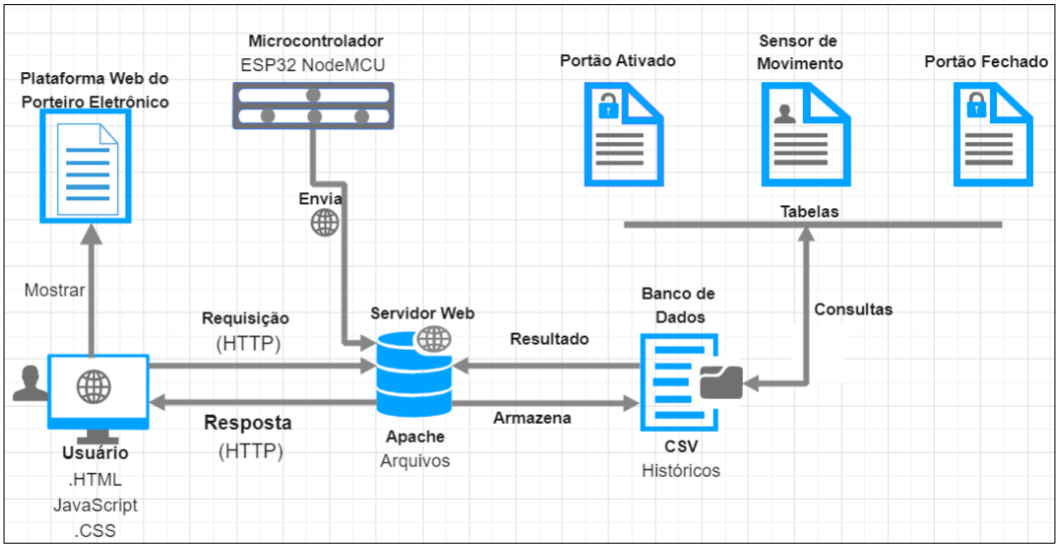
6.3 Resultados Alcançados

Quantos aos resultados alcançados, foram satisfatórios pois apresentam um projeto com funcionamento relativamente viável com grande possibilidade de crescimento e aplicação em uma residência que tenha a necessidade de um porteiro eletrônico.

O diagrama da Figura 53, apresenta o funcionamento do porteiro eletrônico como um todo. É importante destacar as etapas que trabalham em paralelo enviando informações em tempo real. O *delay* (tempo de atraso) de aproximadamente 6 segundos é considerado relativamente baixo, de acordo com processamento das informações, até serem exibido na plataforma, conforme se descreve:

- Dados capturado pelo microcontrolador é enviado por servidor web;
- Dados é armazenado como arquivo *CSV* no banco de dados;
- Dados são apresentados em forma de tabelas;
- Usuário faz uma requisição, logo o servidor retorna como resposta dados armazenado no banco de dados e exibe na plataforma *web*;

Figura 53: Diagrama de Fluxo de Funcionamento do Porteiro Eletrônico



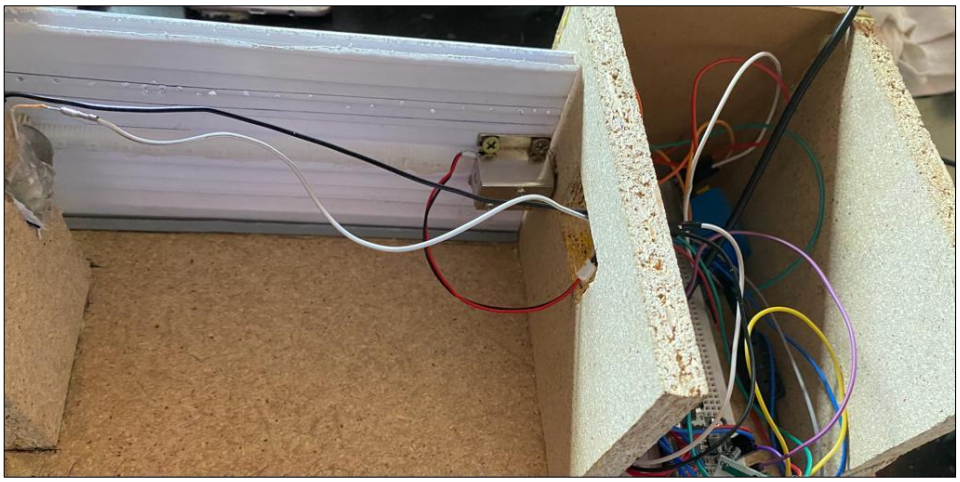
Fonte: Elaborada pelo autor.

