UNIVERSIDADE DO ESTADO DO AMAZONAS-UEA
ESCOLA SUPERIOR DE TECNOLOGIA-EST
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

IGOR BRAINO FRAGA MARTINS

# PROTÓTIPO DE SISTEMA DE CONTROLE E MONITORAMENTO INFORMATIVO DE POSIÇÃO PARA UM ELEVADOR PREDIAL

Manaus
2023

**IGOR BRAINO FRAGA MARTINS**

# PROTÓTIPO DE SISTEMA DE CONTROLE E MONITORAMENTO INFORMATIVO DE POSIÇÃO PARA UM ELEVADOR PREDIAL

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada a banca avaliadora do Curso de Engenharia de Controle e Automação da Escola Superior de tecnologia da Universidade do Estado do Amazonas - UEA, como pré-requisito para a obtenção do título de bacharel de Engenheira de Controle e Automação.

Orientador: Moisés Pereira Bastos, Me.
Coorientador: Cleto Cavalcante de Souza Leal, Me.

MANAUS
2023

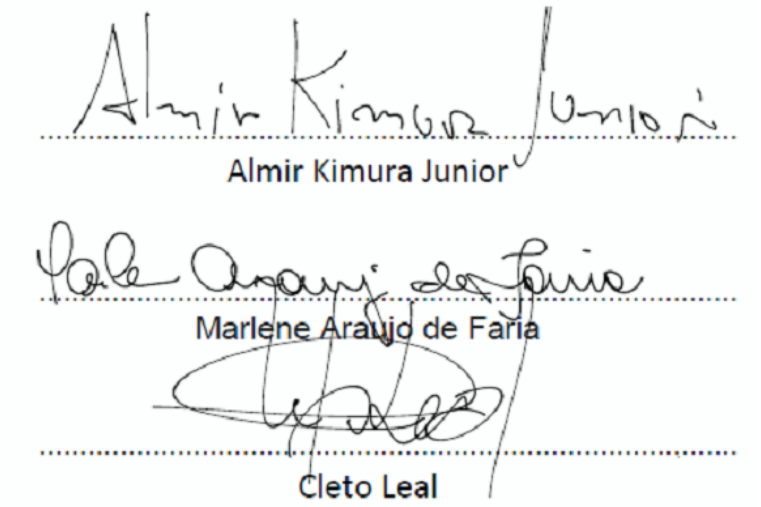
IGOR BRAINO FRAGA MARTINS

# PROTÓTIPO DE SISTEMA DE CONTROLE E MONITORAMENTO INFORMATIVO DE POSIÇÃO PARA UM ELEVADOR PREDIAL.

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada a banca avaliadora do Curso de Engenharia de Controle e Automação da Escola Superior de tecnologia da Universidade do Estado do Amazonas - UEA, como pré-requisito para a obtenção do título de bacharel de Engenheira de Controle e Automação.

Orientador: Moisés Pereira Bastos, Me.
Coorientador: Cleto Cavalcante da Silva Leal, Me.

Aprovado em 24 de março de 2023




Manaus
2023

## AGRADECIMENTOS

Agradeço primeiramente a Deus por tudo, por ter me dado a oportunidade de cursar engenharia na Universidade do Estado do Amazonas, onde entrei em 2012 e por ter me conduzido até esta etapa tão importante em minha vida, sempre me dando força e me abençoando com saúde muita paciência. A jornada não foi fácil principalmente por trabalhar no terceiro turno e ter que está na sala de aula durante o dia, mas com certeza foi muito produtiva.

A minha família por todo incentivo moral, espiritual. Ressaltando meus pais que em momentos que eu me encontrava desmotivado me incentivaram a continuar. Sempre estiveram caminhando comigo e o mais importante de tudo me cobrando. E a minha cônjuge Ariane Simão Santigo, que meu ajudou muito quanto aos afazeres para que eu tivesse tempo de me dedicar a faculdade e descansar para o trabalho.

Ao meu orientador, professor Me. Moisés Pereira Bastos, por sua paciência e orientação, pois me ajudou muito, principalmente na escrita da monografia, e mesmo sabendo das minhas dificuldades não me abandou até que eu concluísse o trabalho.

Ao meu coorientador professor Me. Cleto Cavalcante de Souza Leal foi quem iniciou desenvolvimento deste trabalho na disciplina de TCC1 como orientador, tanto no desenvolvimento do tema quanto ao levantamento das tecnologias a serem utilizadas.

A Escola Superior de Tecnologia EST/UEA e todo o seu corpo docente, por me proporcionarem todo o aprendizado necessário para a minha formação.

Agradeço também aos meus amigos, a secretária do curso Anna Carolina Quintino Nogueira e o coordenador Dr. Israel Mazaira Morales, que no final cursor deram uns puxões de orelha para que finalizasse a graduação não me deixando trancar o curso.

*“Nunca ande por trilhas, pois assim só irá até onde os outros já foram.”*

*Freeman Dyson*

## RESUMO

Nos dias atuais o ser humano está conectado à internet na maior parte do tempo, onde isto facilita muito a vida de alguns. Com o avanço da tecnologia agora não só as pessoas passaram a estar conectados na internet e sim os objetos. Desta forma a comunicação a distância entre humano e objetos, que podem ser colocados na rede, torna-se mais comum. Neste trabalho apresenta-se uma melhoria a uma edificação, onde a mesma contém dois elevadores que não mostram o posicionamento de sua cabine ao usuário, desta forma, fazendo com que o usuário os utilize de forma incorreta. Para se deslocarem os usuários realizam a chamada dos dois elevadores e faz o uso do que chegar primeiro. Logo, o segundo elevador realiza seu deslocamento no vazio uma vez que não terá ninguém para seu uso. Neste trabalho foi desenvolvido um protótipo que conta com três módulos baseado em programação em C. Foi utilizado os microcontroladores da família ESP (ESP12 e ESP32) que já vem com conexão Wifi na própria placa de prototipação, onde será utilizado internet das coisas através do protocolo MQTT para comunicação dos módulos. Em que um é responsável em verificar a localização da cabine no prédio, o segundo é de interação com o usuário e seleção da cabine mais viável e por último o módulo que interliga o protótipo ao sistema do elevador. Ao final foi criado o protótipo do sistema em três módulos, em que o conjunto se interliga através da internet utilizando o protocolo MQTT para troca de informações. O display atualiza em tempo real a localização da cabine e do módulo existente no controlador conforme as informações são atualizadas no servidor.



**Palavras - chave:** MQTT. Iot. ESP-12. ESP-32.

## ABSTRACT

Nowadays, human beings are connected to the internet most of the time, which makes life much easier for some. With the advancement of technology now, not only people have become connected to the internet, but also objects. In this way, long-distance communication between people and objects, which can be placed on the network, becomes more common. This work presents an improvement to a building, where it contains two elevators that do not show the position of their cabin to the user, thus causing the user to use them incorrectly. To move users, call both elevators and use the one that arrives first. Therefore, the second elevator performs its displacement in the void since there will be no one to use it. In this work, a prototype was developed that has three modules based on programming in C. We used microcontrollers from the ESP family (ESP12 and ESP32) that already come with a Wifi connection on the prototyping board itself, where the internet of things will be used through the protocol MQTT for module communication. In which one is responsible for verifying the location of the cabin in the building, the second is for interaction with the user and selection of the most viable cabin, and finally the module that connects the prototype to the elevator system. At the end, the prototype of the system was created in three modules, in which they are interconnected through the internet using the MQTT protocol to exchange information. The display updates in real time the location of the cabin and the existing module in the driver as the information is updated on the server.

# LISTA DE ILUSTRAÇÕES

LISTA DE ABREVEATURAS E SIGLAS

| | |
|---|---|
| *ULA* | *Unidade Lógica Aritmética* |
| *CPU* | *Central Processor Unit* |
| *PROM* | *Programmable Read Only Memory* |
| *RAM* | *Random Access Memory* |
| *WI-FI* | *Wireless Fidelity* |
| *GPIO* | *General Purpose Input/Output* |
| *I2C* | *Inter-Integrated Circui* |
| *UART* | *Universal Asynchrounous Receiver/Transmiter* |
| *PWM* | *Pulse-Width Modulation* |
| *IDE* | *Integrated Development Environment* |
| *USB* | *Universal Serial Bus* |
| *IOT* | *Internet Of Things* |
| *MQTT* | *Message Queue Telemetry Transport* |
| *QoS* | *Qualidade de Serviços* |
| *TCP/IP* | *Transmission Control Protocol/Internet Protocol* |
| *RXD* | *Recepção de dados* |
| *TXD* | *Transmissão de dados* |
| *VCC* | *Voltagem Corrente Contínua* |
| *RST* | *Reset* |
| *LED* | *light-emitting diode* |
| *GND* | *Graduated Neutral Density ou Terra* |

**SUMÁRIO**

# 1. INTRODUÇÃO

Desde a criação dos elevadores, os mesmos vêm sofrendo evoluções. A iluminação de lâmpadas fluorescentes e incandescentes sendo substituídos por iluminação a LED, motores de corrente contínua trocados por motores de indução utilizando inversores de frequência tornando-os mais confortáveis. (TROLLER, 2015) Em alguns prédios residenciais há falta de informação da localização da cabine do elevador, isto é, não está disponível ao usuário a localização exata do elevador, no caso da existência no mínimo dois elevadores.

Com essa falta de informação, ocorre acionamento ao mesmo tempo dos dois elevadores ou então, embora a cabine esteja no mesmo andar existe a possibilidade do usuário chamar o segundo elevador, entre outras possibilidades. Desta forma ocorre o maior gasto de energia devido aos acionamentos indevidos propositais, maior gasto de manutenção, menor vida útil dos componentes mecânicos, hidráulicos e elétricos, entre outros fatores.

Este trabalho tem como objetivo desenvolver um protótipo, para realizar o controle de informação da posição do elevador de acordo com o andar que o mesmo se encontra e disponibiliza ao usuário, sendo que, não sejam necessárias instalações invasivas ao patrimônio, ou seja, toda sua comunicação entre andares e elevador seja sem fio. Logo, não será preciso quebrar paredes para sua instalação.

## 1.1 TEMA

Protótipo de sistema de controle e monitoramento informativo de posição para um elevador predial.

## 1.2 FORMULAÇÃO DO PROBLEMA

A inexistência de informação ao usuário do andar em que o elevador se encontra, ocasionando eventuais deslocamentos desnecessários para o atendimento de chamadas indevidas.

## 1.3 HIPÓTESE

É possível a implementação do protótipo onde ficará disponível ao usuário a informação, da posição do elevador, em um display de um sistema embarcado utilizando a tecnologia NodeMCU32-ESP32, que se comunicará com outro módulo existente no elevador através de uma rede WIFI, onde o terá outra placa de prototipação ESP32 e sensores inseridos na cabine que serão acionados conforme sua movimentação.

## 1.4 OBJETIVOS

### 1.4.1 OBJETIVO GERAL

Projetar e construir um protótipo para detecção da posição de elevadores que mostre ao usuário sua localização exata por andar por meio de um display conectado a microcontroladores, permitindo-o realizar chamadas do lado externo a cabine tendo sua comunicação toda via *wireless.*

### 1.4.2 OBJETIVOS ESPECÍFICOS

a) Realizar levantamento bibliográfico relacionado aos temas do trabalho;

b) Realizar interligação dos módulos, de forma que eles se comuniquem através da internet utilizando o protocolo MQTT;

c) Identificar a localização do elevador em relação aos andares através da sua própria movimentação;

d) Construir um protótipo para realização dos testes de comunicação entre o modulo do elevador e os módulos dos andares.

## 1.5 JUSTIFICATIVA

A construção do protótipo de um sistema de controle e monitoramento informativo do posicionamento de um elevador engloba a aplicação de vários conceitos e estudos nas disciplinas de engenharia de controle e automação, tais como: Física; Linguagem de programação; Microcontroladores e microprocessadores; Automação e supervisão de processos; Controle e automação; Circuitos elétricos;

Eletrônica digital, Modelagem e simulação de sistemas discretos e redes de computadores.

O projeto de pesquisa justifica-se pelo fato de evitar deslocamentos desnecessários, com isso, implica diretamente redução de gastos de energia elétrica, aumento da vida útil dos componentes mecânicos, elétricos e hidráulicos. Onde a implementação do sistema é simples e com custo reduzido.

# 2 REFERENCIAL TEÓRICO

Nesta parte do trabalho apresentam-se os recursos teóricos necessários para desenvolvimento e criação do projeto de pesquisa e protótipo.

## 2.1 ELEVADORES

Caixa metálica suspensa por cabos de aço onde realiza movimentos verticais para transporte de pessoas, cargas entre outros. De acordo com (BALDO, 2017) é uma cabine com movimentos verticais destinados a pessoas e cargas de modo estacionário.

Segundo (PAULA, 2014) é composto por um sistema complexo para transporte de pessoas e cargas de forma segura e confortável com vários elementos para garantir a segurança do conjunto.

### 2.1.1 ORIGEM DO ELEVADOR

A descrição pela primeira vez de uma cabine vertical com uma plataforma suspensa para transporte de pessoas e cargas foi realizada pelo arquiteto romano Vitruvias no século I a.C. Onde o mesmo é conectado a uma manivela externa interligado por roldanas em que os movimentos de subida e descida são realizados através de força humana (BALDO, 2017).

Segundo (BALDO, 2017) o primeiro elevador criado para transporte de pessoas foi em 1743, no palácio de Versalhes. Onde o rei Luís XV tinha seu aposento interligado com sua amante, no qual foi utilizado um motor a vapor para enrolar e desenrolar o cabo ao redor do cilindro.

Com o passar do tempo o homem busco melhorias, em 1823 foi construído o primeiro elevador hidráulico para transporte de 20 pessoas. Em 1930 foi desenvolvido o primeiro elevador com acionamento mecânico e sistema hidráulico acionado por uma máquina a vapor (BALDO, 2017). Mesmo com todos os avanços tecnológicos o transporte de pessoas ainda era bastante perigoso com riscos de acidentes. Até Quando foi apresentado uma solução por Elisha Graves Otis, com um dispositivo de frear o elevador caso os cabos se rompam (PAULA, 2014). Esse dispositivo continua

sendo utilizado nos elevadores, porém seu acionamento é através dos limitadores de velocidade quando a cabine ultrapassa a velocidade pré-determinada (BALDO, 2017).

Segundo site Atualize elevadores um elevado e composto por seis conjuntos básicos de modo geral:

**Casa de Máquinas:** Compartimento onde são colocados os equipamentos e componentes responsáveis pela movimentação e funcionamento do Elevador, como a máquina de tração, quadro de comando, painel seletor, limitador de velocidade entre outros. A mesma é alocada na parte superior. Devido a avanços tecnológicos, existem modelos que dispensam a presença de Casa de Máquinas, o motor fica apoiado nas guias (trilhos do elevador), e o quadro de comando é embutido ao lado da porta, de acordo com o fabricante, no primeiro ou do último pavimento.

**Cabine:** local onde são transportados as pessoas e/ou cargas.

**Caixa Corrida ou Passadiço:** Local onde fica os guias (trilhos) que corre a cabine e o contrapeso, as calhas com a fiação elétrica fixa, fiação elétrica móvel, cabos (comando, da tração, cabo de aço do limitador de velocidade) e limites de segurança nos extremos.

**Contrapeso:** responsável pelo transporte e balanceamento de carga utilizando menos energia na operação, além de permitir o equilíbrio das cargas distribuídas por todo o equipamento.

**Pavimento de Acesso ou Patamar:** São os locais de parada da cabine para entrada e ou saída de passageiros ou carga.

**Fundo do poço:** É a parte inferior da caixa corrida ondem ficam instalados os dispositivos de segurança como polia tensora, mola ou pistão, para-choque, limites de segurança, chave PAP, botão de emergência, iluminação e tomada.

O Quadro de Comando é um dos itens mais importantes deste trabalho, ele é responsável por toda a lógica de controle, gerenciando o sistema e processando as informações de todos os comandos do equipamento. É onde será incluída a melhoria de chamada dos elevadores. Local onde é processado todos os dados de chamadas e movimento da cabine. Esses quadros são cubículos constituídos de bobinas, relês, transformadores, chaves de força, inversor de frequência, e placas de circuitos

eletrônicos. (site atualize Elevadores). A figura 1 mostra as principais partes do elevador.

Figura 1- Partes básicas de um elevador


Fonte: site Atualize Elevadores

## 2.2 WIRELESS

As necessidades da sociedade tornam-se cada vez mais complexas com o passar do tempo, com isso, o mercado da comunicação tem sofrido uma enorme expansão para satisfazê-la. Desta forma a comunicação sem fio desenvolveu múltiplas alternativas e cada uma delas incorpora aplicações distintas. (LIMA; NUNES, 2015).

"*A palavra "wireless" é um termo em inglês e significa "sem fio" (wire - fio, less - sem ou menos) possui alguns sinônimos tais como: comunicação sem fio, computação móvel e rede de computadores sem fio.*" (MENDES apud NENOKI EDUARDO, 2013, p.14).

De acordo com (LIMA; NUNES, 2015) comunicação sem fio transporta seus dados através do ar no espaço combinados a conectividade dos equipamentos para recepção tratamento e envio de acordo com a flexibilidade de usuários, na qual se destacam as tecnologias como infravermelho, radio frequência (RFID) ou micro-ondas. Com avanço da tecnologia as redes sem fio passaram a atender aplicações com transmissão em baixas taxas de dados, por exemplo controle remotos de

equipamentos eletrônicos, deixando de ser voltada apenas para aplicações corporativas.

Segundo (TEXEIRA, 2014) redes sem fio apresentam as seguintes vantagens como: Mobilidade, flexibilidade, facilidade de instalação e economia. Trazendo também suas limitações tecnológicas como velocidade e alcance quando são comparados a sistemas de redes estruturadas.

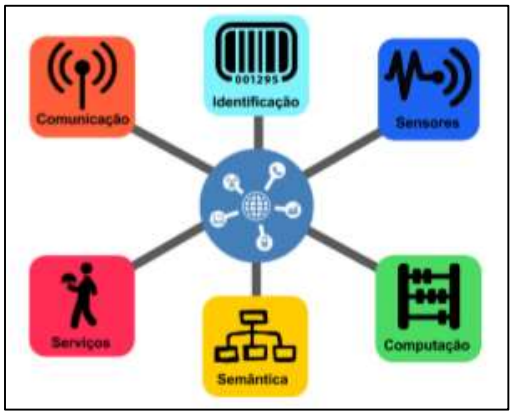
## 2.3 INTERNET DAS COISAS

Internet das coisas termo que deriva do inglês internt of thing (IoT) nada mais do que a comunicação entre diversos dispositivos através da transferência de dados pela internet.