O sistema mostrou resultados satisfatórios, com desempenho de funcionamento relativamente alto entres os dispositivos eletrônicos, e suficiente para alcançar os objetivos do trabalho.

A parte do ambiente de testes consiste em um modelo de representativo do portão eletrônico (maquete). A maquete é dividida em um compartilhamento do lado esquerdo onde é usado para posicionar e organizar o microcontrolador e alguns periféricos, também existe

uma saída na lateral do lado de dentro do portão para posicionar a fiação que liga o microcontrolador com o motor e a fechadura, como ilustra a Figura 54.

Figura 54: Porteiro Parte Interna.



Fonte: Elaborada pelo autor.

7 CONSIDERAÇÕES FINAIS

Este projeto teve como objetivo a criação de um protótipo de sistema residencial automatizado de um porteiro eletrônico, com foco principal em realizar decisões, através da leitura de sensor e controle de comandos do microcontrolador, promovendo aspectos relacionado à praticidade e comodidade do usuário no acesso à residência.

Para alcançar os objetivos deste trabalho submeteu-se o estudo teórico sobre automação residencial atribuído a microcontroladores com uso de IoT. A alta evolução de ambiente automatizado caracteriza a necessidade de criação de projetos que possam potencializar a automação de uma residência garantido proporcionar segurança e acessível para o dia-a-dia das pessoas. A proposta deste trabalho surgiu para proporcionar praticidade no controle e monitoramento do portão da residência, sem a necessidade da presença física no ambiente.

Quanto às ferramentas VS Code, *Auth0*, utilizou-se de versões gratuitas, que forneceram o suporte necessário para a criação do projeto, garantido o atendimento de todos os objetivos do trabalho. As mudanças e substituições que ocorreram durante a construção do projeto foram todas necessárias e possibilitaram como um todo o funcionamento do sistema relativamente viável para este tipo de aplicação, embora ainda apresente pontos de melhoria.

A realização dos testes possibilitou verificar e validar a execução da aplicação correspondente ao objetivo proposto e apresentaram resultados eficientes que permitiram responder à questão de pesquisa, sendo possível ao usuário controlar e monitorar a abertura e o fechamento do portão eletrônico.

O presente trabalho possibilitou estudo em bibliografias com vários conhecimentos integrados na área de automação residencial aplicada a comunicação de dispositivos IoT.

De modo geral, o propósito do projeto mostrou-se promissor para outras aplicações desse mesmo tipo, como aberturas de portas e janelas, levando em consideração a segurança, conforto e praticidade quando o assunto é automação residencial. Na correria as pessoas sempre buscam alternativas que possam proporcionar de algum modo agilidade na execução de suas tarefas para aquele momento. A necessidade de se encontrar ferramentas para auxiliar e apoiar, a utilização de tecnologias avançadas muitas vezes está ligada a convivência e garantindo segurança para todos que a usam.

7.1 Sugestões Para Trabalhos Futuros

Como sugestão de trabalhos futuros é recomendado que ocorra um alargamento e aperfeiçoamento do projeto já criado:

- Implementação de uma câmera para reconhecimento facial, aprimorando mais segurança no acesso a residência;
- Hospedagem do servidor *web* em plataforma externa garantido mais espaço para armazenamento de dados e escalabilidade;
- Criptografia dos históricos de dados capturado pelo sensor de movimento;
- Desenvolvimento de um aplicativo;

REFERÊNCIAS

ATZORI, L., Iera, A., e Morabito, G. (2010) "**The *Internet* of Things: A survey**". Computer Networks Journal, Vol. 54, Ed. 15. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1389128610001568?via%3Dihub/. Acesso em 24 mai. 2022.

ASHTON, Kevin. **That '*Internet* of Things' Thing. RFID Journal**. Publicado em: jun. 2009. Disponível em://www.rfidjournal.com/articles/view?4986/. Acesso em: 23 mai. 2022.