De acordo com (WANZELE; FULBER; MERLIM, 2016) internet das coisas tem como finalidade interligar por meio da internet diversos equipamentos usuais do cotidiano ou receber informações através da transferência de dados colhidos por redes de sensores, onde os mesmos podem ser processados e retornar benefícios aos usuários. Essa definição pode ser observada em diversos projetos de automação residencial, onde objetos podem ser controlado remotamente assim como enviar informações do local em questão como temperatura e umidade por exemplo.

Na opinião de (SANTOS, 2016) IoT pode ser visualizado como um arranjo de diversas tecnologias conectadas à internet, onde elas são constituídas por blocos básicos de construção mostrado na figura 2:

Figura 2- Bloco básico da IoT




Fonte: (SANTOS, 2016)

- Identificação - Para direcionamento dos dados sem conflito na rede, é importante que a identificação seja única exemplo o endereçamento de IP.
- Sensores e atuadores - No qual o primeiro e responsável e para sentir e capitar estímulos do meio e enviá-los através de dados que são processados para análise dessa informação ou armazenamento. Já os atuadores são os que realizam a manipulação no meio em resposta ao que foi captado pelo sensor.
- Comunicação - São técnicas utilizadas para interligar objetos inteligentes a rede exemplo: WiFi.
- Computação - unidade de processamento desses objetos.
- Serviço – refere-se à função empregada aos sensores e atuadores a determinado sistema. Exemplo: localização geográfica, leitura de temperatura.
- Semântica - Está ligado de como retirar conhecimento dos objetos ligados na IoT.

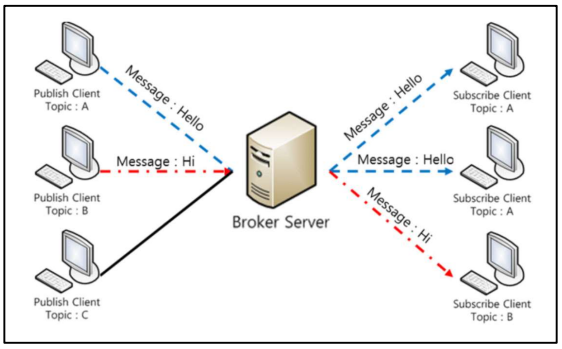
### 2.3.1 PROTOCOLO DE COMUNICAÇÃO

Para comunicação de vários dispositivos em uma rede necessita-se de uma padronização, cujo mais utilizado por diversas empresas e o protocolo MQTT.

Protocolo MQTT (*Message Queue Telemetry Transport*) apresenta características de envio de mensagem que exigem pouca largura de banda, licença livre, com níveis de qualidade de segurança (QoS) e é leve. Assim simplificando o uso dos recursos de dispositivos que muito das vezes tem um poder computacional restrito (SANTOS, 2016).

Esse protocolo trabalha com o padrão de mensagem *Publish/Subscriber* (publicador/assinante) que são os clientes, onde eles são intermediados por um *broker* que responsável pelo direcionamento dos pacotes de dados enviado pelo *publish* até os *subscriber* correto como mostra a figura 3. Para que ocorra essa conexão, o cliente envia uma mensagem ao *broker* que responde com um CONNACK (JUNIOR, 2017).

Figura 3- Diagrama de fluxo de dados via protocolo MQTT




Fonte: (Lee; Kim, 2013).

De acordo com (JUNIOR, 2017) a exemplificaremos mensagens de conexões figura 4 de cliente ao *broker* e resposta do mesmo, o CONNACK, conceituando cada parâmetro para melhor entendimento.

Figura 4- Mensagem de conexão ao broker

MQTT-Packet:

**CONNECT**

| contains: | Example |
|---|---|
| clientId | "client-1" |
| cleanSession | true |
| username (optional) | "hans" |
| password (optional) | "letmein" |
| lastWillTopic (optional) | "/hans/will" |
| lastWillQos (optional) | 2 |
| lastWillMessage (optional) | "unexpected exit" |
| lastWillRetain (optional) | false |
| keepAlive | 60 |

Fonte: (ANSARI; REHMAN; MUGHAL, 2018)

ClientId - Identificador do cliente, sendo necessário ser único para cada *broker*.

CleanSession - Indica a persistência da conexão do cliente ao *broker*, quando sim a *flag* é falsa, com isso, caso ocorra uma perda de conexão entre cliente e o *broker*, o segundo salvará as mensagens de Qos 1 e 2 perdidas e o tópicos do cliente. Caso verdadeira, não salvará nada sendo necessário uma nova assinatura do cliente ao se reconectar.

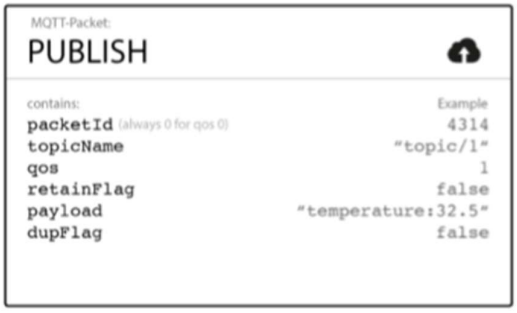
Username e password - autenticação do Cliente podendo ser utilizado certificado de segurança dependendo do *broker* utilizado.

LastWill - quando algum cliente se desconecta inesperadamente esses campos do LastWill informa ao *broker* que dever comunicar aos Clientes ainda conectados da desconexão.

KeepAlive - tempo que a conexão permanece aberta após inatividade do Cliente.

No CONNACK, contém duas informações sessionPresente e returncode. Sendo que a primeira indica uma persistência anterior na conexão e a outra é uma a *flag* de indica o êxito na tentativa de conexão. Depois da comunicação estabelecida entre o *Broker* e cliente, o último pode enviar dados assumindo a condição de *publish* ou recebendo dados sendo um *subscriber* (JUNIOR, 2017). Quando *publish* e necessário o envio da mensagem na figura 5.

Figura 5- *Publish message*




Fonte: blog HiveMQTT

PacketId- Identificador único entre *client/broker*, cujo tem função de identificação de mensagem.

TopicName- nome dado ao tópico de publicação para o *broker* organizá-lo para que o subscriber escolha o tópico desejado.

Qos- É a qualidade de serviço oferecido que pode variar de 0 (zero) a 2(dois), onde 0 não há garantia de entrega, 1 garante a entrega do pacote pelo menos uma vez e por fim 2 que garante que o pacote seja entregue uma vez ao *broker*.

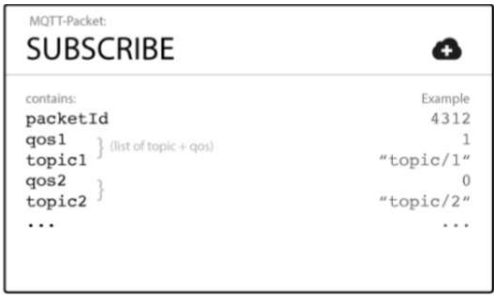
RetainFlag- informa ao *broker* se o mesmo deve salvar a última mensagem publicada pelo cliente para que quando haja uma nova conexão de um cliente *subscriber* o mesmo receba imediatamente.

Payload - É a informação enviada, ou seja, o Dado a ser trabalhado que deve ter no máximo 250MB.

DupFlag - indica se a mensagem foi reenviada, caso tenha ocorrido algum problema no reconhecimento da mensagem pelo destinatário.

Após a publicação da mensagem no *broker*, os *subscriber* pode ter acesso aos tópicos depois de realizar sua inscrição com uma *Subscriber message*. Representado na figura 6 a seguir.

Figura 6- *Subscriber message*




Fonte: blog HiveMQTT

O packetId, Qos e tópico referente repete-se com a mesma função da *message publish*, e o *broker* responde com uma *suback message*, que contendo o *pocketId* e *returnCode* indique ao *subscriber* quais tópicos foram inscritos e quais ocorreram falha por não ter permissão para receber mensagens desse tópico ou o mesmo foi criado incorretamente (JUNIOR,2017).

Como servidor está sendo utilizado o Eclipse Iot, um servidor gratuito, ilustrado na figura 7, onde ele é disponibilizado com forma de incentivar a criação de projetos desde que não seja utilizado para fins comerciais, para utilização do servidor MQTT e liberado ao host mqtt.eclipseprojets.io e a porta 1883 caso necessite acesso de porta criptografada utiliza-se a 8883, não sendo necessário nenhuma configuração na nuvem, pois os acessos são realizados por tópicos. Este servidor utilizar o corretor de mensagem Mosquito MQTT de código aberto (eclipse iot, 2001).

Figura 7- Servidor eclipse IoT


Fonte: Eclipse iot.

## 2.4 HISTÓRIA DOS MICROCONTROLADORES

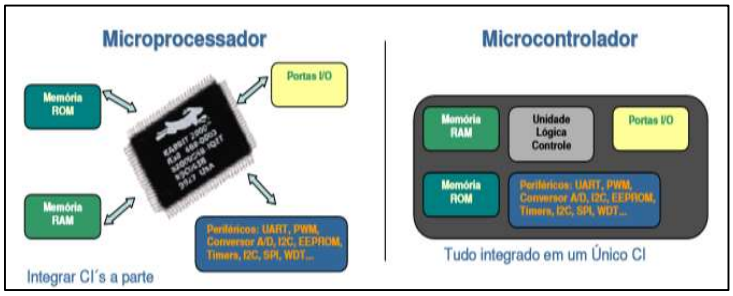
De acordo com (ARAUJO; CAVALCANTE; SILVA, 2019), a história dos microcontroladores inicia-se em 1971, com a criação do primeiro microcontrolador, o TMS1000 contendo microprocessador de 4 bits integrando memória de leitura e escrita, um clock e portas de entrada e saída, criado por dois engenheiros da Texas instruments Gary Boom e Michael Cochrom sendo usado em calculadores até 1974, quando foi oferecido a indústria eletrônica sendo aprimorado com o passar dos anos e em paralelo a Intel lança o microcontrolador 8048 embarcado em teclados e vídeo games em seguida no ano de 1980 lançando o microcontrolador 8051 com 4 kilobytes de memória programável, 128 Bytes de memória de dados e um processador de 8 bits.

Os microcontroladores eram inviáveis para fins domésticos e pequenos projetos, por não ser reprogramável, mas em 1990 isso muda com a criação das memórias eletricamente programáveis, EEPROM (*Electrically-Erasable Programmable Read-Only Memory*), podendo assim reutilizar os microcontroladores em outros projetos, com o avanço da tecnologia utiliza-se a memória FLASH baseada na anterior citada, porém com uma capacidade e facilidade na sua programação podendo ser apagados e escritos em alta velocidade (ARAÚJO; CAVALCANTE; SILVA, 2019).

### 2.4.1 MICROPROCESSADORES E MICROCONTROLADORES

No passado os circuitos eletrônicos eram muito grandes, complexo e caro tornando inviável para alguns projetos, pois eram criados a partir de componentes separados como resistores, diodos, capacitores, indutores, transistores dentre outros. Porém, com o passar do tempo foram surgindo os circuitos integrados que realizavam a mesma função dos componentes antes mencionados com uma diferença, todos encapsulados reduzindo significativamente o tamanho dos equipamentos eletrônicos com uma grande variedade de funções que vai das mais simples como AND, OR ou NOT até mais complexas como a capacidade de realizar inúmeras tarefas em um único chip como os microprocessadores e microcontroladores conforme a figura 8. (ARAÚJO; CAVALCANTE E SILVA, 2019).

Figura 8- Microprocessadores e Microcontroladores.




Fonte: (CHASE, 2007)

Microprocessadores de acordo com (CHASE, 2007) são componentes com alto poder de processamento para cálculos matemáticos e endereçamento de memória externa, ou seja, todos os periféricos são externos que para realizar acesso utilizam barramentos de dados, controle e endereço. Por tanto para que o mesmo funcione e necessário que esteja ligado aos seus periféricos. Completando a ideia com (ARAUJO; CAVALCANTE; SILVA, 2019), esse componente precisa ser gerenciado por sistemas que ficam armazenados em uma memória externa e outros circuitos para que haja interação com o usuário.

Microcontroladores contém num único chip: interface de entrada/saída analógica e digital, memória RAM e FLASH, interface de comunicação serial, conversores analógicos/digitais e temporizadores/contadores. Esses pequenos sistemas computacionais têm seus programas próprios, famosos firmwares, podendo

adquirir mais funções se somados a outros periféricos externos (CHASE, 2007). Completando com (BARROS; CAVALCANTE, 2003) esse componente não tem derivação dos microprocessadores e apresenta um repertório de instruções adaptado para fins de criação que são os sistemas embarcados, tendo assim, menor poder de processamento.

## 2.5 SISTEMAS EMBARCADOS

De acordo com (CHASE; ALMEIDA, 2007) esse termo surgiu no fim do século 1960. Na época existia um pequeno programa escrito em assembler, ele era um controle funcional de telefones, que com o passar do tempo foi utilizado em outro dispositivo adaptando as entradas e saídas de acordo da necessidade desse outro dispositivo sem alterar linhas do código original. Após o passar do tempo com os microprocessadores, começaram a ser desenvolvidos softwares em linguagem de máquina para vários processadores.

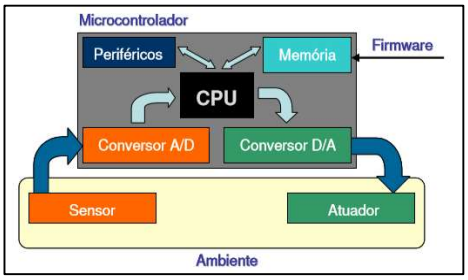
Também chamado de sistema embutido é um circuito integrado a um equipamento ou sistema com capacidade computacional encapsulado capaz de realizar apenas uma determinada tarefa, diferente dos computadores com sistemas operacionais permitindo que sejam utilizados aplicativos e programas com diferentes funcionalidades. Sendo que o usuário final pode ou não interagir com o equipamento dependendo da existência da interface (CUNHA, 2007).

Atualmente existem milhares de sistemas embarcados, a grosso modo pode-se dizer que um sistema embarcado é qualquer sistema novo inserido em algo existente para realizar uma determinada função. Um exemplo é o alarme dos carros antes inexistente. Então foi inserido um sistema que reconhece comandos para travar o veículo e confirme que o mesmo está trancado e quando ele é aberto de forma forçada, ou seja, antes de desativar o sistemas de travas, o sistema embarcado em questão é responsável apenas em emitir um sinal sonoro.

Segundo a (CHASE; ALMEIDA, 2007) necessita-se de um “cérebro”, uma unidade de processamento para gerenciamento, onde serão tratados os estímulos externos e de acordo com o que foi lido realizar as tarefas definidas em programa e enviá-las aos atuadores os resultados esperados. A figura 9 idealiza um diagrama básico de um sistema embarcado. E para realização dessa função um

microcontrolador ou microprocessador é ideal, no qual para desenvolvimento deste trabalho será utilizado o primeiro citado.

Figura 9- Diagrama básico de sistema embarcado



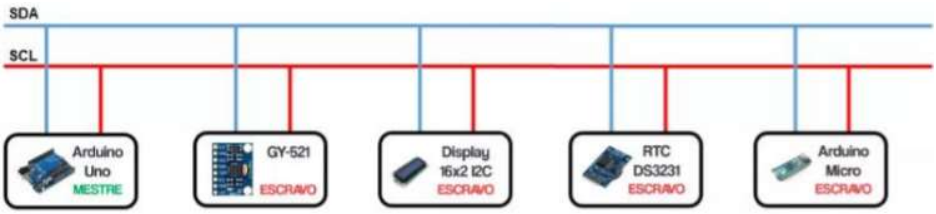

Fonte: (CHASE; ALMEIDA, 2007)

## 2.6 PROTOCOLO I2C

De acordo com (ANTONIOLLI, 2020) é um protocolo desenvolvido para realizar a comunicação de dispositivos no mesmo circuito impresso na década de 80, onde I2C significa “Inter-Integrated Circuits”, ou Circuito Inter-integrado. Nesse protocolo não há necessidade de chip de seleção ou lógica de arbitragem.

O protocolo I2C, consiste em um sistema que interliga vários dispositivos eletrônicos através de dois barramentos, onde a comunicação é baseada hierarquicamente em mestre/escravo, no qual, um dispositivo mestre coordena o restante chamados de escravos. Pode-se manter até 127 dispositivos em uma mesma rede. O barramento divide-se em DAS (Serial Data) que é o responsável pela troca de dados entre os dispositivos e o SCL (Serial Clock) mantém a confiabilidade do sistema através da sincronia dos dispositivos, podemos observar a estrutura do barramento na figura 10 (FIRMINO; MATEUS, 2020).

Figura 10- Exemplo do barramento I2C

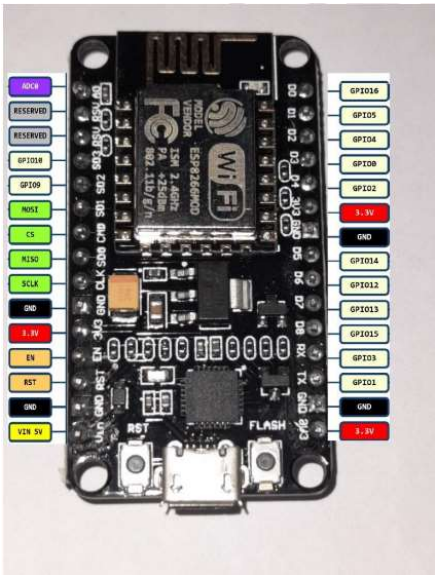



Fonte: (FIRMINO; MATEUS (2020) Apud Vida de Silício, 2017);

## 2.7 NodeMCU8266 ESP-12

NodeMCU Devkit 1.0 é constituído de um ESP-12E em uma placa que facilita o seu uso, o qual ainda possui um ESP8266EX, regulador de tensão e interface USB. De acordo com (PARIHAR, 2019), o módulo Wifi ESP8266 NodeMCU, E uma placa que agrupou ao chip ESP8266, o regulador de tensão de 3,3V e comunicação usb-seria, com programação na linguagem LUA ou a IDE do arduíno. Contendo ainda uma antena embutida, para comunicação ao computador um conector micro-usb, 11 pinos de I/O e conversor analógico-digital. Este modulo vem com ESP-12F conforme mostra a figura 11.