AUTH0. **Visão geral do Auth0**. 2022. Disponível em: https://auth0.com/docs/get-started/auth0-overview. Acesso em: 10 de nov. 2022.

ARDUINO&CIA. **Como usar um sensor de presença PIR como Arduino**. 2014. Disponível em: https://www.arduinoecia.com.br/sensor-presenca-arduino-modulo-pir-dyp-me003/. Acesso em 31 out. 2022.

BANDURSKI P. **Publicar WebSocket na camada de experiencia. Dzone**, 2022. Disponível em: https://dzone.com/articles/publish-web-socket-in-the-experience-layer. Acesso em: 08 de nov. 2022.

BBC, NEWS. **3 grandes vantagens do 5G que mudarão para sempre nossa experiência na *Internet*.** 2019. Disponível em: https://www.bbc.com/portuguese/geral-48499353. Acesso em: 05 nov. 2022.

CLARK et al., 2021. Tussle in Cyberspace: **Definindo a Internet do Amanhã**. IEEE/ACM Transactions on Networking, 13(3):462–475.

CAVALLI, Olga. ***Internet* das coisas e inovação na América Latina.** [S.l.: s.n.], 2016. Mimeogr.

CHASE, Otávio. **Sistemas Embarcados.** 2007. Disponível em: http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf. Acesso em 24 mai. 2022.

CISCO. **O que é um access point?**. 2022. Disponível em: https://www.cisco.com/c/pt_br/solutions/small-business/resource-center/networking/what-is-access-point.html. Acesso: 04 de nov. 2022.

CISCO, IBSG. **A *Internet* das coisas como a próxima evolução da *Internet* está mudando tudo.** White paper month = april, Cisco *Internet* Business Solutions Group, 2011. Disponível em: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

DIAS, Renata Rampim de Freitas**.** ***Internet*** **das coisas sem mistérios:uma nova inteligência para os negócios**. São Paulo: Netpress Books, 2016.

DANIELA. **O que são variáveis,** 2019**. DEVMEDIA.** Disponível em:https://www.devmedia.com.br/o-que-sao-variaveis/40728. Acesso em: 09 de nov. 2022.

ESPRESSIF systems IOT Team. **Esp32 nodemcu**, 2021. Disponível em: https:// esp32_datasheet_en.pdf (espressif.com). Acesso em: 12 mai. 2022.

FERNANDO k. **ESP32: Detalhes internos e pinagem.** 2018. Disponível em: https://www.fernandok.com/2018/03/esp32-detalhes-internos-e-pinagem.html/. Acesso em: 24 mai. 2022.

FILIPEFLOP. **Micro Servo MG90S TowerPro**, 2021. Disponível em: https://www.filipeflop.com/produto/micro-servo-mg90s-towerpro/. Acesso em: 14 mai. 2021.

ILUMINIM. **Fonte de Alimentação: O que é e para que serve?**. 2022. Disponível em: https://blog.iluminim.com.br/fonte-de-alimentacao-o-que-e-e-para-que-serve/. Acesso em: 02 nov. 2022.

SOPRANO. **Fechadura Elétrica: Conheça os Diferenciais do Produto**. 03 dez. 2020. Disponível em: https://www.soprano.com.br/blog/fechadura-eletrica-conheca-os-diferenciais-do-produto. Acesso em: 11 mai. 2022.

JUCA, Sandro. **Aplicações Práticas de sistemas embarcados Linux utilizando Raspberry Pi** [recurso eletrônico] / Sandro Jucá e Renata Pereira. 1ª ed. - Rio de Janeiro: PoD, 2018.

MAGRANI, Eduardo. **A internet das coisas** / Eduardo Magrani. 1ª ed. - Rio de Janeiro: FGV Editora, 2018.

MANYIKA. J. et al. **A *Internet* das coisas: mapeando o valor além do hype**. Technical report, Mckinsey Global Institute, 2015. Disponível em: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-*Internet*-of-things-the-value-of-digitizing-the-physical-world/pt-br. Acesso em: 18 abr. 2022.

MOZILLA DEVELOPER. **O que é um servidor *web* (*web* server)?.**2022. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Common_questions/What_is_a_web_server. Acesso em: 07 de nov. 2022.