Figura 11- NodeMCU ESP-12 com pinagem




Fonte: (PARIHAR, 2019)

Especificações:

- ESP8266 ESP-12F
- Wireless padrão 802.11 b/g/n
- Antena embutida
- Conector micro-usb
- Modos de operação: STA/AP/STA+AP
- Suporta 5 conexões TCP/IP
- Portas GPIO: 11 mostrados na tabela 1
- GPIO com funções de PWM, I2C, SPI, etc
- Tensão de operação: 4,5 ~ 9V
- Taxa de transferência: 110-460800bps
- Suporta Upgrade remoto de firmware
- Conversor analógico digital (ADC)
- Distância entre pinos: 2,54mm
- Dimensões: 49 x 25,5 x 7 mm

Tabela 1: Tabela de pinos do ESP-12

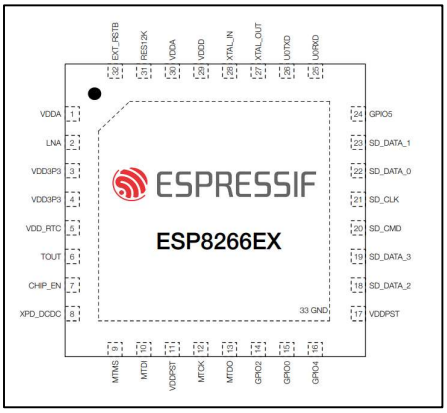
| Nome | descrição |
|---|---|
| VCC | 3.3V |
| GPIO 13 | Pode ser usado como SPI MOSI |
| GPIO 12 | Pode ser usado como SPI MOSI |
| GPIO 14 | Pode ser usado como SPI MOSI |
| GPIO 16 | |
| CH_PD | Enable do ESP8266<br>Nível lógico 1 - Ativado<br>Nível lógico 0 - Desativado |
| ADC | Entrada analógico para digital |
| Reset | 1 - Normal 0 - Reset |
| TXD | Tramissão serial |
| RXD | Recepção serial |
| GPIO 4 | GPIO normal |
| GPIO 5 | GPIO normal |
| GPIO 0 | Deve estar em nível lógico 1 para inicialização do programa(boot mode) e 0 para carregar o programa (flash mode) |
| GPIO 2 | deve estar em nível lógico 1 para inicialização do programa (boot mode) |
| GPIO 15 | Deve estar em nível lógico 0 para carregar o programa (flash mode) e para inicialização do programa (boot mode) |
| GND | Terra |

Fonte:(OLIVEIRA, 2017)

### 2.7.1 ESP8266EX

Criado pela Espressif, este microchip conforme mostra a figura 12 possui WiFi integrado e baixo consumo de energia com um Processador RISC Tensilica L106 32 bit com clock máximo de 160 MHz (PARIHAR, 2019).

Figura 12- ESP8266EX




Fonte: (Datasheet ESP8266, 2023)

É um dispositivo específicos através GPIOs, e possibilita uma solução autônoma e completa a partir de rede Wi-Fi, podendo funcionar com as duas alternativas, tanto para hospedagem de aplicativos quanto para descarregar as funções da rede Wi-Fi de outro processador de aplicativo. O componente é frequentemente combinado com sensores externos e outras aplicações.

A Plataforma de Conectividade Inteligente da Espressif Systems (ESCP) demonstra recursos sofisticados no nível do sistema de mudança rápida de contexto de suspensão/ativação visando eficiência energética VoIP, polarização de rádio adaptativa para operação de baixa potência, avanço processamento de sinal e recursos de cancelamento de estímulo e de coexistência de rádio para celular comum, Bluetooth, DDR, LVDS, Atenuação de interferência do LCD (SAIKRISHNA; VIJAYKIRAN, 2017) traduzido por google tradutor.

Tabela 2: Especificações Gerais do ESP8266-12E NodeMCU

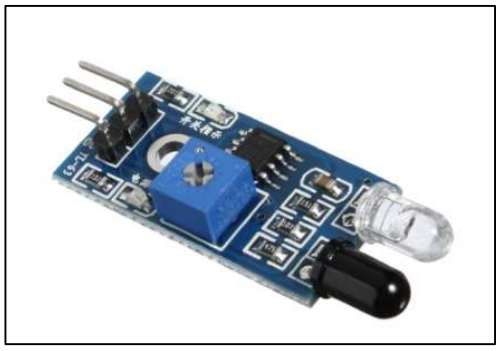
| | |
|---|---|
| Voltagem | 3.3V |
| Consumo de Corrente | 10 $\mu$A |
| Memória Flash | 16MB max (512k normal) |
| Processador | Tensilica L106 32 bit |
| Velocidade do processador | 80-160MHz |
| RAM | 32K + 80K |
| GPIOs | 17(multiplexada com outras funções) |
| Analogico para digital | 1 entrada com 1024 de resolução |
| Suporte 802.11 | b/g/n/d/e/i/k/r |
| Maxima corrente de conexão TCP | 5 |

Fonte: (OLIVEIRA, 2017)

## 2.8 SENSOR DE OBSTÁCULO INFRAVERMELHO IR

Sensor de detecção infravermelho composto por dois leds, sendo um emissor outro receptor, no qual ele emite uma luz em uma frequência não visível a olho nu, que ser cortado por uma barreira refletirá a luz e a leitura e realizada pelo receptor e identifica que existe algo na frente figura 13. O sensor também disponibiliza um potenciômetro para ajuste da distância, onde a mesma varia de 2mm a 80mm, que o anteparo deve ser identificado, (ROBOCORE, 2018).

Figura 13: Sensor infravermelho



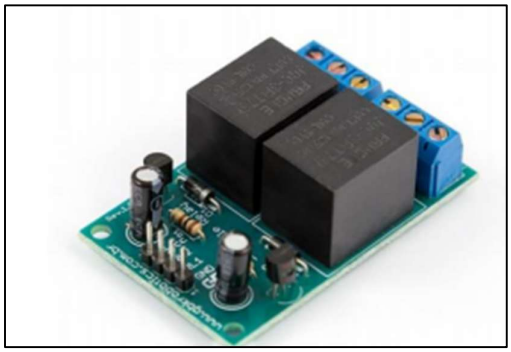
Fonte: (ROBOCORE, 2018)

## 2.9 MÓDULO RELÉ

Podemos defini-lo como um interruptor mecânico atuado por uma fonte elétrica, daí o nome eletromecânico. Ele possui os basicamente uma bobina, um

atuador mecânico e os contatos isolados normalmente aberto (NA) e normalmente fechado (NF), o módulo relé pode ser encontrado em diversos tipos e modelos, variando a quantidade de canais (2-16 relés por placa), tensão e corrente de operação. Na figura 14 é visto um modulo relé de 2 canais.

Figura 13- relé modulo 2 canais



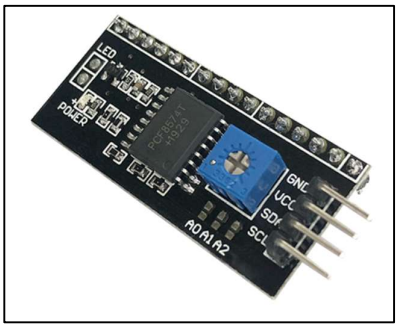
Fonte: GBK, robotics.

Seu princípio de funcionamento se dá por efeito eletromagnético, onde é inserido uma tensão nos polos da bobina que gera um campo magnético que faz a função de atração magnética dos contatos, com a movimentação desses contatos ocorre junção dos contatos NO e a separação dos contatos NF. Quando é interrompido a alimentação da bobina uma mola realiza o retorno dos contatos para posição inicial (BRAGA, 2017).

## 2.10 MÓDULO I2C DISPLAY

De acordo com (ABID; RUMON, 2020) a placa I2C para interface display como mostra a figura 15 baseia-se no protocolo I2C compatível com displays LCD 16x2 e LCD 20x4, onde ele possui os 2 pinos de alimentação de 5V e mais 2 pinos de comunicação para controle do display de LCD. O mesmo funciona com uma interface entre os microcontroladores e o Display utilizado apenas 2 pinos.

Figura 14- Placa I2C de interface para Display LCD

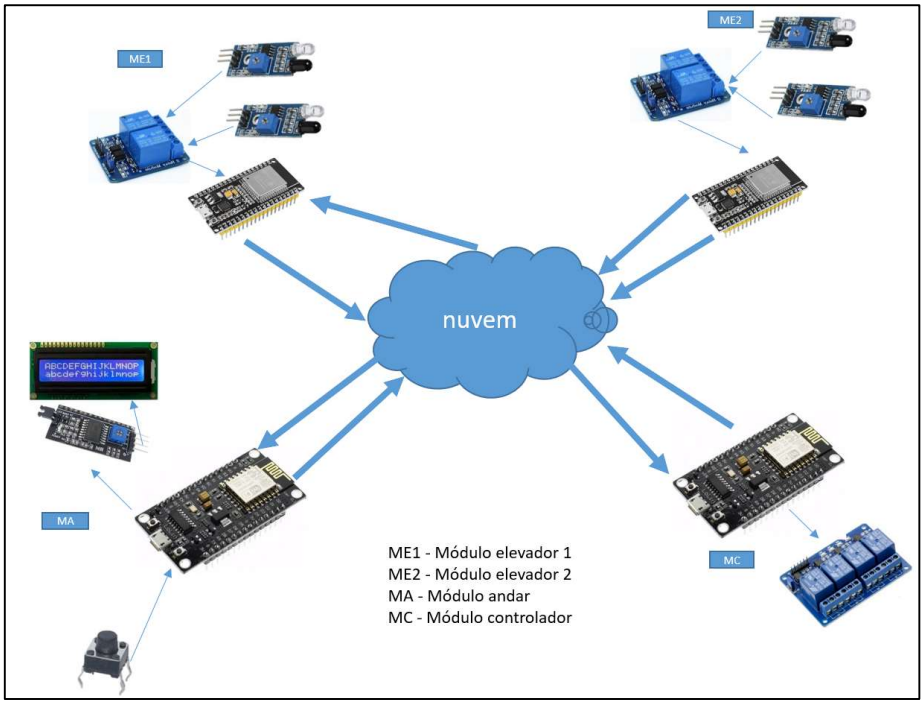



Fonte: (ROBOCORE, 2018)

# 3 METODOLOGIA

Para este trabalho foi criado um protótipo que visa reconhecer a posição em que um elevador residencial se encontra e apresente ao usuário do lado externo a cabine, sistema atualmente inexistente nos dois elevadores disponíveis em um determinado prédio. O mesmo oferece interação entre usuário e o sistema comum de qualquer elevador, com a diferença que a escolha da cabine mais viável para o andar solicitado é realizada de forma autônoma pelo próprio projeto. Para tal, utilizou-se a plataforma Arduino para programação dos microcontroladores da família ESP com implementação de uma arquitetura não invasiva, isto é, sem modificar a estrutura da edificação. A comunicação entre os módulos ocorre através da internet, via *wireless*, em que é utilizado o protocolo MQTT.

Figura 15- Diagrama ilustrativo da disposição dos módulos.



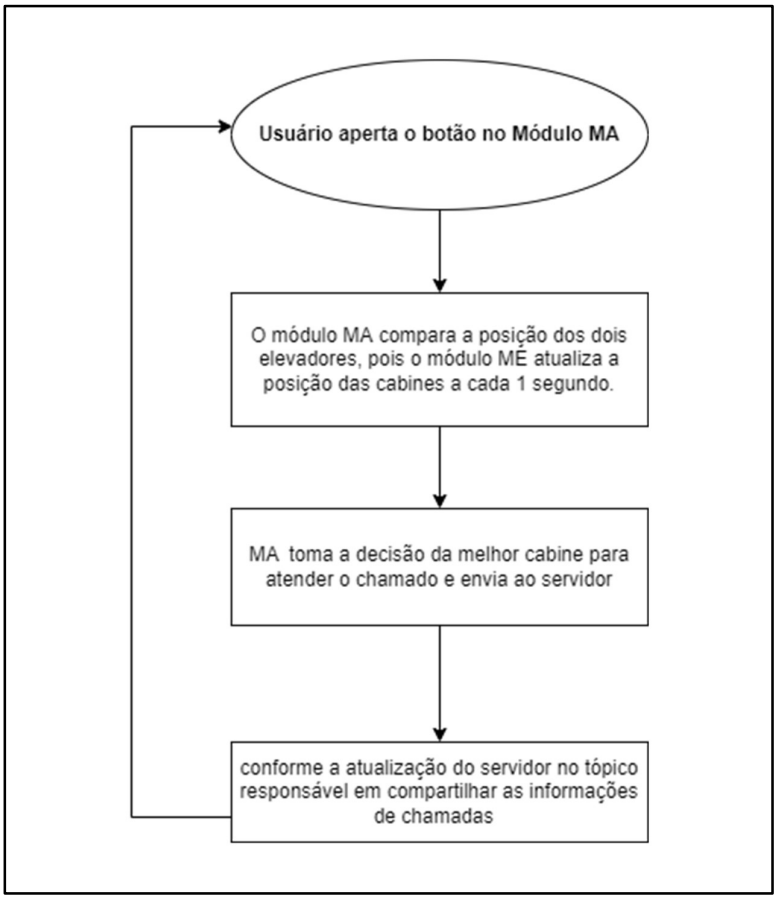

Fonte: Autor.

O sistema conta com basicamente três módulos: ME (módulo inserido no elevador), MA (módulo inserido no andar) e MC (módulo inserido no controlador do elevador) conforme disposição dos módulos apresentado na figura 15 O ME é responsável em disponibilizar no servidor MQTT o posicionamento do elevador por

meio do microcontrolador ESP-32. O MA recolhe informação do posicionamento dos elevadores no servidor e disponibiliza ao usuário por meio do display. Em caso de solicitação de chamada do elevador pelo usuário, o MA envia à nuvem a informação de qual cabine é mais viável ao atendimento da chamada. Por fim, o MC recolhe informações disponíveis no *broker* referente as solicitações de chamadas direcionadas ao elevador correspondente, assim acionando a entrada interligada ao botão referente ao andar solicitado.

A partir do fluxograma mostrado na figura 16, pode-se observar o passo a passo das ações realizadas pelos módulos por meio de dois tópicos ligados a cada elevador. Onde um compartilha informações do andar em que a cabine está localizada e no outro, dados dos andares que realizaram a solicitação de chamada.

Figura 16- Diagrama ilustrativo do fluxo da arquitetura




Fonte: Autor.

## 3.1 PREPARAÇÃO DA IDE E NUVEM

### 3.1.1 Configuração da IDE

Logo de início foram realizados testes nos microcontroladores, porém, para trabalhar com o NodeMCU8266 e com NodeMCU32s-ESP32 no IDE do Arduino é necessário realizar a instalação das placas. No qual, para a placa NodeMCU8266 realizou-se os seguintes passos no IDE segundo THOMSEN (2016):

- Ir em Arquivos/Preferências, no campo "URLs Adicionais de gerenciamento de Placas" onde foi inserido o link: *http://arduino.esp8266.com/stable/package_esp8266com_index.json*
- *Ir em Ferramentas/Placas:/Gerenciador de placas, procurar pelo pacote do ESP8266 e instalar.*
- *Em ferramentas/Placas, escolher a placa NodeMCU 1.0 (ESP-12E Module).*
- *Após esse processo realizar a instalação do drive CH340G no PC (Computador Pessoal) para poder haver conexão entre o PC e o microcontrolador.*
- *Por fim, foram instaladas as bibliotecas necessárias durante a programação: PubSubClient (para utilizar o protocolo MQTT) e ESP8266WiFi (para realizar conexão do microcontrolador com a rede Wifi).*

Com relação ao NodeMCU32 foi realizado um processo diferente, conforme (ALMEIDA, 2018) listado abaixo:

- Baixar e instalar o Python (versão 2.7 ou superior) e Git;
- Executar a interface gráfica do Git
- Selecionar a opção "*Clone Existing Repository*" no campo "source location" e inserir o link: *https://github.com/espressif/arduino-esp32.git*
- No campo "source Directory" selecionar ao local de instalação do arduino.
- Após instalação, na pasta *tools* executar o arquivo get.exe.
- *Após esse processo realizar a instalação do drive cp2102 para habilitar comunicação entre o PC e microcontrolador.*
- Foi instalado a biblioteca de conexão ao wifi do ESP32, WiFi.h, e liquidCrystal_I2C.h para estabelecer uma comunicação serial entre micro e o display LCD via protocolo I2C.

### 3.1.2 Nuvem

A nuvem é o meio utilizado para comunicação dos módulos através do servidor gratuito Eclipse Iot, com o auxílio da internet e utilizando o protocolo de comunicação MQTT. Neste broker não é necessário realizar nenhuma configuração. Os dados são

enviados à nuvem pelos clientes publicadores, as informações ficam disponíveis nos tópicos em que foram publicados e os clientes assinantes acessam os tópicos em que desejam informações, assim baixando os dados para processamento.

No início da programação de cada módulo são realizadas as definições de rede MQTT, como criação do ID para cada cliente, que deve ser único para identificação do seu módulo, caso tenha um igual durante a comunicação o último a se comunicar derruba o anterior. Outra definição realizada é o nome do tópico de publicação, o mesmo que será assinado pelos clientes através desse nome e por fim, inserimos a informação do servidor definindo o url e porta disponível, no nosso caso a 1883, como mostra figura 17 no trecho do programa do módulo andar. Essa definição é realizada em todos os clientes conforme suas características.

Figura 17- Trecho do programa onde é realizado as definições do broker


Fonte: Autor.

Nas linhas de 13 a 20 da figura 17 ocorre as definições, onde:

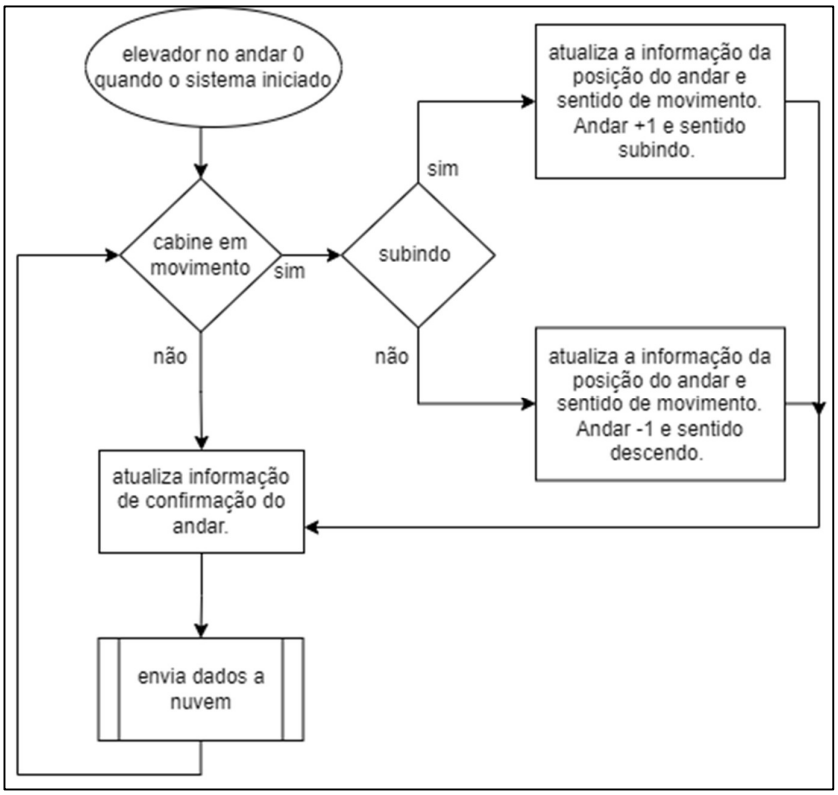
- Linha 13 - Cria-se o nome de identificação do módulo para comunicação com a nuvem, nesse exemplo temos "Andar02".

- Linhas 14 a 17 - São criadas constantes definidas com o nome dos tópicos que são acessados pelo módulo para baixar informação.
- Linhas 19 e 20 - São criadas constantes com o nome dos tópicos onde são publicadas as informações do módulo, nesse exemplo é o MA, de acordo com o que é definido na programação.

## 3.2 ME (Módulo Elevador)

O Módulo Elevador é inserido na cabine de cada elevador, tendo como objetivo principal identificar o posicionamento da cabine e compartilhá-lo através da nuvem a cada segundo. Assim assumindo o papel de cliente publicador perante o protocolo MQTT. O mesmo é fixado na cabine externa sendo acionado por atuadores alocados no poço do elevador conforme o movimento realizado pela cabine. Uma outra função é a confirmação para sistema de solicitação dos andares onde a cabine está passando, que entenderemos mais à frente a importância dessa ação. O fluxograma da figura 18 exemplifica a funcionalidade do módulo.

Figura 18- Fluxograma de funcionalidade do módulo que fica no elevador




Fonte: Autor.

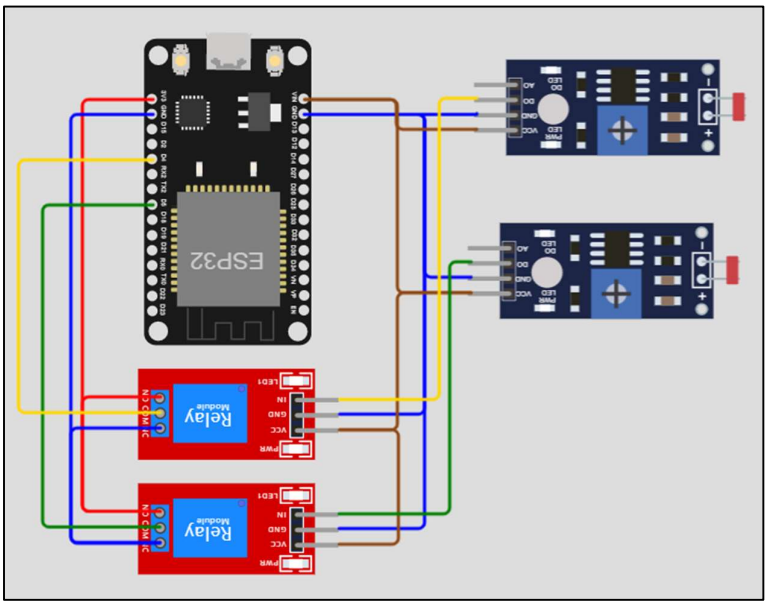
Nas condições iniciais do módulo o mesmo necessita que ao ser iniciado a cabine esteja no andar mais baixo possível. Com isso o sistema entende que é o primeiro compartimento do prédio na direção vertical.

Como podemos observar no fluxograma da figura 19 o módulo literalmente captura informações por meio do movimento do elevador e as disponibiliza no Broker em tópicos específicos. Esses envios de dados ocorrem a cada 1 segundo. Um movimento é identificado mediante a cada mudança da localização da cabine entre andares no poço. Caso não haja mudança do posicionamento da cabine, são reenviados os últimos dados, completado 1 segundo do último envio, retornando ao ponto de verificação de movimento da cabine. Se for identificado movimentação inicia-se a verificação do sentido de deslocamento da cabine para então atualizar a informação do andar que se encontra e sentido de movimento realizado pelo elevador. Finalizando com o envio a nuvem e retornando à verificação de movimento da cabine.

### 3.2.1 Hardware

Este módulo é composto por dois módulos sensores ópticos, dois módulos relés de um canal e uma placa de prototipagem ESP32 conforme o esquema na figura 19. Em que, todos são alimentados a 5V.

Figura 19- esquemático do ME

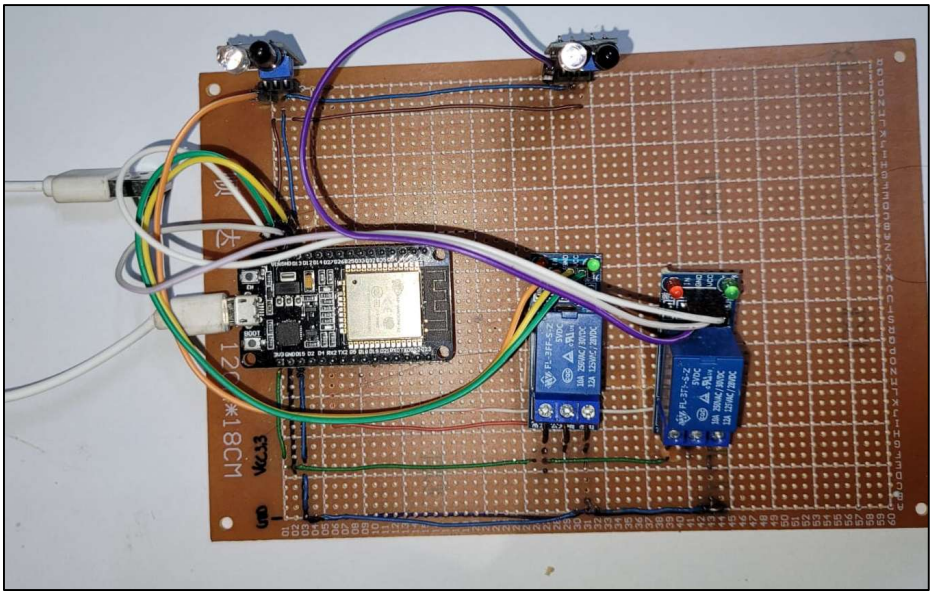



Fonte: Autor

Os sensores ópticos do tipo infravermelho emitem feixe de luz e recebe o reflexo quando há algo na frente. Assim identificando os atuadores existentes no poço do elevador, que conforme o movimento da cabine, ocorre uma ordem de atuação dos sensores. Desta forma identifica-se o sentido de deslocamento dela. Desta forma o sinal emitido pelo par de sensores é enviado a entrada dos módulos relé.

Os módulos relés realizam o acionamento de suas saídas a partir dos sinais de entradas emitidos pelos sensores já falado anteriormente. Onde temos o ponto comum da saída de cada relé para emitir o sinal para a placa de prototipagem. Como sabemos a saída do relé de um canal é composto por dois contatos. Um normalmente aberto (NA) e um normalmente fechado (NF) com um ponto em comum entre eles. Com isso, quando não há sinal nível lógico alto na entrada o sinal enviado pelo ponto comum é o inserido no contato NF e quando o sensor é acionado o sinal passar a ser o presente no contato NA. Os sinais interligados na saída dos módulos são gerados pela própria placa de prototipagem, 3.3V no NA e GND no NF conforme mostra a figura 20.

Figura 20- ME (Módulo Elevador)



Fonte: Autor

Por fim, temos a placa de prototipagem nodemcu-32 onde fica gravado o programa para processamento. O Esp32 trabalha numa frequência de clock que pode chegar até 240MHz de acordo com o modelo, sendo o mais comum 160 MHz, o

mesmo possui 2 microprocessadores Xtensa ® 32 bit LX6 com velocidade de processamento até 600 DMIPS e apresenta as seguintes características principais ao nosso projeto:

- 2 interface I2C de 5Mbps.

- 34 gpios.

- Interface WiFi 802.11bgn – 802.11n (2.4Hz), até 150Mbps.

- Consumo de WiFi 240 mA.

- ROM interna de 448KBytes.

- RAM interna de 520KBytes.

- Dois Grupos de timers – 4 timers de 64 Bits.

Nesta placa fica armazenada toda a programação do módulo, para o processamento e compartilhamento dos dados. Os sinais de saída dos relés são inseridos nas GPIOs de entrada D4 e D5. Conforme a lógica do programa é processada as informações dos elevadores são enviados aos tópicos correspondentes no broker.

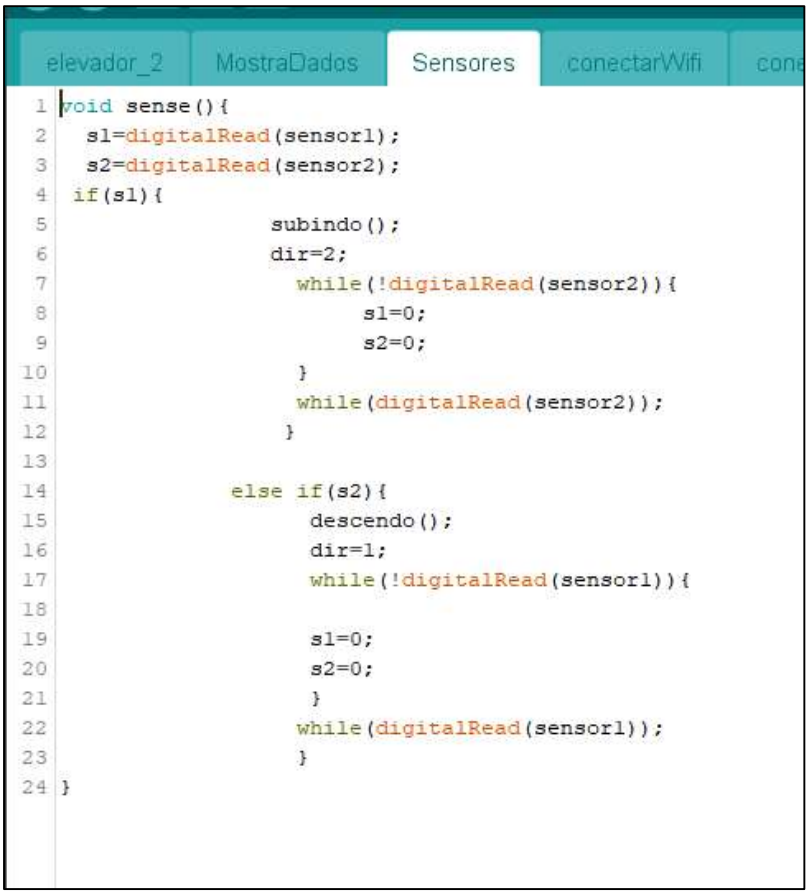
### 3.2.2 Software

O programa principal é estruturado da seguinte maneira: inicia-se com as definições de protocolo MQTT e conectividade, após isso ocorre a chamada de uma sequência de funções para o processamento de leitura dos sensores até o envio das informações ao servidor. Onde podemos destacar as funções: sense(), mostrarDados() e as funções suindo() e descendo() que são chamadas dentro da primeira citada.

A função sense() é responsável por realizar a captura das leituras dos sensores e baseada na programação de dados estruturados tipo LIFO (Last-in-first-out ou primeiro que entra é o último que sai). A partir da ordem de acionamento dos sensores é identificado o movimento da cabine. Em que dependendo do sentido, é realizada chamada das funções subindo(), caso o deslocamento seja de baixo para cima ou realiza chamada da função descendo() caso contrário conforme podemos ver na figura

21, nas linha 5 e 15 respectivamente. De modo geral, a função sense() tem o papel de realizar modificações da variável “dir” (sentido do movimento da cabine) e variável “andar” através sub-rotinas subindo() ou descendo(). Com isso, é identificado o andar atual da Cabine e seu sentido de deslocamento. Para melhor entendimento o fluxograma da figura 22 exemplifica.

Figura 21-Trecho do programa onde identifica o movimento do elevador




Fonte: Autor.

A função subindo() adiciona uma unidade a variável “andar” e atribui o valor 2 a variável “dir”.

A função descendo() subtrai uma unidade da variável “andar” e atribui valor 1 variável “dir”.

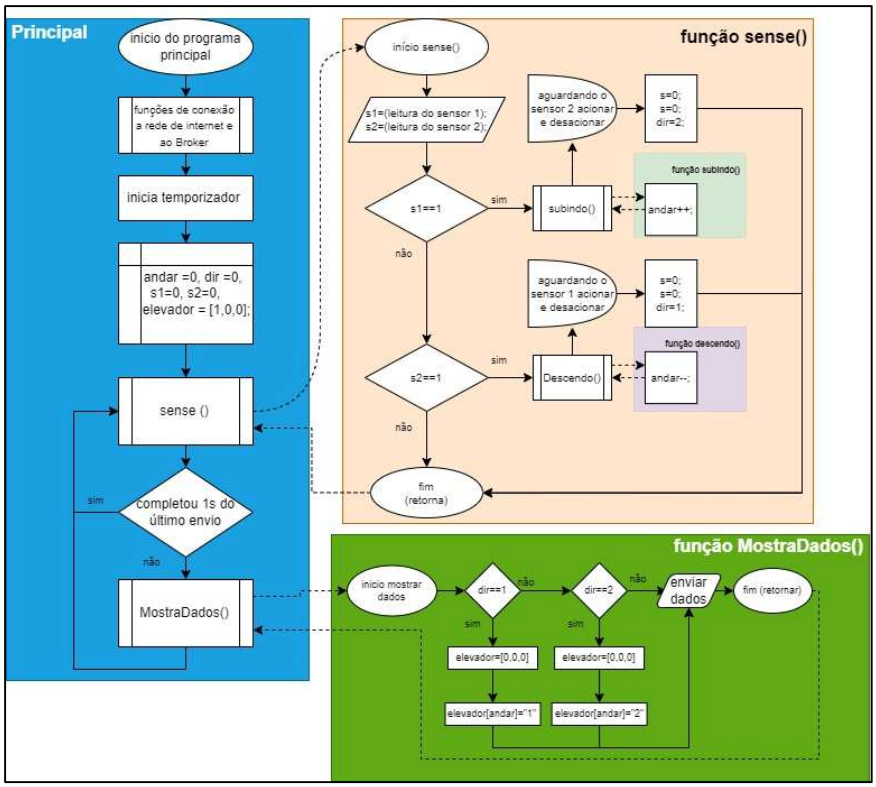
A função mostrarDados() é responsável em compartilhar as informações de posicionamento do elevador a cada 1 segundo a partir das variáveis “dir” e “andar” da seguinte maneira: É realizado o zeramento de um vetor chamado “elevador”, que tem

o número de posições igual ao número de andares do prédio. Neste vetor na posição onde índice do mesmo é igual ao valor da variável “andar” é inserido o valor da variável “dir”. Por fim é realizado envio do vetor “elevador” ao broker no tópico “elev01”. Com isso, é enviado o vetor com as informações do andar que a cabine se encontra e o movimento que ela está realizando. Exemplo:

[0,2,0] - O “2” indica que o elevador está com movimento de subida e a posição que o numeral está inserido, no índice 1, indica que a cabine se encontra no andar 1.

[0,1,0] - O “1” indica que o elevador está com movimento de descida e a posição que o numeral está inserido, no índice 1, indica que a cabine se encontra no andar 1.

Figura 22-Fluxograma do funcionamento do programa para conhecimento do andar que o elevador 1 se encontra.




Fonte: Autor.

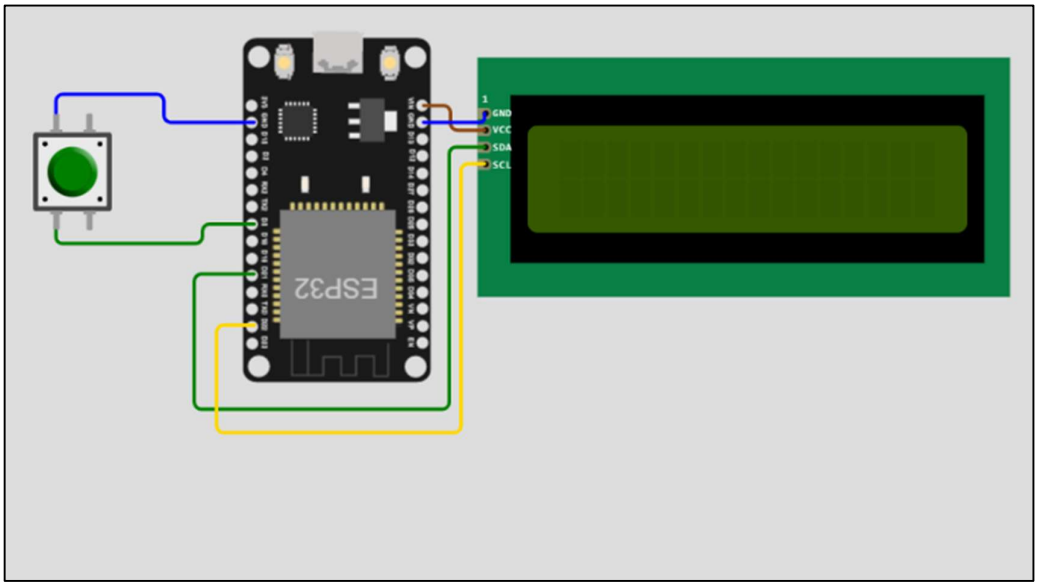
Uma outra função desempenhada pelo módulo é de cliente assinante do tópico “chama”. Onde neste, temos um vetor com informações de solicitações de chamadas do elevador realizado pelo usuário no andar correspondente. No momento que a

cabine está no mesmo andar solicitado é atualizado com o digito 0 na posição do vetor correspondente ao andar, assim indicando que foi concluído o atendimento a aquele chamado e envia novamente o vetor ao tópico “chama”.

## 3.3 MA (Módulo andar)

Este módulo fica localizado no lado externo da cabine no seu respectivo andar, onde ele desempenha duas funções: disponibilizar ao usuário o andar em que a cabine se encontra dentro do poço através de um Display LCD de 16x2 e realizar chamada do elevador mais viável. De forma geral é onde ocorre a interação com o usuário e são obtidos os resultados do sistema proposto. Conforme o esquemático apresentado na figura 23 a interação ocorre através de um display e um botão.

Figura 23- Esquemático do módulo andar.