MUNDOELETRÔNICO. **Arduino L298 H-bridge Motor Driver Board 2a Módulo Ponte H.** 2022. Disponível em: https://www.mundoeletronica.com.br/produtos/arduino-l298-h-bridge-motor-driver-board-2a-modolu-ponte-h/. Acesso em: 31 de out. de 2022.

MINERVA, Roberto; BIRU, Abyi; ROTONDI, Domenico. **Towards a Definition of the *Internet* of Things (IoT).** IEEE *Internet* Initiative - Telecom Italia. 27 maio 2015. Disponível em: <https://pt.scribd.com/doc/306069323/IEEE-IoT-Towards-Definition-*Internet*-of-Things-Revision1-27MAY15>. Acesso em: 20 de abril. 2022.

MIYADAIRA, A. N. **Microcontroladores PIC18:** Aprenda e Programe em Linguagem C. 4. ed. São Paulo: Érica, v. 1, 2009.

MADHAVI PINGILI. **Constante. O Tutor 4U,** 2019. Disponível em: https://maestrovirtuale.com/constante-programacao-conceito-tipos-exemplos/. Acesso em: 09 de nov. 2022,

MADEIRA, D. **Proteus: Manual para simulação do Arduino. Embarcados**, 2015. Disponível em: Acesso em: 04 nov. 2022.

OLIVEIRA, Sergio. ***Internet* das Coisas com ESP8266, Arduino e Raspberry PI.** 1. ed. São Paulo: Novatec Editora Ltda, 2017.

ODATA. **Conectividade: entenda porque é a potência que está movendo a era digital**. 2022. Disponível em: https://odatacolocation.com/blog/conectividade/. Acesso em: 05 de nov. 2022.

PAULUS, G. B. et al. (2017). **Sistema de Automação Residencial: Acessibilidade no controle.** XXV Seminário de Iniciação Científica: Salão do Conhecimento. Universidade Regional do Noroeste do Estado do Rio Grande do Sul. 13 p.

PINTO. **Conectividade em IoT: tecnologias que garantem a efetividade das soluções.** 2020. Disponível em: https://v2com.com/2020/11/26/conectividade-em-iot-inteligencia-dispositivos/. Acesso em: 04 de nov. 2022.

SILVEIRA, C. (2016). **O Que é Indústria 4.0 e Como Ela Vai Impactar o Mundo.** Fonte: Citisystems: Disponível em https://www.citisystems.com.br/industria-4-0/. Acesso em: 25 fev. 2022.

REIS F. (2015). **27 – Lógica de Programação – Funções.** Disponível em: http://www.bosontreinamentos.com.br/logica-de-programacao/27-logica-de-programacao-funcoes/. Acesso em: 08 de nov. 2022.

SILVEIRA, C. (2016). **Sensor: você sabe o que é quais os tipos?** Fonte: Citisystems: Disponível em https://www.citisystems.com.br/sensor-voce-sabe-que-quais-tipos/. Acesso em: 13 mai. 2022.

SILVA, José Jakson. **Casa Inteligente e *Internet* das Coisas (IoT) – Sustentabilidade, Economia e Segurança** / José Jakson da Silva – 1ª Edição – São Paulo, Editora Literando: 2021.

SKETCHUP. **Onde ótimas ideias funcionam**. 2022. Disponível em: https://www.sketchup.com/pt-BR/. Acesso em: 24 mai. 2022.

SOUZA, I. D. **Entenda o que é HTTP e o quão importante esse protocolo é para o seu site. Rockcontent**, 2019. Disponível em: https://rockcontent.com/br/blog/http/ Acesso em: 08 de nov. 2022.

SANTOS. et al. ***Internet* das Coisas: da Teoria à Prática. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, p. 50, 2016.

TARGA, Marcelo Santos; SILVA, Marcos Cleve; CEZAR, Vicente Rodolfo Santos. **Uso de microcontrolador Arduino para a determinação da permeabilidade do solo.** Revista Técnica Ciências Ambientais, v. 1, n. 1, p. 1-14, 2019.

TAVARES, L. **O que é um Servidor *Web* (*Web* Server). Melhores Hospedagem de Sites,** 2019. Disponível em: https://www.melhoreshospedagemdesites.com/servidor-*web*/. Acesso em: 04 nov. 2022.