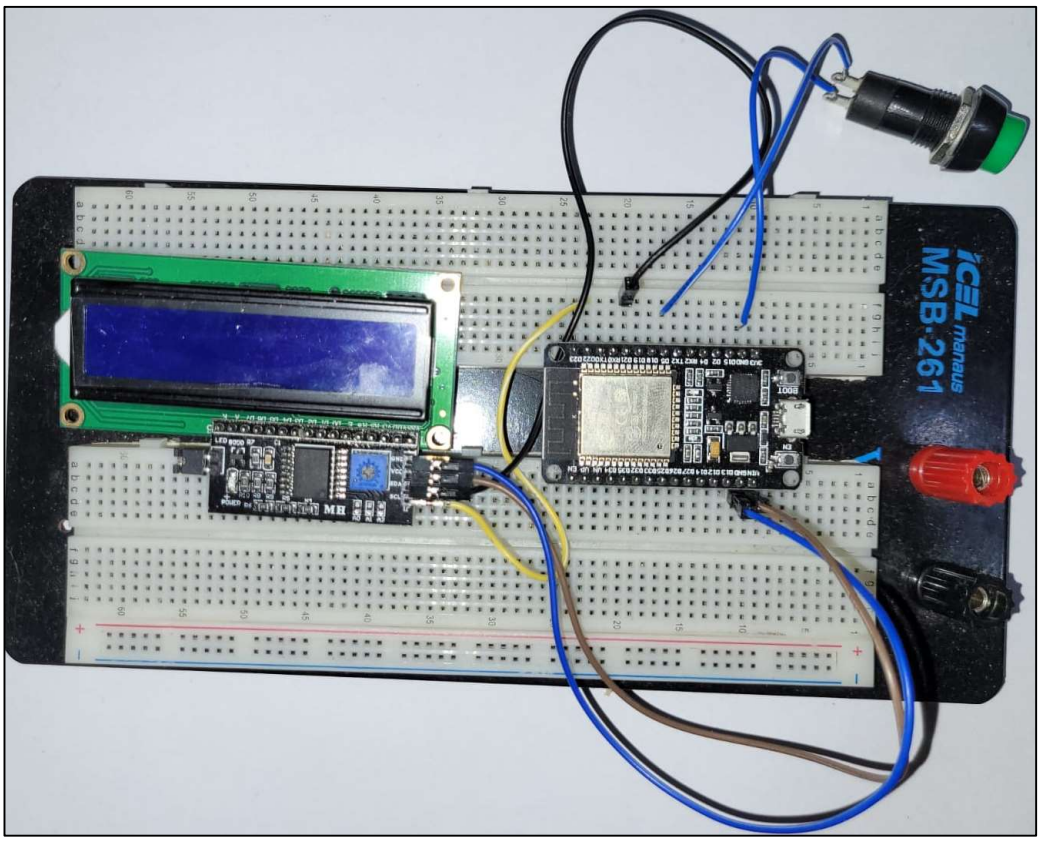

Fonte: Autor.

### 3.3.1 Hardware

Foi iniciado o desenvolvimento da montagem do circuito no protoboard conforme a figura 24, no qual é necessário ter troca de dados entre o microcontrolador ESP32 e seus periféricos (display e botão). O display LCD é interligado na placa de prototipagem NodeMCU-32 por meio de uma placa de interface, chamado de módulo I2C. O qual permite ser utilizado quatro fios, sendo um par, VIN e GND, para

alimentação e outro para comunicação serial SDA (Serial data) encarregado de enviar e receber dados e o SCL (Serial clock) para criar um clock que faz o sincronismo dos sistemas para a transferência de dados, que são ligados nos pinos GPIO21 e GPIO22 respectivamente da placa de prototipagem. Já o botão pulsador é inserido com uma extremidade no GND e a outra no GPIO5 do ESP32. Quando o botão é pressionado, a entrada do microcontrolador fica em nível lógico baixo (0V), uma vez que os contatos interligam a entrada ao GND, assim reconhecendo o evento externo.

Figura 24- MA (Módulo andar)



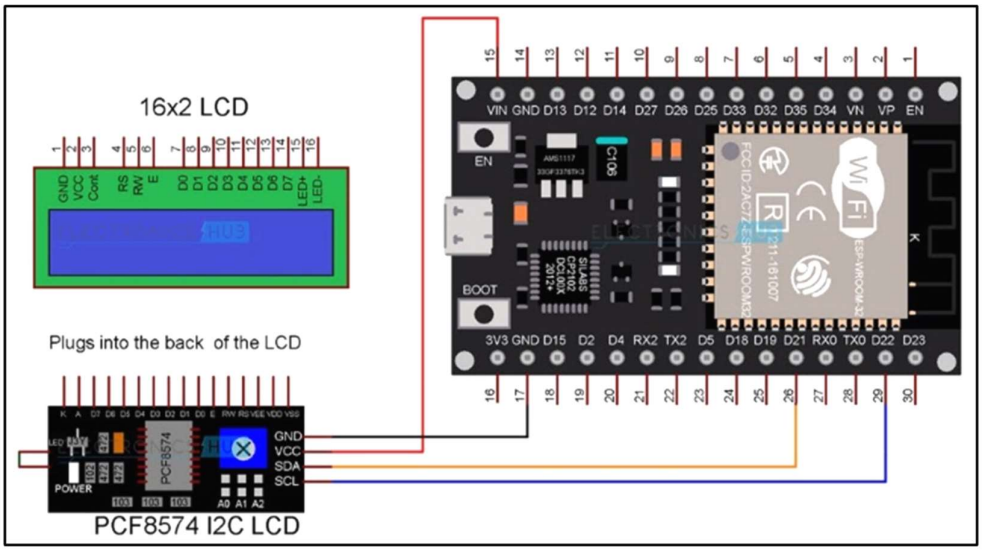

Fonte: Autor.

Na placa de prototipagem nodemcu-32 ocorre todo o gerenciamento do módulo com o sistema através da internet, pois é este *firmware* que recolhe informações da nuvem e mostra no display. Quando pressionando o botão, através das informações baixadas realiza a escolha do elevador mais viável e envia ao servidor a solicitação.

3.3.1.1 Display matricial com o protocolo I2C

Para interação do usuário foi utilizado um display de LCD matricial 16x2 com tela de cristal líquido, onde é apresentada a localização do elevador e quando solicitado mostrar qual cabine o usuário deve se dirigir. De acordo com (NUNES, 2013) é um dispositivo amplamente utilizado por sua facilidade na implementação, em que o mesmo permite mostrar uma variedade de números símbolos dentre outros. Display Matricial 16x2, pois conta com 16 colunas e 2 linhas totalizando 32 caracteres para escrita. Este modelo tem controlador próprio que permite comunicação com a placa através do plano de protocolos. Neste caso será utilizado protocolo I2C para comunicação entre o ESP32 e a tela de lcd conforme figura 26.

Figura 25- Ligação do display ao ESP32 através da placa I2C.




Fonte: LaptrinhX

Por fim, foi inserido um botão Push bottom para que o usuário realize a solicitação de chamada do elevador. Como podemos observar a interação com o usuário, apesar de pouca e simples, o programa desse modo tem grande importância devido realizar os cálculos para as chamadas inteligentes que veremos a seguir.

3.3.2 Software

É um programa estruturado sendo um principal que logo de início realiza chamadas de funções de comunicação do protocolo e de conexão Wi-Fi. Que após

isso, em seu loop é realizada uma sequência de chamadas de função. Inicia recebendo informações dos elevadores como posicionamento e sentindo. Dando continuidade verificando o estado do botão para realizar os cálculos das distâncias dos elevadores e escolha do que seja viável ao atendimento. Finalizando com envio da solicitação de chamada ao servidor no tópico correspondente ao controlador do elevador escolhido através do programa, caso o botão seja pressionado. Onde destacou-se as rotinas: Principal do módulo andar, mantemConexoes, conectarWifi, conectaMQTT, lcd_start, but, imprimir_display e recebepacote.

No programa principal de início são realizadas às chamadas de biblioteca, declarações das variáveis globais, definições do protocolo MQTT e de comunicação do Wi-Fi. Em seguida e realizado as configurações de identificação do módulo e os tópicos que serão utilizados durante a execução do programa tanto para publicação, quanto para assinatura no servidor. Inicia-se com a chamada da função lcd_Start() para inicialização da tela de lcd. Em sequência ocorre a chamada das funções de conectividade. Primeiro é chamado a conectaWi-Fi(), para colocar dispositivo conectado à rede internet através da rede Wi-Fi, utilizando definições de usuário e senha realizados no início do código. Para finalizar o setup do programa principal é inserido duas rotinas. Uma é conectaMQTT para carregamento do servidor que trabalharemos e a função callback(), onde esta é uma interrupção que é executada toda vez que o tópico no servidor em que o módulo está inscrito é atualizado.

Ainda no código principal, porém na parte de loop do mesmo, onde está a sequência que se repetirá durante toda execução, temos as rotinas que mantém o funcionamento do módulo. Logo de início tem-se a rotina mantemConexoes() que garante a conexão do módulo com à rede Wi-Fi e com Broker MQTT. Em seguida, ocorre a comunicação com os tópicos em que o módulo é assinante no servidor a cada 50ms. Com isso quando ocorre atualização do tópico em que o módulo está conectado ocorre a interrupção e a função recebePacote() é chamada.

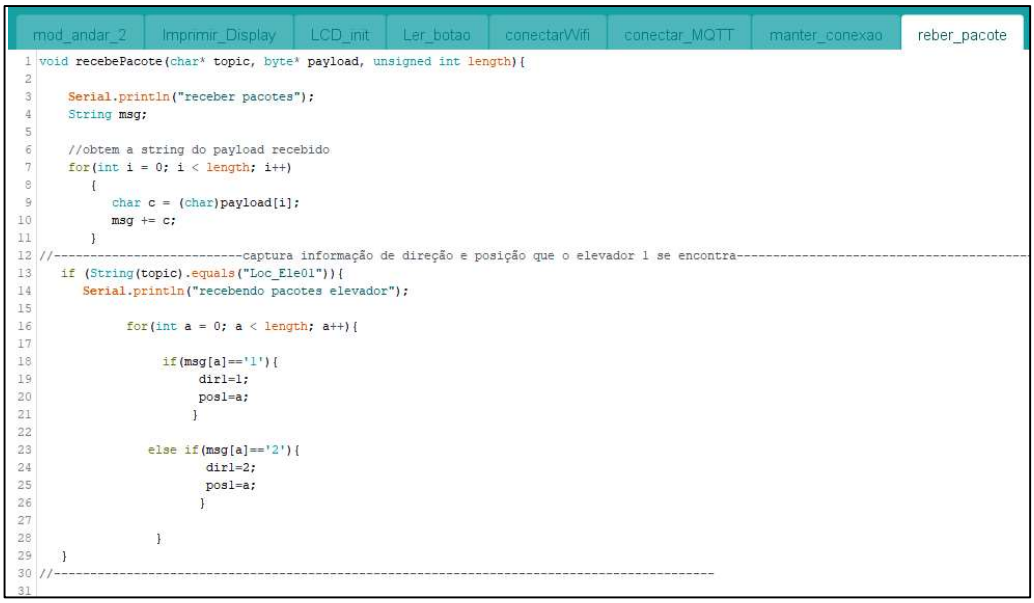
A função recebePacote(char*topic, byte*Payload, unsigned int length) é quem recebe as informações que foram atualizada na nuvem. Através dos ponteiros topic, Payload e a variável length é baixado da nuvem o nome do tópico atualizado, a mensagem transmitida e o tamanho da mensagem. A partir daí trabalha-se com esses dados para serem direcionados a variáveis correspondentes.

Para atualização do tópico “Loc_Ele01”, apresentado na figura 26, é capturado informação de direção e posição que o elevador 1 se encontra. É realizado uma varredura na string mensagem que recebeu através do ponteiro Payload, onde for encontrado o caractere “1” ou “2” o mesmo é salvo junto a sua posição no vetor nas variáveis “dir1” e “pos1” respectivamente (linhas 16 a 28). Com isso, é identificado o sentido que elevadores se move e o andar que o mesmo está.

Numeral “1” - Elevador descendo.

Numeral “2” – Elevador subindo.

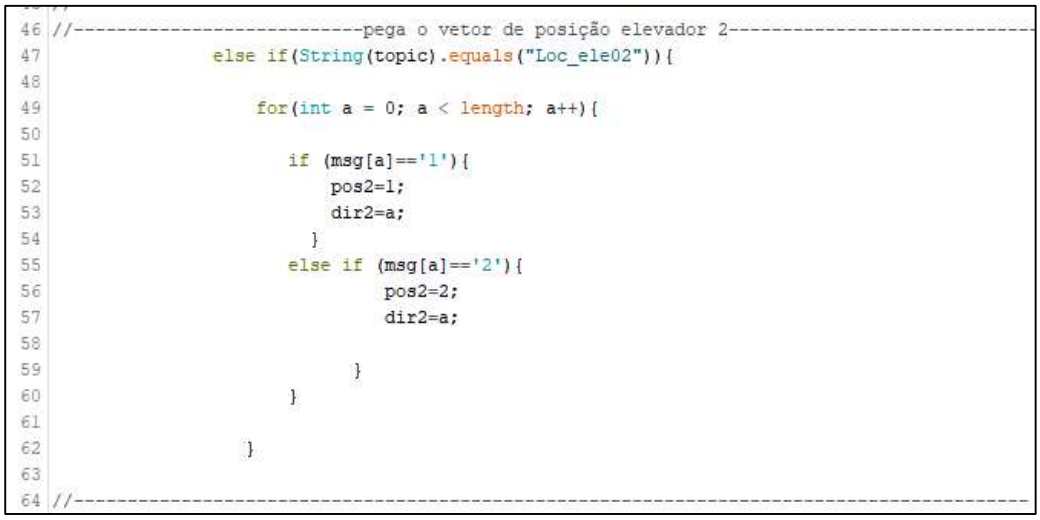
Figura 26- Trecho do programa que captura informação do elevador 1




Fonte: Autor.

O tópico “Loc_Ele02” tem mesma características do “Loc_Ele01”, porém os dados são referentes ao elevador 2 e as informações são transferidas às variáveis “pos2” e “dir2” de posição e direção respectivamente nas linhas 52 e 53 ou 56 e 57, dependendo do movimento do elevador, conforme o trecho do programa na figura 27.

Figura 27- Trecho do programa que captura informação do elevador 2



```
46 //----------------------------pega o vetor de posição elevador 2-----------------------------
47                else if(String(topic).equals("Loc_ele02")){
48
49                    for(int a = 0; a < length; a++){
50
51                       if (msg[a]=='1'){
52                           pos2=1;
53                           dir2=a;
54                         }
55                       else if (msg[a]=='2'){
56                                pos2=2;
57                                dir2=a;
58
59                             }
60                       }
61
62                   }
63
64 //-------------------------------------------------------------------------------------------
```

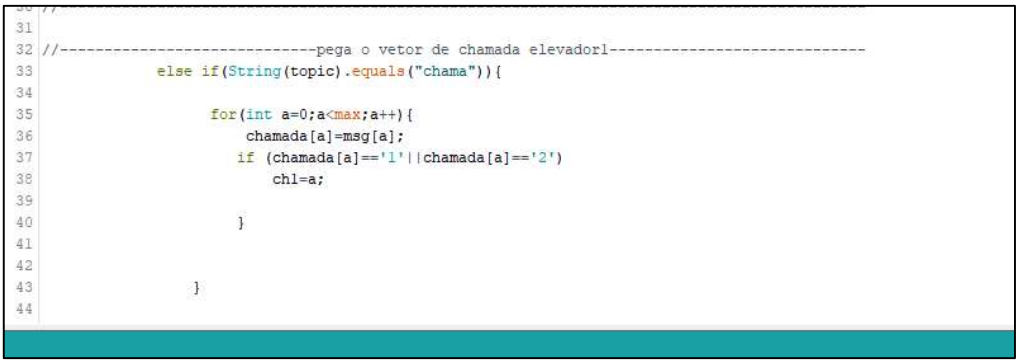
Fonte: Autor.

Outro tópico é o “chama”, que disponibiliza aos assinantes um vetor com informações referente às solicitações de chamadas do elevador 1, isso depois da escolha da cabine mais viável ao atendimento, onde o índice do vetor representa os andares do prédio e os caracteres contidos nele são:

- 0 (andar sem chamadas ao elevador)
- 1 (botão de chamadas do elevador foi pressionado)
- 2 (botão de chamada do elevador não pressionado).

O bloco responsável em receber informações desse tópico inicia na linha 33 do trecho de programa mostrado na figura 28, em que temos uma condição de comparação da informação do nome do tópico, contida na variável “topic”, ao nome “chama”. Caso a condição for verdadeira, indica que a atualização foi no tópico em questão, então inicia-se a varredura na variável tipo string “msg”, copiando os caracteres para um vetor “chamada”. Em paralelo, caso encontre caracteres diferente de 0, a posição é armazenada na variável “ch1”. Com isso é capturado o andar mais alto que foi solicitado e o elevador ainda não chegou, conforme o trecho compreendido entre as linhas 35 e 40 do programa na figura 28.

Figura 28- Trecho do programa que captura informação de chamadas do elevador1.

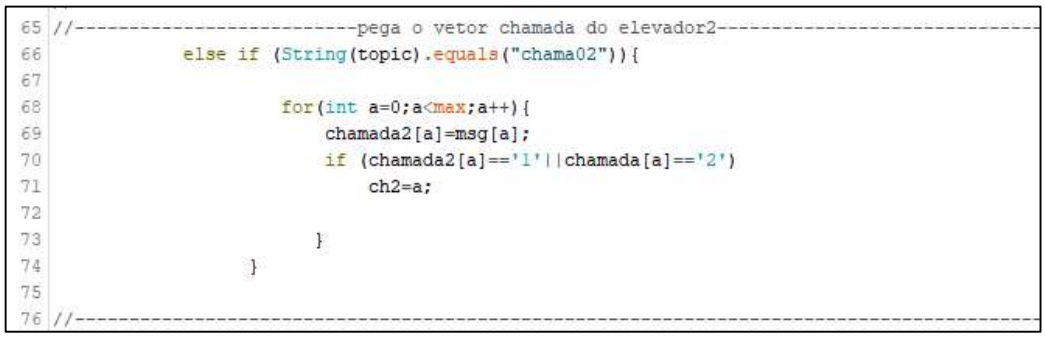


```
31
32 //----------------------------pega o vetor de chamada elevador1----------------------------
33             else if(String(topic).equals("chama")){
34
35                   for(int a=0;a<max;a++){
36                       chamada[a]=msg[a];
37                      if (chamada[a]=='1'||chamada[a]=='2')
38                          ch1=a;
39
40                      }
41
42
43                 }
44
```

Fonte: Autor.

Por fim, o tópico chama02 tem função idêntica ao tópico “chama” com a diferença que as informações são referentes ao elevador 2 e a variável de captura do andar mais alto é “ch2” conforme o trecho do programa na figura 29.

Figura 29- Trecho do programa que captura informação de chamadas do elevador 2.



```
65 //----------------------------pega o vetor chamada do elevador2-----------------------------
66             else if (String(topic).equals("chama02")){
67
68                     for(int a=0;a<max;a++){
69                         chamada2[a]=msg[a];
70                         if (chamada2[a]=='1'||chamada[a]=='2')
71                             ch2=a;
72
73                        }
74                   }
75
76 //-------------------------------------------------------------------------------------------
```

Fonte: Autor.

As próximas duas funções ocorrem após a combinação dos armazenamentos das rotinas anteriores. São as funções de interação com o usuário onde tem-se: imprimir_ display() que é responsável em imprimir as informações capturadas do tópico referente ao elevador que o módulo interage, exemplo se o módulo de estar lincado ao elevador 1, mostrará o andar em que o mesmo se encontra e se está subindo ou descendo.