USINAINFO, Eletrônica & Robótica. **Módulo Relé 5V 10A 2 Canais com Optoacopladores**. 2022. Disponível em: https://www.usinainfo.com.br/rele-arduino/modulo-rele-5v-10a-2-canais-com-optoacopladores-2300.html. Acesso em: 10 mai. 2022.

USINAINFO, Eletrônica & Robótica. **Fechadura Elétrica Solenoide 12V NF Compacta FEC-91=Lingueta Superior**. 2022. Disponível em: https://www.usinainfo.com.br/mini-fechadura-eletrica-solenoide/fechadura-eletrica-solenoide-12v-nf-compacta-fec-91-lingueta-superior-8073.html. Acesso em: 11 mai. 2022.

VSCODE. **Código do Visual Studio.** 2022. Disponível em: https://code.visualstudio.com/docs. Acesso em: 05 de nov. 2022.

WAZLAWICK, Raul Sidnei. **Metodologia de Pesquisa para Ciência da Computação**. 2. ed. Rio de Janeiro: Elsevier Editora Ltda, 2014.

YOUSIF. et al. **IoT Technologies during and Beyond COVID-19:** A Comprehensive Review. Future *Internet*, v. 13, n. 105, 2021. Disponível em: https://doi.org/10.3390/fi13050105. Acesso em: 25 fev. 2022.