A função But() é encarregado de verificar leitura do botão e solicitar um elevador, caso o mesmo seja pressionado. É realizado cálculo da distância dos elevadores ao andar solicitado, assim verificando qual o mais viável ao atendimento. Atualiza o vetor “chamada” desse elevador, com carácter “1” na posição correspondente ao andar em

que o módulo está inserido, concluindo com o envio do vetor ao broker. Quando o botão é solto ocorre apenas uma atualização do vetor “chamada” no lugar do “1” é inserido “2”, então é realizado o reenvio desse vetor de solicitação ao servidor.

Foram criadas três variáveis globais, sendo as de controle, variável “flag” para não enviar várias vezes o status do botão, variável “button” identifica status do botão e a variável “soltar”, que guarda a informação de qual foi escolhido quando o botão é pressionado pelo usuário. A última variável citada é utilizada quando o botão deixa de ser pressionado, enviando ao servidor o vetor “chamada” com o caractere “2” no índice correspondente ao andar que o módulo está inserido, desta forma atualizando o tópico referente ao elevador correto.

Em relação a escolha inteligente do elevador foi realizado como idealiza o fluxograma na figura 30. Após capturar as informações das cabines e andares na função “receberPacotes” é realizado o cálculo da distância do elevador mais próximo ao andar de solicitação seguindo seu fluxo de movimento. Como estamos trabalhando com vetor, é feito uma varredura no vetor, porém não é armazenado o vetor de posições nem o de chamada, apenas as informações de direção, posição e solicitações.

Então, através da variável “pos1” a variável de controle “i” é inicializada com a posição em que se encontra o elevador. Dependendo do movimento que o elevador, identificado pela variável “dir1”, essa variável “i”, aumenta ou diminui uma unidade de seu valor. Caso o elevador esteja no movimento de subida a mesma acumula uma unidade e descendo subtrai uma unidade do seu valor com a condição de parada “i=x”. Onde x é uma constante com o valor do andar que o módulo (MA) está inserido, a cada mudança do valor de “i” uma variável “cont” iniciada em zero no início da função, soma uma unidade, ou seja, “cont” armazenará a quantidade de andares que a cabine do elevador vai se mover até alcançar o andar em que o módulo se encontra, no caso “x”.

A variável “i” terá mudança no sentido da variação do seu valor sempre que o andar solicitado estiver oposto ao movimento da cabine do elevador. Isso ocorrera em duas situações:

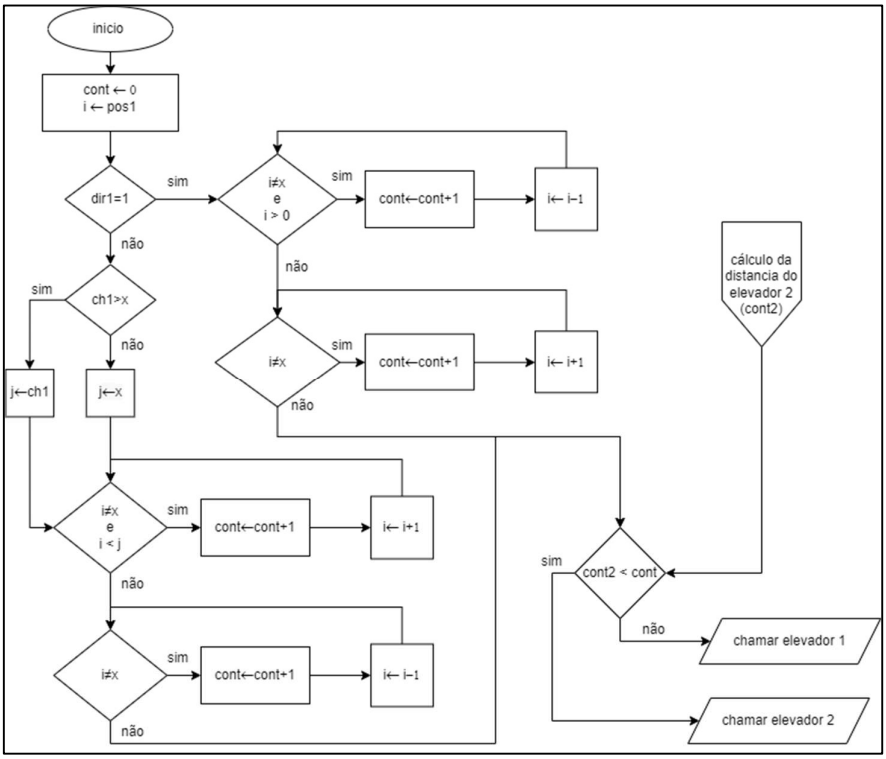
1. Quando o andar de solicitação “x”, for menor que o andar que a cabine se encontra e a mesma esteja em movimento de subida, desta forma o

valor de “i” incrementa até se igualar ao valor de “ch1” e então começa a decrementar até que o valor seja igual ao valor de “x”.

2. Quando o andar de solicitação “x”, for maior que o andar que a cabine se encontra e a mesma esteja em movimento de descida, desta forma o valor de “i” decrementa até se igualar a 0 e então começa a incrementar até que o valor seja igual ao valor de “x”.

Essa contagem de movimentos é realizada para as duas cabines, sendo que “cont” armazena o valor da distância da cabine do elevador 1 e “cont 2” armazena o valor da distância da cabine do elevador 2. Após as duas contagens é realizado a chamada do elevador correspondente a menor distância. Na figura 31 mostra o fluxograma para captura da distância de um elevador até o anda de referência “x”.

Figura 30- Fluxograma de cálculo da distância da cabine ao andar solicitado.




Fonte: Autor.

Após a seleção do elevador mais próximo é enviado ao tópico do elevador correspondente. Na figura 31 podemos ver o trecho de chamada da cabine. Pode-se observar que há a condição de comparação entre os contadores, “cont” e “cont2”, na linha 68, se verdadeira, é inserido diretamente na posição “x” do vetor “chamadas” o valor 1, botão pressionado, e a variável “soltar” recebe o valor 1, que será usado na

próxima condição quando o botão for solto, e envia o vetor “chamada” ao tópico de publicação, chama, representado pela constante “TOPIC_PUBLISH”.

Caso a condição da linha 68 seja falsa, o programa excuta a linha 75, onde o elevador 2 e o mais próximo do andar solicitado. Nesse bloco o que diverge da primeira condição é:

- O valor que a variável “soltar” recebe que é 2, indicando que o elevador 2 foi o indicado.
- O vetor “chamada2” que recebe o valor 1, indicando que o botão foi pressionado, posição x.
- E o tópico de publicação deixa de ser o “chama” e sim o “chama02”, representado pela constate “TOPIC_PUBLISH1”.

Já na linha 84 temos uma nova condição, verifica se o botão foi solto. Caso seja verdadeira as linhas seguintes, verifica o valor da variável “soltar”:

- Se ela for 1, indica que o foi no elevador 1 que foi pressionado o botão. Com isso, é inserido o valor 2 na posição “x” no vetor “chamada”, e o mesmo é publicado no tópico “chama” do servidor.
- Se não, significa que foi o elevador 2 solicitado. Com isso, o botão que foi solto foi o do elevador 2. Sendo assim, é inserido o valor 2 na posição “x” no vetor “chamada2”, e o mesmo é publicado no tópico “chama02” do servidor.

Figura 31- trecho do programa referente a função but() realiza a chamada do elevador.

```
66
67
68            if(cont<=cont2){
69             soltar=1;
70             chamada[x]='1';
71             MQTT.publish(TOPIC_PUBLISH, chamada);
72             lcd.setCursor(0, 5);
73             lcd.print("chamada E1");
74               }
75           else{
76                soltar=2;
77                chamada2[x]='1';
78                MQTT.publish(TOPIC_PUBLISH1, chamada2);
79                lcd.setCursor(0, 5);
80                lcd.print("chamada E2");
81
82               }
83    }
84  if(digitalRead(button)==1 && flag==1){
85          flag=0;
86          if(soltar==1){
87                  chamada[x]='2';
88                  MQTT.publish(TOPIC_PUBLISH, chamada);
89                  lcd.clear();
90             }
91
92          else if(soltar==2){
93                       chamada2[x]='2';
94                       MQTT.publish(TOPIC_PUBLISH1, chamada2);
95                       lcd.clear();
96                  }
97
98     }
99
100 }
```

Fonte: Autor.

## 3.4 MC (Módulo Controlador)

Essa parte do protótipo é composto por um microcontrolador NodeMCU8266 (ESP-12), um módulo de relé de três canais e como forma de visualização temos três relés representando a solicitação de chamada de cada andar. Este módulo é embarcado no controlador do elevador. Conforme ocorre a solicitação dos andares do usuário o relé correspondente é acionado, através desse sinal saberemos qual andar solicitou o elevador. Os ledes indicaram o estado dos botões dos andares do elevador durante a solicitação de chamada.

### 3.4.1 Hardware

O principal componente é a placa de prototipagem NodeMCU8266, o nome se dá pela combinação de “nó” e “MCU” (unidade de microcontrolador), logo o mesmo é um firmware de código aberto construído pela empresa Espressif Non - Os SDK para o Esp 8266. O hardware é uma placa que integra uma USB em uma placa menor com MCU e antena. Estamos utilizando o ESP-12 do ESP8266, que é um Soc (System on chip) com wi-fi integrado ao núcleo tensilica Xtensa LX106. Em que há um ESP8266 com o microcontrolador Tensilica L106 de 32 bits. Onde e realizado todos os cálculos e controles de entradas e saídas conforme execução do programa. Que a mesma pode ser alimentada tanto pelos pinos como pela micro-usb.

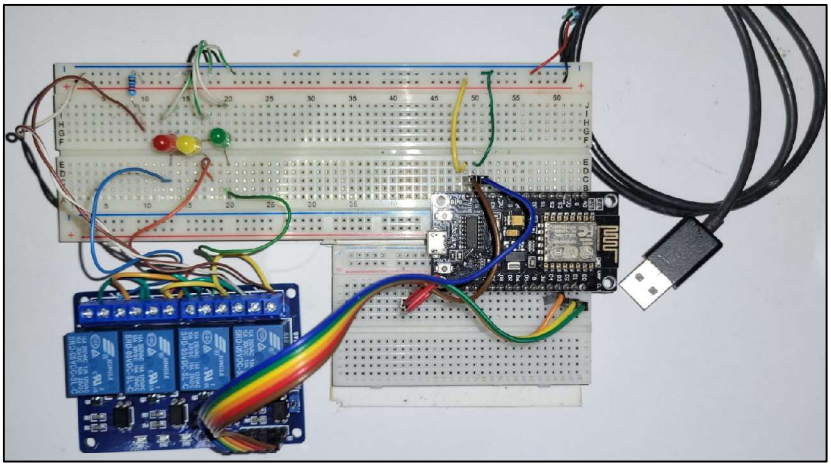
Outro componente é o relé de três canais, em que cada canal representa um andar, e a saída de cada canal fica interligado paralelamente aos sinais dos botões existentes no elevador. Foram colocados leds para verificação desses sinais de saída.

Temos uma alimentação por uma fonte de 5V no Vim do microcontrolador e no Vcc do módulo relé, e 0V no GND de ambos. Tem como principal função interligar o sistema ao controlador do elevador. Quanto ao restante contamos com a GPIO (D1, D2 e D4) do ESP interligado nas entradas (IN1, IN2 e IN3) respectivamente do banco de relé. Na saída dos contatos dos relés, é inserido 5V em série com uma resistência 220 ohms ligadas em série com os contatos normalmente abertos (NA) de todos os canais e 0V nos contatos normalmente fechado (NF).

O comum de cada canal está seriado individualmente um led (canal1, 2 e 3 estão ligados respectivamente nos leds vermelho, amarelo e verde) como forma de visualizar o comportamento das saídas conforme a figura 32.

Com o protótipo em funcionamento no prédio os sinais de alimentação dos botões ficam inseridos na NA em vez da fonte de 5V, já o comum de cada canal seria interligado diretamente na entrada do controlador correspondente ao andar de que receba o sinal do botão de chamada do elevador. Ou seja, o botão do elevador ficara em paralelo ao contato do relé normalmente aberto.

Figura 32- Módulo controlador (MC)



Fonte: Autor.

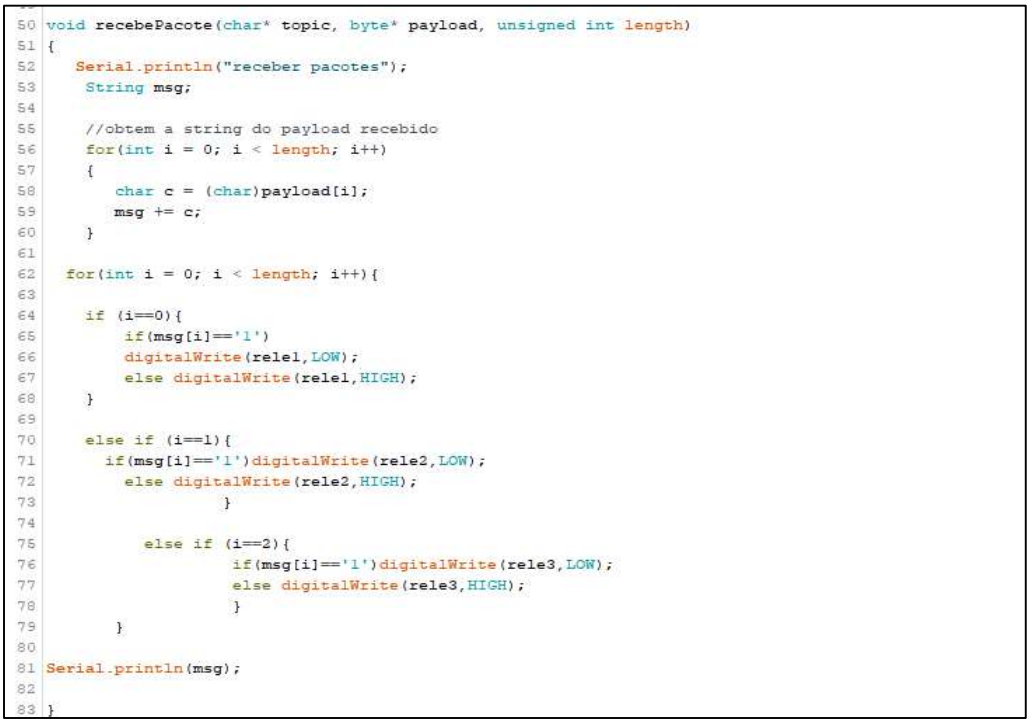
### 3.4.2 Software

Quanto a programação é ligeiramente simples apresentando as mesmas funções de conectividade dos MA e ME, de wifi e conexão com o servidor via MQTT. Porém exercendo a função de cliente assinante, ou seja, *Subscribe*. Da linha 50 a 60 de forma idêntica ao que foi visto no módulo MA recebe informação do servidor, que é um vetor de caracteres publicado pelos módulos existente nos andares (MA) quando é solicitado uma chamada do elevador ao andar. No momento que a cabine está no mesmo andar que foi solicitado o mesmo atualiza a informação do tópico “chama” se for o elevador 1 ou “chama02” se for o elevador 2 como o valor 0.

É feito uma varredura em uma string “msg”, utilizando a variável “i” como controle quando é encontrado o caractere “1” no vetor, significa que o botão está pressionado, é emitido um sinal de nível logico alto na GPIO corresponde ao andar identificado pelo índice, variável “i”, conforme mostra as linhas (65 e 66), 71 e 76 da figura 33. Quando o caractere é diferente de “1” então é enviado um sinal de nível lógico baixo a GPIO correspondente ao andar podendo ser 0 - nenhuma chamada realizada do andar

correspondente ou 2 - botão solto, conforme trecho do programa na figura 33 nas linhas 67, 72 e 77.

Figura 33 - Trecho do programa mostrando o acionamento do relé



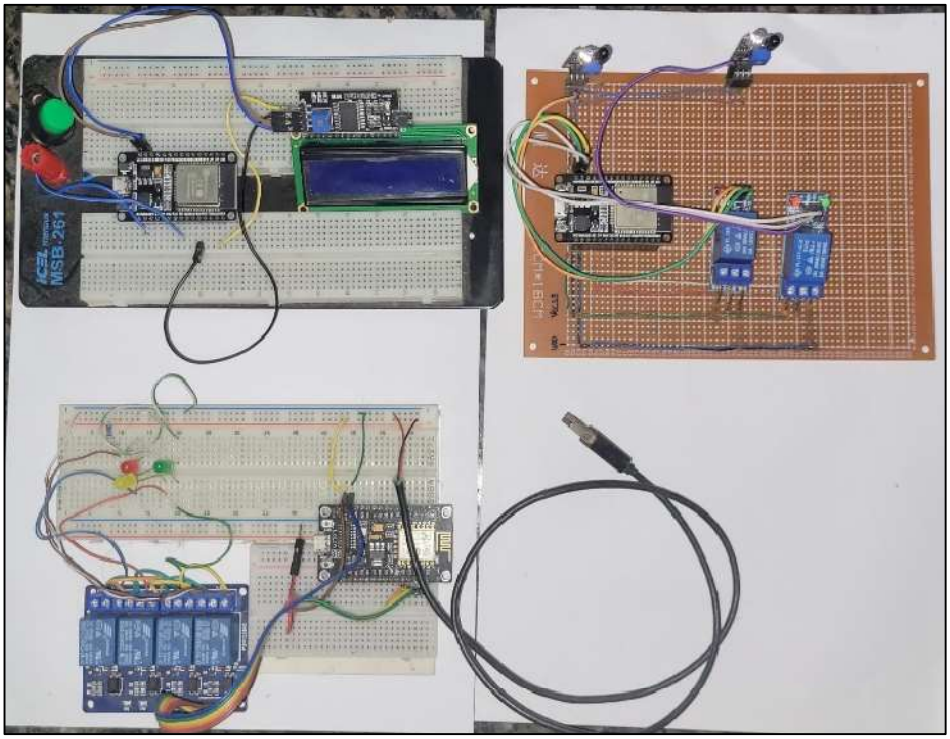
```
50 void recebePacote(char* topic, byte* payload, unsigned int length)
51 {
52    Serial.println("receber pacotes");
53     String msg;
54
55     //obtem a string do payload recebido
56     for(int i = 0; i < length; i++)
57     {
58        char c = (char)payload[i];
59        msg += c;
60     }
61
62   for(int i = 0; i < length; i++){
63
64     if (i==0){
65         if(msg[i]=='1')
66         digitalWrite(relel,LOW);
67         else digitalWrite(relel,HIGH);
68     }
69
70     else if (i==1){
71       if(msg[i]=='1')digitalWrite(rele2,LOW);
72         else digitalWrite(rele2,HIGH);
73                   }
74
75           else if (i==2){
76                    if(msg[i]=='1')digitalWrite(rele3,LOW);
77                    else digitalWrite(rele3,HIGH);
78                    }
79        }
80
81 Serial.println(msg);
82
83 }
```

Fonte: Autor.