APÊNDICE A – CÓDIGO DO MICROCONTROLADOR

main.cpp

```
// Bibliotecas
#include <ESP32Servo.h>
#include <Arduino.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
#include <WiFi.h>

Servo servoMotor;
// Credenciais de rede local
const char* ssid = "CLAUDIO_NET";
const char* pass = "[redacted]";

// Criando servidor e socket
AsyncWebServer server(8081);
AsyncWebSocket ws("/ws");

#define BUILTIN_LED 2
#define FECHADURA_GPIO 13
#define MOTOR_GPIO 12
#define SENSOR_GPIO 34
#define LED_GPIO 2
#define MOTOR_DIRECIONAL_CONTROL1 27
#define MOTOR_DIRECIONAL_CONTROL2 26
#define MOTOR_DELAY 85
#define FECHADURA_DELAY 420
#define SENSOR_DELAY 4500

#define ABRIR false
#define FECHAR true
#define ATIVAR true
#define DESATIVAR false
#define MOTOR_ABRIR 0
#define MOTOR_FECHAR 120

void (*outerWrite)(uint8_t, uint8_t);
void (*outerAnalogWrite)(uint8_t, int);
int (*outerRead)(uint8_t);
void (*outerDelay)(uint32_t);

// #region
void fechadura(bool operation){
  if(!operation)
    outerWrite(FECHADURA_GPIO, HIGH);
  else
```

```
    outerWrite(FECHADURA_GPIO, LOW);
}

void led(bool operation){
  if(operation)
    outerWrite(LED_GPIO, HIGH);
  else
    outerWrite(LED_GPIO, LOW);
}

void motor(int direction){
  if(direction > 0){
    outerWrite(MOTOR_DIRECIONAL_CONTROL1, LOW);
    outerWrite(MOTOR_DIRECIONAL_CONTROL2, HIGH);
    outerDelay(MOTOR_DELAY);
    outerWrite(MOTOR_DIRECIONAL_CONTROL1, LOW);
    outerWrite(MOTOR_DIRECIONAL_CONTROL2, LOW);
    outerDelay(FECHADURA_DELAY);
    fechadura(DESATIVAR);
  }
  else {
    fechadura(ATIVAR);
    outerDelay(FECHADURA_DELAY);
    outerWrite(MOTOR_DIRECIONAL_CONTROL1, HIGH);
    outerWrite(MOTOR_DIRECIONAL_CONTROL2, LOW);
    outerDelay(MOTOR_DELAY*0.89);
    outerWrite(MOTOR_DIRECIONAL_CONTROL1, LOW);
    outerWrite(MOTOR_DIRECIONAL_CONTROL2, LOW);
  }
}

int lerSensor(){
  return outerRead(SENSOR_GPIO);
}
// #endregion

// bool ledState = false;

void notifyAllClients(){
  // ws.textAll(String(ledState));
}

bool ledState = false;

void handleWebSocketMessage(void *arg, uint8_t *data, size_t len){
  AwsFrameInfo *info = (AwsFrameInfo*)arg;
  if(info->final && info->index == 0 && info->len == len && info->opcode ==
WS_TEXT){
```

```
    data[len] = 0;
    if(strcmp((char*)data, "toggleLed") == 0){
      ledState = !ledState;
    }
    if(strcmp((char*)data, "ABRIR PORTAO") == 0){
      motor(ABRIR);
      ws.textAll("PORTAO ABERTO");
    }
    if(strcmp((char*)data, "FECHAR PORTAO") == 0){
      motor(FECHAR);
      ws.textAll("PORTAO FECHADO");
    }
  }
}

void onevent(
  AsyncWebSocket *server,
  AsyncWebSocketClient *client,
  AwsEventType type,
  void *arg,
  uint8_t *data,
  size_t len
){
  switch (type){
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from %s\n", client-
>id(), client->remoteIP().toString().c_str());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client->id());
      break;
    case WS_EVT_DATA:
      handleWebSocketMessage(arg, data, len);
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      Serial.printf("Some error occured\n");
      break;
  }
}

void initWs(){
  ws.onEvent(onevent);
  server.addHandler(&ws);
}

String processor(const String& var){
  return String();
```

```
}

const char html[] PROGMEM = R"rawliteral(<h1>O ESP está em
funcionamento</h1>)rawliteral";

void setup() {
  ESP32PWM::allocateTimer(0);

  // ESP32PWM::allocateTimer(1);
  // ESP32PWM::allocateTimer(2);
  // ESP32PWM::allocateTimer(3);
  servoMotor.setPeriodHertz(50);
  servoMotor.attach(MOTOR_GPIO, 1000, 2000);

  outerWrite = digitalWrite;
  outerAnalogWrite = analogWrite;
  outerRead = digitalRead;
  outerDelay = delay;

  pinMode(FECHADURA_GPIO, OUTPUT);
  digitalWrite(FECHADURA_GPIO, HIGH);
  pinMode(SENSOR_GPIO, INPUT);
  pinMode(LED_GPIO, OUTPUT);
  pinMode(BUILTIN_LED, OUTPUT);
  pinMode(MOTOR_DIRECIONAL_CONTROL1, OUTPUT);
  pinMode(MOTOR_DIRECIONAL_CONTROL2, OUTPUT);

  // Configurando a serial
  Serial.begin(115200);

  // Connecting Wifi
  WiFi.begin(ssid, pass);
  while(WiFi.status() != WL_CONNECTED){
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.printf("Conectado no WiFi pelo IP: %s\n",
WiFi.localIP().toString().c_str());

  initWs();

  // Root route
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *req){
    req->send_P(200, "text/html", html, processor);
  });

  // Starting server
```

```
  server.begin();
}

void loop() {
  ws.cleanupClients();
  // Serial.printf("Sensor state: %u\n", digitalRead(SENSOR_GPIO));
  if(digitalRead(SENSOR_GPIO)){
    led(ATIVAR);
    ws.textAll("SENSOR ATIVADO");
    delay(SENSOR_DELAY);
    led(DESATIVAR);
  }
}
```