# 4 RESULTADOS OBTIDOS

A principal proposta do trabalho foi a criação de um protótipo, onde foi realizado a montagem de três módulos conforme mostra figura 34 para realização dos testes. Onde sua comunicação seria totalmente via wireless, onde se teve êxito e com resultados positivos.

Figura 34- Módulos MA, ME e MC do protótipo



Fonte: Autor

## 4.1 TESTES DO PROTÓTIPO

Os testes do protótipo foram realizados de três formas. Os módulos individualmente, depois a comunicação entre o módulo elevador com o módulo andar e por fim os três módulos em conjunto.

O primeiro módulo criado foi o ME e para testes foi utilizado a porta serial. Ao acionarmos os sensores em determinada ordem, o vetor a ser enviado na nuvem alterava os seus valores de acordo com a tabela 3.

Tabela 3: Testes dos acionamentos dos sensores e o comportamento do vetor a ser enviado.

| 1° acionamento | 2° acionamento | Vetor: elevador1 |
|---|---|---|
| - | - | [0,0,1] |
| Sensor 1 | Sensor 2 | [0,2,0] |
| Sensor 1 | Sensor 2 | [2,0,0] |
| Sensor 2 | Sensor 1 | [0,1,0] |
| Sensor 2 | Sensor 1 | [0,0,1] |

Fonte: Autor, conforme os testes realizados

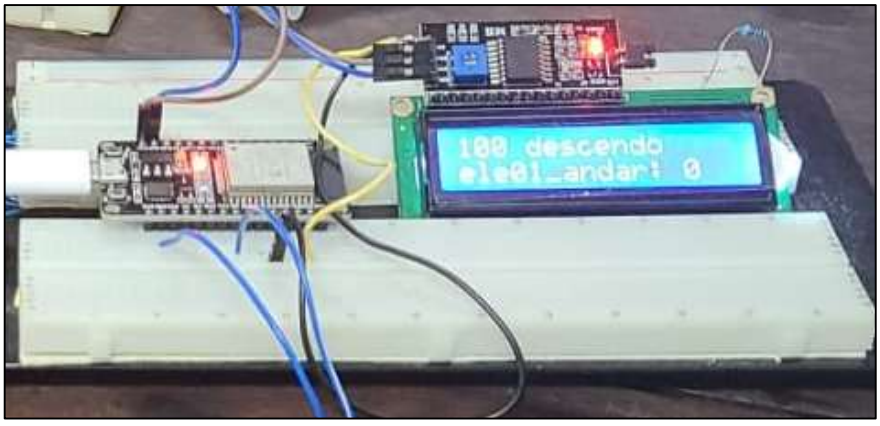
Outro teste realizado foi o acionamento de apenas 1 sensor, onde o mesmo aguardava o acionamento do próximo sensor. Em seguida foi finalizado o módulo andar (MA), que para os testes foram comunicados à rede wifi. Realizando a mesma sequência de testes anteriores, foi gerada a tabela 4. A saída para o usuário fica conforme mostrado na figura 35.

Tabela 4: Testes dos acionamentos dos sensores e o comportamento do vetor a ser enviado.

| 1° acionamento módulo ME | 2° acionamento módulo ME | Display no módulo MA |
|---|---|---|
| - | - | Andar 0 descendo [1,0,0] |
| Sensor 1 | Sensor 2 | Andar 1 subindo [0,2,0] |
| Sensor 1 | Sensor 2 | Andar 2 subindo [0,0,2] |
| Sensor 2 | Sensor 1 | Andar 1 descendo [0,1,0] |
| Sensor 2 | Sensor 1 | Andar 0 descendo [1,0,0] |

Fonte: Autor.

Figura 35- Módulo MA disponibilizando ao usuário localização do elevador 1.




Fonte: Autor.

O último teste realizado foi com os três módulos conectados à internet e mantendo a comunicação via servidor. No módulo andar, para que fosse possível realizar o teste, por falta de um segundo módulo elevador, necessitou-se que enviássemos o programa várias vezes ao microcontrolador do MA, onde alterou-se os valores iniciais das variáveis que guardavam as informações referente ao elevador 2. Os resultados são mostrados na tabela 5.

Tabela 5: Testes do acionamento do MC

| 1° acionamento módulo ME elevador 1 | 2° acionamento módulo ME Elevador1 | Valores simulados elevador 2 | Display no módulo MA Elevador1 | Quando botão acionado estado do LED amarelo 2 MC do elevador1 |
|---|---|---|---|---|
| - | - | Andar 0 descendo [1,0,0] | Andar 0 descendo [1,0,0] | Led amarelo acionar conforme o pressionamento do botão. |
| Sensor 1 | Sensor 2 | Andar 0 descendo [1,0,0] | Andar 1 subindo [0,2,0] | Led amarelo acionar conforme o pressionamento do botão. |
| Sensor 1 | Sensor 2 | Andar 1 subindo [0,2,0] | Andar 2 subindo [0,0,2] | Led amarelo acionar conforme o pressionamento do botão. |
| Sensor 2 | Sensor 1 | Andar 1 subindo [0,2,0] | Andar 1 descendo [0,1,0] | Led não aciona |
| Sensor 2 | Sensor 1 | Andar 0 descendo [1,0,0] | Andar 0 descendo [0,1,0] | Led não aciona |

Fonte: Autor.

Na tabela 5 está o resultado da simulação de uma sequência de chamadas no andar 2, em que a última coluna representa a saída de modulo MC do elevador 1, onde na primeira linha ao ser pressionado o botão no módulo MA, como os dois elevadores estão no mesmo andar a preferência é para o elevador 1, por isso o led amarelo aciona. Já nas linhas 2 e 3 da tabela 5 o led amarelo foi acionando por conta da distância onde o elevador 1, sendo o mais próximo ao andar solicitado. Devido a manipulação do valor inicial das variáveis correspondente ao elevador 2, conseguimos fazer com que o sistema entendesse que elevador dois estava no andar 1 e subindo. Desta forma o Led amarelo fica apagado, pois o elevador solicitado selecionado foi o 2. Por fim, o último teste e o led não acionou novamente, por conta que o elevador 2 que foi chamado, devido o elevador 1 ainda vai no andar 0, para poder retornar ao andar 2 para o atendimento do chamado.

## 4.2 CUSTOS DO PROJETO

O projeto visou desenvolver um protótipo a baixo custo, onde os gastos praticamente foram os valores dos componentes utilizados. Todos os componentes foram adquiridos em Manaus. Além dos componentes comprados como mostra a tabela 6, foram utilizados alguns componentes que já tínhamos, como por exemplo fonte do celular e o cabo usb.

Tabela 6: Custos com o projeto

| Quant. | Componentes | Valor unitário (R$) | Valor total (R$) |
|---|---|---|---|
| 2 | ESP 32 | 69,00 | 138,00 |
| 1 | NodeMCU8266 | 49,90 | 49,90 |
| 1 | Display LCD 16x2 | 30,00 | 30,00 |
| 1 | Módulo rele de 4 canais | 30,00 | 30,00 |
| 2 | Modulo relé de 1 canal | 23,00 | 46,00 |
| 1 | Módulo I2C | 10,00 | 10,00 |
| 2 | Sensor de barreira arduino | 10,00 | 20,00 |
| 1 | Push botton | 2,00 | 2,00 |
| 2 | Protoboard 830 pontos | 12,50 | 25,00 |
| 1 | Protoboard 1660 pontos | 80,00 | 80,00 |
| | Total | | 430,90 |

Fonte: Autor

# 5 CONCLUSÃO

Por tudo que foi exposto ao longo deste trabalho, consolidou se como objetivo geral, a proposta da elaboração de um protótipo que monitore a posição das cabines de um elevador e controle as chamadas indesejadas delas. Visando desta forma a redução de utilização de energia elétrica e aumento de vida útil dos componentes devido desgaste por conta do uso inapropriado dos elevadores. Além do que, mostra que é possível um sistema funcionar via wireless, assim podendo despertar interesse no desenvolvimento novas pesquisas de comunicação entre objetos de um sistema através da internet sem interferência humana.

A comunicação dos módulos do protótipo, apesar de algumas instabilidades, comportou-se adequadamente para a aplicação deste projeto, uma vez que o mesmo não exige altas velocidades nas trocas de informações, sem esquecer que é utilizado um servidor gratuito. Vale ressaltar que essa falta de estabilidade não compromete o comportamento do elevador, pois o protótipo em questão apenas auxilia o sistema que já existe.

Com a utilização dos kits de prototipagem NodeMCU8266 e ESP32 é possível perceber que utilizamos o básico de suas funções e ainda se tem muito a ser explorado dessa tecnologia em trabalhos acadêmicos, pois durante a jornada de desenvolvimento deste trabalho verificou-se pouco conteúdo sobre a mesma e os que foram encontrados utilizam se apenas de suas funções básicas como a conectividade à internet.

Todos os módulos utilizaram o mesmo programa nas funções de conectividade tanto para internet quanto para a utilização do protocolo MQTT na comunicação aos tópicos no servidor. Foi desenvolvido um programa para cada módulo de acordo com a sua interação com o sistema já existente, onde o modulo elevador interage com a cabine para saber seu deslocamento, o modulo andar faz a interação com o usuário e o modulo de controle faz a interface do protótipo com o sistema do elevador. Desta forma, permitiu-se o desenvolvimento do protótipo sem fio, com isso facilitando instalação sem alterações físicas do elevador.

O protótipo foi desenvolvido baseado em um prédio de 3 andares, sendo possível alteração do programa para outros edifícios. Onde é necessário apenas o aumento do número de relés no módulo controlador.

## 5.1 DIFICULDADES ENCONTRADAS

As primeiras dificuldades que foram encontradas durante a execução do projeto foi entender a forma de comunicação dos módulos com a internet utilizando o protocolo MQTT, pois foi a primeira vez que tive contato com a tecnologia. A maioria dos projetos encontrados nesta pesquisa foram feitos no ambiente de programação Lua, sendo tal linguagem de programação não usual aos autores deste projeto.

A segunda dificuldade fez com que atrasasse a entrega do meu tcc em um ano. A Instabilidade de comunicação dos módulos à rede de internet, até então, era desconhecida. O projeto funcionava normalmente, mas de forma intermitente ele travava, sendo que a vezes retornava quando reiniciava o módulo. Foram realizados vários testes pois havia possibilidade de microcontrolador, revisado o programa e nada, até reinício automático coloquei no programa do mesmo. Porém o problema persistia, foi quando descobrir que se tratava da conexão à rede Wifi, pois local em que eu estava fazendo os testes ficava uma distância de uns 7 metros do roteador com três paredes como barreira em uma casa de laje de concreto. Após trabalhar próximo ao roteador sem barreiras obtive sucesso no funcionamento do protótipo.

## 5.2 TRABALHOS FUTUROS

Nesta seção são apresentadas propostas de melhorias para trabalhos futuros, visando a melhoria da funcionalidade do protótipo através da IOT.

O servidor eclipse de código aberto para testes funciona perfeitamente, porém para melhor desempenho do projeto faz-se necessário a criação do próprio broker utilizando outras tecnologias para melhor gerenciamento, como exemplo o raspberry Pi utilizando broker mosquitto.

Uma outra grande melhoria que o pode ser realizada no projeto é a implementação da visualização e realização de chamadas do elevador via telefone celular, assim deixando os usuários ainda mais a vontade.

# 6 REFERENCIAS BIBLIOGRAFICA

ALMEIDA, Gabriel. Embarcados: Configurando o ambiente de desenvolvimento do ESP32 no Windows, disponivel em:< https://embarcados.com.br/ambiente-esp32-no-windows/ > acessado em: 10 de jan 2019.

ANTONIOLLI, A. **Sistema de monitoramento automatizado para controle de qualidade de água em sistema aquapônico**. , 2019.

BALDO, G. L.; HIDRÁULICO, P. D. E. U. M. E. UNIVERSIDADE DO VALE DO RIO DOS SINOS-UNISINOS UNIDADE ACADÊMICA DE GRADUAÇÃO CURSO DE ENGENHARIA MECÂNICA. [s.d.].

BARROS, E.; CAVALCANTE, S. Introdução aos sistemas embarcados. **Artigo apresentado na Universidade Federal de Pernambuco-UFPE**, p. 36, 2010.

BRAGA, NEWTON C. Relé: Conceito e Aplicações. 1° ed. São Paulo. Brasil 2012.

CARDOSO NETO, C.; CARVALHO DE ALMEIDA, M. Ã.; GIL TEIXEIRA, V. C. REDES WIRELESS. **REVISTA DE TRABALHOS ACADÊMICOS-CAMPUS NITERÓI**, 2014.

CHASE, O.; ALMEIDA, F. Sistemas embarcados. **Mídia Eletrônica. Página na internet:< www. sbajovem. org/chase>, capturado em**, v. 10, n. 11, p. 13, 2007.

CUNHA, A. F. O que são sistemas embarcados. **Saber Eletrônica**, v. 43, n. 414, p. 1–6, 2007.

DA SILVA, R. O.; ARAUJO, W. M.; CAVALCANTE, M. M. Visao geral sobre microcontroladores e prototipagem com arduino. **Tecnologias Em Projeção**, v. 10, n. 1, p. 36–46, 2019.

D. B. Ansari, Atteeq-Ur-Rehman, and R. A. Mughal, “Internet of Things (IoT) protocols: A brief exploration of MQTT and CoAP,” International Journal of Computer Applications, vol. 179, no. 27, pp. 9–14, 2018

DE OLIVEIRA, S. **Internet das coisas com ESP8266, Arduino e Raspberry PI**. [s.l.] Novatec Editora, 2017.

Eclipse IoT, comunidade pertecente a fundação eclipse, disponivel em: <https://iot.eclipse.org/projects/sandboxes/> acesso 20 de jan 2022.

ESP8266EX Datasheet. 2018. Disponível em: <https://www.espressif.com/sites/default/files/documentation/0aesp8266ex_datasheet_en.pdf> acesso em: 10 de mar 2023.

FIRMINO, M. DE S.; MATEUS, W. DA R. F. Thymos pet–dosador alimentar automático para animais domésticos. Engenharia Elétrica-Tubarão, 2020.

GBK Robotcs, Disponível em :< https://gbkrobotics.wixsite.com/gbkrobotics/boards > Acesso dia: 10 fev 2018.

M. B. Abid, M. R. Rumon, T. Sraboni, R. Hossain, F. Ahmed, and J. Uddin, “Design and implementation of an enotice board using a nodemcu,” in Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 332, 2020. https://doi.org/10.1007/978-3-030-60036-5_21

LAPTRINHX, Disponível em: <https://laptrinhx.com/how-to-interface-pcf8574-i2c-lcd-with-esp8266-nodemcu-936725032/> Acesso dia: 15/10/2022

LIMA, Mayka Souza; NUNES, Maria Augusta Silveira Netto. Estudo Prospectivo Relativo ao Padrão ZigBee para Redes sem Fio e sua Utilização na Computação em Nuvens – Monografia (Mestrado em ciência da computação) - Universidade Federal de Sergipe, São Cristovão - SE, 2015, Disponível em: <https://scientiaplena.emnuvens.com. br/sp/article/view/2066/1105>. Acesso em: 20 abr. 2018.

NENOKI, Eduardo. ZIGBEE – Estudo da tecnologia e aplicação no sistema elétrico de potência. 2013. 50 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

M. Saikrishna and G. Vijaykiran, "IOT Based Home Electrical Appliances Control Using Node MCU", *International Journal of Scientific Engineering and Technology Research*, vol. 06, no. 04, pp. 0783-0788, February 2017, ISSN 2319-8885.

PARIHAR, Y. S. Internet of things and nodemcu. journal of emerging technologies and innovative research, v. 6, n. 6, p. 1085, 2019.

PAULA, P. M. DE. **Protótipo de elevador didático utilizando controlador programável com PIC16F877A**. Universidade Tecnológica Federal do Paraná, , 2014.

ROBOCORE, Disponivel: https://www.robocore.net/sensor-robo/sensor-de-obstaculo, com Acesso: 12 ago 2018.

RICARDO JUNIOR, O. Sistema de monitoramento residencial baseado em Internet das Coisas. Monografia.Universidade Estadual de Londrina. 2017.

SANTOS, B. P. et al. Internet das coisas: da teoria à prática. **Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos**, v. 31, 2016.

S. Lee, H. Kim, D.-k. Hong and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level", *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2013.

THONSEN, Adilson. site Marke Hero. Disponível em: < https://www.makerhero.com /blog/programar-nodemcu-com-ide-arduino/>. Acesso em: 20 abril. 2020/FilipeFlop agora é MakerHero - Referência em Eletrônica

WANZELER, T.; FULBER, H.; MERLIN, B. Desenvolvimento de um sistema de automação residencial de baixo custo aliado ao conceito de Internet das Coisas (IoT). **XXXIV Simpósio Brasileiro de Telecomunicações. Santarém, PA**, p. 40–44, 2016.