APÊNDICE B – CÓDIGO DA PLATAFORMA WEB

app.tsx

```
import React, { useState, useEffect, useCallback } from 'react';
import logo from './logo.svg';
import './App.css';
import { useAuth0 } from "@auth0/auth0-react";

import { styled } from '@stitches/react';
import { blue } from '@mui/material/colors';
import { greenA, redA, blueDark } from '@radix-ui/colors';
import Button from '@mui/material/Button';
import { TextField } from '@mui/material';
import { ToastContainer, toast } from 'react-toastify';
import { BeatLoader } from 'react-spinners';
import axios from 'axios';
import { DataGrid, GridColDef, GridValueGetterParams } from '@mui/x-data-
grid';


import {
  MainContainer,
  Container,
  Title,
  RegularText,
  OpenCloseButtonsContainer,
  IpContainer,
} from './components';


const columns: GridColDef[] = [
  {
    field: "DATA",
    headerName: "DATA",
    type: "string",
    width: 180,
  },
  {
    field: "CAIXA DE EVENTOS",
    headerName: "HISTÓRICO",
    type: "string",
    width: 500,
  }
]


function App() {
```

```
  const {
    loginWithRedirect,
    isAuthenticated,
    user,
    isLoading,
    logout,
    getAccessTokenSilently,
  } = useAuth0();

  const [ connLoading, setConnLoading ] = useState<boolean>(false);
  const [ ip, setIp ] = useState('');
  const [ ws, setWs ] = useState<null | WebSocket>(null);
  const [ logs, setLogs ] = useState<Array<{ DATA: string, "CAIXA DE
EVENTOS": string }>>([]);
  const [ autorizado, setAutorizado ] = useState<boolean>(false);

  useEffect(() => {
    // console.log('isAuthenticated', isAuthenticated);
    (async () => {
      try{
        // console.log(
        //   await getAccessTokenSilently()
        // )
      }
      catch(err){
        console.error("FAILED TO OBTAIN TOKEN");
        console.error(err);
      }
    })()
  }, []);

  useEffect(() => {
    (async () => {
      if(isAuthenticated){
        setLogs(await axios.get("http://localhost:8080/get_logs", { headers:
{ 'X-User': JSON.stringify(user) } }).then(res => res.data));
        setAutorizado(await
axios.get("http://localhost:8080/pode_usar_portao", { headers: { 'X-User':
JSON.stringify(user) } }).then(res => res.data));
      }
    })()
  }, [isAuthenticated, user]);

  if(isLoading){
    return <h1>Carregando...</h1>
  }
```

```
  if(!isAuthenticated){
    loginWithRedirect();
    return <></>
  }

  return !autorizado ?
    (
      <div>
        Você não tem permissão
        <Button variant='contained' color='primary' onClick={ () => logout()
}>Logout</Button>
      </div>
    ) :
    (
      <MainContainer>
        <ToastContainer />
        <style>
          {`
          .textCaptalize{
            text-transform: capitalize !important;
          }
          `}
        </style>
        <Title>Porteiro Eletrônico</Title>
        <Container>
          <RegularText>
            Bem vindo, { user?.name || '' }.<br/>
            Gerencie a portaria em nossa interface universal
          </RegularText>
          <OpenCloseButtonsContainer>
            <Button variant='contained' disabled={ !ws } onClick={ _ =>
ws?.send('ABRIR PORTAO') }>Abrir</Button>
            <Button variant='contained' color='error' disabled={ !ws }
onClick={ _ => ws?.send('FECHAR PORTAO') }>Fechar</Button>
          </OpenCloseButtonsContainer>

          <IpContainer>
            <TextField style={{ width: '100%' }} label='IP:'
variant='filled' onChange={ evt => setIp(evt.target.value) } value={ip} />
            <Button disabled={ !!ws } className='textCaptalize' size='small'
variant='contained' color='success' onClick={ evt => {
              if(connLoading) return;
              setConnLoading(true);
              if(ws){
                ws.close();
              }

              if(!ip || !ip.match(/(\d+\.){3}\d+/)){
```

```
                setConnLoading(false);
                return toast.error("Digite um ip válido. Exemplo:
192.168.X.Y", { theme: 'colored' });
              }


              console.log("Trying connecting a new websocket");
              const websocket = new WebSocket(`ws://${ip}:8081/ws`);
              websocket.onopen = evt => {
                if(evt.type == "open"){
                  setWs(websocket);
                  setIp('');
                  toast.success(`Conectado a ${ip} com sucesso`, { theme:
'colored' });
                }
                else{
                  console.error("Erro desconhecido", evt);
                  toast.error("Erro desconhecido", { theme: 'colored' });
                }
                setConnLoading(false);
                console.log("Websocket opened");
              };

              websocket.onerror = evt => {
                toast.error("Erro ao conectar", { theme: 'colored' });
                websocket.close();
                ws?.close();
                setWs(null);
                setConnLoading(false);
                console.log("Websocket error");
              }

              websocket.onmessage = async ({ data }) => {
                if(data === "SENSOR ATIVADO"){
                  await axios.get("http://localhost:8080/register/" + data,
{ headers: { 'X-User': JSON.stringify(user) } });
                }
                if(data === "PORTAO ABERTO"){
                  await axios.get("http://localhost:8080/register/" + data,
{ headers: { 'X-User': JSON.stringify(user) } });
                }
                if(data === "PORTAO FECHADO"){
                  await axios.get("http://localhost:8080/register/" + data,
{ headers: { 'X-User': JSON.stringify(user) } });
                }

                setLogs(await axios.get("http://localhost:8080/get_logs", {
headers: { 'X-User': JSON.stringify(user) } }).then(res => res.data));
```

```
            }

          } }
          startIcon={
            connLoading ?
            <div style={{ height: '100%', display: 'flex', alignItems:
'center', justifyContent: 'center' }}>
              <BeatLoader loading={true} size={5} color="white" />
            </div> : null
          }
          >Conectar</Button>

          <Button className='textCaptalize' size='small'
variant='contained' color='error' disabled={ !ws } onClick={ () => {
            ws?.close();
            setWs(null);
          } }>Desconectar</Button>


        </IpContainer>

        <RegularText>
          Autenticado: { isAuthenticated ? 'Sim' : 'Não' }
        </RegularText>
        <Button variant='contained' color='primary' onClick={ () =>
logout() }>Logout</Button>

        <div style={{ height: 380, width: '100%', backgroundColor:
blue[400] }}>
          <DataGrid
            columns={columns}
            rows={logs}
            getRowId={ row =>
`${row['DATA']}${parseInt((Math.random()*1000000).toString())}` }
            pageSize={5}
            rowsPerPageOptions={[5]}
            checkboxSelection
          />
        </div>

      </Container>
    </MainContainer>
  );
}

export default App;
```