## APÊNDICE A- CÓDIGO DO MÓDULO ELEVADOR - ME

```
#include<stdio.h>
#include<string.h>
#include<PubSubClient.h>
#include<WiFi.h>
#define max 4

//WiFi
const char* SSID = "Martins_AMCNetwork";   // SSID / nome da rede WiFi que deseja se conectar
const char* PASSWORD = "1234567";   // Senha da rede WiFi que deseja se conectar
WiFiClient wifiClient;

//MQTT Server
const char* BROKER_MQTT = "mqtt.eclipseprojects.io"; //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883;                      // Porta do Broker MQTT

#define ID_MQTT  "Elevador01"             //Informe um ID único e seu. Caso sejam usados IDs repetidos a ultima conexão irá sobrepor a anterior.
#define TOPIC_PUBLISH "Loc_Ele01"      //Informe um Tópico único. Caso sejam usados tópicos em duplicidade, o último irá eliminar o anterior.
#define TOPIC_PUBLISH1 "chama"
#define TOPIC_SUBSCRIBE "chama"
PubSubClient MQTT(wifiClient);        // Instancia o Cliente MQTT passando o objeto espClient

void subindo();
void descendo();
void mantemConexoes();  //Garante que as conexões com WiFi e MQTT Broker se mantenham ativas
void conectaWiFi();     //Faz conexão com WiFi
void conectaMQTT();     //Faz conexão com Broker MQTT
```

```
void zeramento();
void recebePacote(char* topic, byte* payload, unsigned int length);
void sense();
void MostraDados();

//void enviaValores();

char ch[max]={"000"};
int andar=0,ich=-1;
char elevador[max];

const long intervaloEnvio = 1000;
unsigned long ultimoEnvio = 0;

const int sensor1 = 4;     //
const int sensor2 = 5;      // the number of the LED pin
int s1=0,s2=0,dir=0;
//---------------------------------------------------------------------------------------------
void setup() {
elevador[0] = '1';
elevador[1] = '0';
elevador[2] = '0';

Serial.begin(9600);
delay(10);
  // initialize the sensor1 pin as an output:
  pinMode(sensor1 , INPUT);
  // initialize the sensor2 pin as an input:
  pinMode(sensor2, INPUT);

  conectaWiFi();
  MQTT.setServer(BROKER_MQTT, BROKER_PORT);
```

```
  MQTT.setCallback(recebePacote);
}
//--------------------------------------------------------------------------------------------
void loop() {




    //   Serial.println(elevador);
    //   Serial.print("andar:....");
    //   Serial.println(andar);
       mantemConexoes();
       sense();

       unsigned long tempoAtual = millis();

       if (tempoAtual - ultimoEnvio >= intervaloEnvio)
          {
           ultimoEnvio = tempoAtual;   //Armazena o valor da última vez do envio
           MostraDados();
          Serial.print("vetor posição elevador: ");
          Serial.println(elevador);
          Serial.print("andar do elevador: ");
          Serial.println(andar);
          }

       zeramento();
       MQTT.loop();
     }
```

---

```
        void conectaWiFi() {

          if (WiFi.status() == WL_CONNECTED) {
```

```
    return;
  }

  Serial.print("Conectando-se na rede: ");
  Serial.print(SSID);
  Serial.println("  Aguarde!");

  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI
  while (WiFi.status() != WL_CONNECTED) {
     delay(100);
     Serial.print(".");
  }
  //Serial.println();
  Serial.print("Conectado com sucesso, na rede: ");
  Serial.print(SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());
}
```

---

```
void conectaMQTT() {
   while (!MQTT.connected()) {
      Serial.print("Conectando ao Broker MQTT: ");
      Serial.println(BROKER_MQTT);
      if (MQTT.connect(ID_MQTT)) {
         Serial.println("Conectado ao Broker com sucesso!");
          MQTT.subscribe(TOPIC_SUBSCRIBE);
      }
      else {
         Serial.println("Noo foi possivel se conectar ao broker.");
         Serial.println("Nova tentatica de conexao em 10s");
         delay(10000);
      }
```

```
    }
}
```

```
void mantemConexoes() {
    if (!MQTT.connected()) {
       conectaMQTT();
    }
    conectaWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}
```

```
void sense(){
  s1=digitalRead(sensor1);
  s2=digitalRead(sensor2);
 if(s1){
              subindo();
              dir=2;
               while(!digitalRead(sensor2)){
                   s1=0;
                   s2=0;
               }
               while(digitalRead(sensor2));
              }

           else if(s2){
                descendo();
                dir=1;
                while(!digitalRead(sensor1)){
                s1=0;
                s2=0;
                }
               while(digitalRead(sensor1));
               }
```

```
        }
void subindo(){
  if(andar>=max){
      Serial.println("Pilha cheia\n");
       return;
   }
   andar++;
 }
void descendo(){
     andar--;
     if (andar<0){
           andar=0;
           return;
    }
}
void MostraDados(){
      if(dir==1){
        strcpy(elevador,"000");
         elevador[andar]='1';
      }
      else if(dir==2){
         strcpy(elevador,"000");
         elevador[andar]='2';
      }
       Serial.print("..elevador ");
       Serial.println(elevador);
       Serial.print("andar.:");
       Serial.println(andar);
       delay(100);
       MQTT.publish(TOPIC_PUBLISH, elevador);
       Serial.println("Elevador em movimento. Payload enviado publish.");
      delay(10);
```

```
      Serial.println(elevador);
}
```

```
void recebePacote(char* topic, byte* payload, unsigned int length){
Serial.println("elev no receb ini: --------");
//Serial.println("receber pacotes");
   String msg;
   //obtem a string do payload recebido
   for(int i = 0; i < length; i++)
     {
        char c = (char)payload[i];
        msg += c;
        Serial.print("mensagem recebida vet chama elevador1: ");
        Serial.println(msg[i]);
     }
   for(int i = 0; i < length; i++)
     {
       ch[i]=msg[i];
    //    if(ch[i]=='1'||ch[i]=='2')ich=i;
     }
Serial.print("elev no receb: ");
Serial.println("recolhendo chamada elev1 ");
Serial.print("vet ch: ");
Serial.println(ch);
}
```

```
void zeramento(){
  if(ch[andar]=!'0'){
       ch[andar]='0';

       MQTT.publish(TOPIC_PUBLISH1,ch);
       Serial.println("publicando publish1..");
       Serial.println(ch);
       //ich=-1;
```

```
    }
}
```

## APÊNDICE B- CÓDIGO DO MÓDULO ANDAR - MA

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// DEFINIÇÕES
#define endereco  0x27 // Endereços comuns: 0x27, 0x3F
#define colunas   16
#define linhas    2
#define max 4
#define button 5
#define x 2 // andar 02
#define ID_MQTT                 "Andar02"     //Informe um ID único e seu. Caso sejam usados IDs repetidos a ultima conexão irá sobrepor a anterior.
#define TOPIC_SUBSCRIBE         "Loc_Ele01"   //Informe um Tópico único. Caso sejam usados tópicos em duplicidade, o último irá eliminar o anterior.
#define TOPIC_SUBSCRIBE1        "chama"
#define TOPIC_SUBSCRIBE2        "Loc_ele02"
#define TOPIC_SUBSCRIBE3        "chama02"

#define TOPIC_PUBLISH           "chama"
#define TOPIC_PUBLISH1          "chama02"

//-------Instancias----------------------------------------------------------------------------------------
-
LiquidCrystal_I2C lcd(endereco, colunas, linhas);//definição do andar do módulo;
WiFiClient wifiClient;
PubSubClient MQTT(wifiClient);        // Instancia o Cliente MQTT passando o objeto espClient

//-----------------------------------------------------------------------------------------------------------
char chamada[max]={"000"};
```

```
char chamada2[max]={"000"};


int ch1=0,ch2=5,pos1,pos2=4,dir1,dir2=2,soltar=0;


const char* SSID = "Martins_AMCNetwork";   // SSID / nome da rede WiFi que deseja se conectar
const char* PASSWORD = "1234567";          // Senha da rede WiFi que deseja se conectar


//MQTT Server


const char* BROKER_MQTT = "mqtt.eclipseprojects.io";  //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883;                              // Porta do Broker MQTT


bool flag=0;                 // se botao ta solto ==1


//-----------------------------------------------------------------------------------------------------------
//declarando times de acesso


const long intervaloRecebido =  100;
const long intervaloRecebido1 = 150;
const long intervaloRecebido2 = 200;
const long intervaloRecebido3 = 250;
const long intervaloRecebido4 = 1000;


unsigned long ultimorecebido = 0;
unsigned long ultimorecebido1 = 0;
unsigned long ultimorecebido2 = 0;
unsigned long ultimorecebido3 = 0;
unsigned long ultimorecebido4 = 0;
//---------------------------------------
```

```
//Declaração das Funções
void mantemConexoes();  //Garante que as conexoes com WiFi e MQTT Broker se mantenham ativas
void conectaWiFi();     //Faz conexão com WiFi
void conectaMQTT();     //Faz conexão com Broker MQTT
void lcd_start();
void but();
void Imprimir_Display();
void recebePacote(char* topic, byte* payload, unsigned int length);
//-----------------------------------------------------------------------------------------------------------
void setup() {
  pinMode(button, INPUT_PULLUP); //Define Botao como entrada
  lcd_start();                   //Inicia LCD

  Serial.begin(115200);

  conectaWiFi();

  MQTT.setServer(BROKER_MQTT, BROKER_PORT);
  MQTT.setCallback(recebePacote);
}

void loop() {
   mantemConexoes();
   MQTT.loop();
   unsigned long tempoAtual = millis();
   if (tempoAtual - ultimorecebido >= intervaloRecebido)
         {
          ultimorecebido = tempoAtual;                  //Armazena o valor da última vez do envio
          MQTT.subscribe(TOPIC_SUBSCRIBE);              //----colhendo informaçoes de posição do elevador 1
         }
```

```
  if (tempoAtual - ultimorecebido1 >= intervaloRecebido1)
        {
         ultimorecebido1 = tempoAtual;              //Armazena o valor da última vez do
envio
         MQTT.subscribe(TOPIC_SUBSCRIBE1);          //----colhendo informaçoes de
chamada do elevador 1
        }

  if (tempoAtual - ultimorecebido2 >= intervaloRecebido2)
        {
         ultimorecebido2 = tempoAtual;              //Armazena o valor da última vez do
envio
         MQTT.subscribe(TOPIC_SUBSCRIBE2);          //----colhendo informaçoes de
posição do elevador 2
        }

  if (tempoAtual - ultimorecebido3 >= intervaloRecebido3)
        {
         ultimorecebido3 = tempoAtual;              //Armazena o valor da última vez do
envio
         MQTT.subscribe(TOPIC_SUBSCRIBE3);          //----colhendo informações de
chamada do elevador 1
        }



  but();

  if (tempoAtual - ultimorecebido4 >= intervaloRecebido4)
        {
         ultimorecebido4 = tempoAtual;              //Armazena o valor da última vez do
envio
         Imprimir_Display();
        }
```

```
 }
void conectaWiFi() {
  if (WiFi.status() == WL_CONNECTED) {
     return;
  }
  Serial.print("Conectando-se na rede: ");
  Serial.print(SSID);
  Serial.println("  Aguarde!");

  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI
  while (WiFi.status() != WL_CONNECTED) {
     delay(100);
     Serial.print(".");
  }
  Serial.println();
  Serial.print("Conectado com sucesso, na rede: ");
  Serial.print(SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());
}
void conectaMQTT() {
    while (!MQTT.connected()) {
        lcd.clear(); // LIMPA DISPLAY
        Serial.print("Conectando ao Broker MQTT: ");
        lcd.print("conectando-se ao Broke...");
        Serial.println(BROKER_MQTT);
        lcd.setCursor(0, 1);
        lcd.print(BROKER_MQTT);
        if (MQTT.connect(ID_MQTT)) {
            lcd.clear(); // LIMPA DISPLAY
            lcd.print("conectado com sucesso");
```

```
            Serial.println("Conectado ao Broker com sucesso!");
            Serial.print("controle1 dentro MQTT: ");
        }
        else {
            Serial.println("Nao foi possivel se conectar ao broker.");
            Serial.println("Nova tentatica de conexao em 10s");
            lcd.clear(); // LIMPA DISPLAY
            lcd.print("nao foi possivel conectar");
            delay(5000);
            lcd.clear(); // LIMPA DISPLAY
            lcd.print("nova tentativa em 5s");
        delay(5000);
        }
    }
}
```

```
void mantemConexoes() {
    if (!MQTT.connected()) {
       conectaMQTT();
       }
    conectaWiFi();       //se não há conexão com o WiFI, a conexão é refeita
}
```

```
void recebePacote(char* topic, byte* payload, unsigned int length){

    Serial.println("receber pacotes");
    String msg;

    //obtem a string do payload recebido
    for(int i = 0; i < length; i++)
       {
          char c = (char)payload[i];
          msg += c;
       }
```

```
//-------------------------captura informação de direção e posição que o elevador 1 se encontra----------------------------------------
  if (String(topic).equals("Loc_Ele01")){
    Serial.println("recebendo pacotes elevador");

        for(int a = 0; a < length; a++){

           if(msg[a]=='1'){
               dir1=1;
               pos1=a;
              }
          else if(msg[a]=='2'){
                dir1=2;
                pos1=a;
               }
           }
  }
//-------------------------------------------------------------------------------------------

//-----------------------------pega o vetor de chamada elevador1-----------------------------
          else if(String(topic).equals("chama")){
              for(int a=0;a<max;a++){
                 chamada[a]=msg[a];
                 if (chamada[a]=='1'||chamada[a]=='2')
                    ch1=a;
                 }
            }
//-------------------------pega o vetor de posição elevador 2----------------------------
            else if(String(topic).equals("Loc_ele02")){
               for(int a = 0; a < length; a++){
                    if (msg[a]=='1'){
                    pos2=1;
```

```
                    dir2=a;
                  }
                 else if (msg[a]=='2'){
                        pos2=2;
                        dir2=a;
                     }
                 }
              }
//-------------------------pega o vetor chamada do elevador2-----------------------------
          else if (String(topic).equals("chama02")){
                 for(int a=0;a<max;a++){
                    chamada2[a]=msg[a];
                    if (chamada2[a]=='1'||chamada[a]=='2')
                       ch2=a;
                  }
              }
//-----------------------------------------------------------------------------------------
}
```

```
void lcd_start(){
  lcd.init();         // INICIA A COMUNICAÇÃO COM O DISPLAY
  lcd.backlight();    // LIGA A ILUMINAÇÃO DO DISPLAY
  lcd.clear();        // LIMPA DISPLAY
}
```

```
void but(){
int cont=0,cont2=0,i,j;
  if(digitalRead(button)==0 && flag==0){
        flag=1;
        i=pos1;

        if (dir1==1){
             while(i=!x && i>0){
                cont++;
```

```
        i--;
          }

     while(i=!x){
        cont++;
        i++;
          }
  }

else {

    if(ch1>x) j=ch1;
    else j=x;

    while(i=!x && i<j){
       cont++;
       i++;
         }

    while(i=!x){
       cont++;
       i--;
         }
    }

i=pos2;
if (dir2==1){
      while(i=!x && i>0){
         cont2++;
         i--;
           }
      while(i=!x){
```

```
        cont2++;
        i++;
          }
  }
else {

   if(ch2>x) j=ch2;
   else j=x;

   while(i=!x && i<j){
      cont2++;
      i++;
        }

   while(i=!x){
      cont2++;
      i--;
        }
    }

 if(cont<=cont2){
  soltar=1;
  chamada[x]='1';
  MQTT.publish(TOPIC_PUBLISH, chamada);
  lcd.setCursor(0, 5);
  lcd.print("chamada E1");
   }
else{
    soltar=2;
    chamada2[x]='1';
    MQTT.publish(TOPIC_PUBLISH1, chamada2);
    lcd.setCursor(0, 5);
```

```
            lcd.print("chamada E2");
           }
  }
 if(digitalRead(button)==1 && flag==1){
       flag=0;
       if(soltar==1){
            chamada[x]='2';
            MQTT.publish(TOPIC_PUBLISH, chamada);
            lcd.clear();
         }
       else if(soltar==2){
                chamada2[x]='2';
                MQTT.publish(TOPIC_PUBLISH1, chamada2);
                lcd.clear();
           }
   }
}
```

---

```
void Imprimir_Display(){

if(dir1== 2){
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.print("ANDAR: ");
  lcd.print(pos1);
  lcd.print(" subindo");
  lcd.write(byte(20));
}
 if(dir1== 1){
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.print("ANDAR: ");
  lcd.print(pos1);
```

```
    lcd.print(" Descendo");
    lcd.write(byte(20));
   }
}
```

## APÊNDICE C- CÓDIGO DO MÓDULO CONTROLADOR - MC

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>

#define rele1 5 //D1
#define rele2 4 //D2
#define rele3 2 //D4

//WiFi
const char* SSID = "Martins_AMCNetwork";   // SSID / nome da rede WiFi que deseja se conectar
const char* PASSWORD = "FM20142019";    // Senha da rede WiFi que deseja se conectar
WiFiClient wifiClient;

//MQTT Server
const char* BROKER_MQTT = "mqtt.eclipseprojects.io"; //URL do broker MQTT que se deseja utilizar
int BROKER_PORT = 1883;

#define ID_MQTT  "cont_Ele01"             //Informe um ID unico e seu. Caso sejam usados IDs repetidos a ultima conexão irá sobrepor a anterior.
#define TOPIC_SUBSCRIBE "chama"     //Informe um Tópico único. Caso sejam usados tópicos em duplicidade, o último irá eliminar o anterior.
PubSubClient MQTT(wifiClient);        // Instancia o Cliente MQTT passando o objeto espClient

//Declaração das Funções
void mantemConexoes();  //Garante que as conexoes com WiFi e MQTT Broker se mantenham ativas
void conectaWiFi();     //Faz conexão com WiFi
void conectaMQTT();     //Faz conexão com Broker MQTT
void recebePacote(char* topic, byte* payload, unsigned int length);
```

```
void setup() {
  pinMode(rele1, OUTPUT);
  pinMode(rele2, OUTPUT);
  pinMode(rele3, OUTPUT);

digitalWrite(rele1, HIGH);
digitalWrite(rele2, HIGH);
digitalWrite(rele3, HIGH);
  Serial.begin(115200);
  conectaWiFi();
  MQTT.setServer(BROKER_MQTT, BROKER_PORT);
  MQTT.setCallback(recebePacote);
}
void loop() {
 mantemConexoes();
 MQTT.loop();
}
void recebePacote(char* topic, byte* payload, unsigned int length)
{
   Serial.println("receber pacotes");
    String msg;
    //obtem a string do payload recebido
    for(int i = 0; i < length; i++)
    {
       char c = (char)payload[i];
       msg += c;
    }

  for(int i = 0; i < length; i++){
    if (i==0){
```

```
      if(msg[i]=='1')
      digitalWrite(rele1,LOW);
      else digitalWrite(rele1,HIGH);
   }
   else if (i==1){
    if(msg[i]=='1')digitalWrite(rele2,LOW);
      else digitalWrite(rele2,HIGH);
              }
        else if (i==2){
              if(msg[i]=='1')digitalWrite(rele3,LOW);
              else digitalWrite(rele3,HIGH);
              }
     }
Serial.println(msg);
}
```

---

```
void conectaWiFi() {

 if (WiFi.status() == WL_CONNECTED) {
   return;
 }

 Serial.print("Conectando-se na rede: ");
 Serial.print(SSID);
 Serial.println("  Aguarde!");

 WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI
 while (WiFi.status() != WL_CONNECTED) {
   delay(100);
   Serial.print(".");
 }
 Serial.println();
 Serial.print("Conectado com sucesso, na rede: ");
```

```
  Serial.print(SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());
}
```

```
void conectaMQTT() {
    while (!MQTT.connected()) {
        Serial.print("Conectando ao Broker MQTT: ");
        Serial.println(BROKER_MQTT);
        if (MQTT.connect(ID_MQTT)) {
            Serial.println("Conectado ao Broker com sucesso!");
             MQTT.subscribe(TOPIC_SUBSCRIBE);
        }
        else {
            Serial.println("Noo foi possivel se conectar ao broker.");
            Serial.println("Nova tentatica de conexao em 10s");
            delay(10000);
        }
    }
}
```

```
void mantemConexoes() {
    if (!MQTT.connected()) {
       conectaMQTT();
    }
    conectaWiFi(); //se não há conexão com o WiFI, a conexão é refeita
}.
```