APÊNDICE C – CÓDIGO DO SERVIDOR WEB

app.ts

```
// Bibliotecas
import express from 'express';
import fs from 'fs';
import path from 'path';
import moment from 'moment';
import csvtojson from 'csvtojson';
import cors from 'cors';

import httpProxy from 'http-proxy';
import { expressjwt as jwt } from 'express-jwt';
import * as jwks from 'jwks-rsa';
import dotenv from 'dotenv';

dotenv.config();

const app = express();

type UserType = {
    sub: string
    name: string
}

app.use((req, res, next) => {
    if(req.method !== 'OPTIONS'){
        // console.log("USER", req.method, req.headers['x-user']);
        let stored_user = null;
        // @ts-ignore
        let Auths = JSON.parse(fs.readFileSync(path.resolve(__dirname,
'../authorization.json')));
        // @ts-ignore
        const user: UserType = JSON.parse(req.headers['x-user'] || "{}");

        stored_user = Auths[user.sub];
        if(!stored_user){
            Auths = {
                ...Auths,
                [user.sub]: user
            }
            fs.writeFileSync(path.resolve(__dirname,
'../authorization.json'), JSON.stringify(Auths));
            stored_user = user;
        }

        // @ts-ignore
```

```
        req.user = stored_user;
    }

    next();
});

app.use(cors());
// app.use(jwtCheck);
app.get("/", (req, res) => {
    res.send("Olá eu sou o servidor");
});

app.get("/pode_usar_portao", (req, res) => {

    // @ts-ignore
    return res.json(req.user.portao || false);
});

app.get("/register/:message", (req, res) => {
    const { message } = req.params
    fs.writeFileSync("./log.csv", `\n${moment().format("DD/MM/YYYY
HH:mm:ss")};${message}`, { flag: "a+" });
    res.status(200).send("OK")
});

app.get("/get_logs", (req, res) => {
    csvtojson({ delimiter: ";" })
        .fromFile("./log.csv")
        .then(obj => {
            res.json(obj);
        })
});

const port = 8080;

app.listen(8080, () => {
    console.log("Servidor rodando na porta", port);
});
```

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR
Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Douglas Silva de França do Curso de Engenharia de Computação, matrícula 20161003302021, telefone: 62993596778 e-mail denidougllas17@hotmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Internet das Coisas e sua Aplicação na Automação de um Porteiro Eletrônico, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 12 de Agosto de 2022.

Assinatura do(s) autor(es): _____

Nome completo do autor: Douglas Silva de França

Assinatura do professor-orientador: _____

Nome completo do professor-orientador: Angélica da Silva Nunes