UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Fernando Ferreira

SISTEMA INTELIGENTE DE GERENCIAMENTO E CONTROLE DE VAGAS DE ESTACIONAMENTO

Santa Maria, RS
2021

CT/UFSM,RS	FERREIRA, Fernando	Engenheiro	2021

Fernando Ferreira

SISTEMA INTELIGENTE DE GERENCIAMENTO E CONTROLE DE VAGAS DE ESTACIONAMENTO

Trabalho de Conclusão de Curso apresentada ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção de **Bacharel em Engenharia de Controle e automação**.

Orientador: Prof. Dr. Frederico Menine Schaf

Santa Maria, RS

2021

Fernando Ferreira

SISTEMA INTELIGENTE DE GERENCIAMENTO E CONTROLE DE VAGAS DE ESTACIONAMENTO

Trabalho de Conclusão de Curso apresentada ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção de **Bacharel em Engenharia de Controle e automação**.

Aprovado em 06 de 2021 de 2021:

Frederico Menine Schaf, Dr.
(Orientador)

Anselmo Rafael Cukla, Dr (UFSM)
(Presidente)

Sérgio Luis Sardi Mergen, Dr (UFSM)

Santa Maria, RS

2021

DEDICATÓRIA

Dedico este trabalho aos meus pais, minha irmã, minha namorada e meus amigos!

AGRADECIMENTOS

Agradeço inicialmente à minha família, aos meus pais, Osmar e Gerda, e a minha irmã Caroline, por todo o amor e suporte. Agradeço pela educação que eles me deram ao longo da vida, e pelo apoio para seguir em busca de meus próprios sonhos e pela confiança nos projetos que me comprometi. Agradeço também a minha namorada Angélica por estar sempre ao meu lado durante esses anos.

Agradeço ao meu orientador Prof. Frederico Menine Schaf e ao Professor Claiton Moro Franchi por terem acreditado no meu trabalho, pela paciência e a ajuda ao longo destes semestres.

Agradeço aos meus amigos que sempre me incentivaram e deram suporte, aos colegas e amigos que conheci durante o curso, sem nossa amizade não teria chego aonde cheguei. Agradeço aos meus amigos Rodrigo e Marcelo que sempre estiveram comigo nas horas difíceis, ao Walther e o Jeann por me ajudarem a tornar esse TCC realidade. Minha eterna gratidão, esse TCC também é de vocês.

Agradeço também à Universidade Federal de Santa Maria, pela excelente qualidade no ensino e ao curso de Engenharia de Controle e Automação por todos os conhecimentos que adquiri ao longo da graduação.

E a todos que diretamente ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“O conhecimento serve para encantar as pessoas. Não para humilhá-las”

(MÁRIO SÉRGIO CORTELLA)

RESUMO

SISTEMA INTELIGENTE DE GERENCIAMENTO E CONTROLE DE VAGAS DE ESTACIONAMENTO

AUTOR: FERNANDO FERREIRA
ORIENTADOR: FREDERICO MENINE SCHAF

Nas grandes cidades cada vez mais se tem a presença de veículos pessoais utilizados como meio de transporte para os ambientes de trabalho e estudo, assim como para lazer. Com isso, o número de estacionamentos privativos que oferecem segurança aos veículos tem aumentado. Porém, o desperdício de tempo para que se encontre uma vaga de estacionamento, assim como a alocação de pessoal para auxílio dos motoristas por parte dos estabelecimentos, acarretam custos desnecessários de ambos os lados. Os sistemas atuais implementados utilizam em sua maioria tickets como ferramenta para controle do número de vagas disponíveis, onde o responsável do estacionamento consegue definir quantos veículos existem no seu estabelecimento, porém não é possível definir qual a sua localização dentro do mesmo. Existe assim a necessidade de um sistema robusto, onde seja possível informar tanto ao usuário como o responsável pelo estabelecimento, o número e a localização das vagas disponíveis dentro do estacionamento. Por isso, esse trabalho propõe um sistema inteligente que direciona os motoristas para vagas desocupadas através de sinalizadores na parte superior das vagas. Para o desenvolvimento deste projeto, foram utilizados dos mais variados conhecimentos adquiridos durante a graduação, e através destes foi optado pela utilização de um micro controlador chamado ESP-32, sensor de distância ultrassônico HC-SR04 Node-RED uma ferramenta de desenvolvimento baseada em fluxo para programação visual e o auxílio de um banco de dados para armazenamento dos dados deste trabalho. Portanto, o intuito desse trabalho é contribuir para o melhor funcionamento dos sistemas de distribuição de vagas de estacionamento através de um sistema de gerenciamento de vagas inteligente, além disso contar com um sistema supervisório que possa auxiliar numa melhor visualização da ocupação das vagas pelos motoristas, diminuindo assim o tempo de procura por uma vaga de estacionamento livre. O sistema obteve o resultado esperado, tanto no projeto de *hardware* quanto no *software*. Este sistema esta preparado para futuras melhorias, como maior alcance da rede através da implementação da rede *mesh* nos dispositivos ESP-32 e também na implementação de placas indicadoras de vagas desocupadas para os motoristas.

Palavras-chave: Estacionamento. Gerenciamento de Vagas.

ABSTRACT

INTELLIGENT PARKING SPACE MANAGEMENT AND CONTROL SYSTEM

AUTHOR: FERNANDO FERREIRA
ADVISOR: FREDERICO MENINE SCHAF

In large cities, there is na increasing presencence of personal vehicles being used as means of transportation for work and study places, as well as for leisure. As a result, the number of private car parks that provide security for vehicles has increased. However, the time wasted in order to find a parking place, as well as the allocation of personnel to assist drivers by the estabilishments, entails unnecessary costs for both sides. The current systems implemented mostly use tickets as a tool to control the number of available spaces, in wich the person in charge of the parking lot manages to define how many vehicles are there at his estabilishment, but it is not possibible to define it's location within it. Thus, there is a need for a robust system, in wich is possible to inform both the user and the person in charge of the estabilishment, the number and location of available spaces within the parking lot. Therefore, this work proposes an intelligent system that orientate drivers to vacant parking spots, through signalers at the top of the parking spaces. For the development of his project, it was used the most varied knowledge acquired during graduation, and through these it was decided to use a micro controller called ESP-32, ultrasonic distance sensor HC-SR04 Node-RED, a flow-based development tool for visual programming and the aid of a database for storing the data of this work. Therefore, the aim of this work is to contribute to the better functioning parking spaces distribution systems through an intelligent parking management system, in addition to having a supervisory system that can assist in a better visualization of the parking spaces occupation by drivers, thus decreasing the search time for na available parking space. The system obtained the expected result, both in the hardware project and in the software. This system is prepared for future improvements, such as greater reach of the network through the implementation of network mesh in ESP-32 devices and also the implementation of signs indicating unoccupied spaces for drivers.

Keywords: Parking. Content.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IP	*Internet Protocol*
TCP	*Transmission Control Protocol*
CPU	*Unidade central de processamento*
RAM	*Random Access Memory*
Wi-Fi	*Wireless Fidelity*
USB	*Universal Serial Bus*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IBM	*International Business Machines Corporation*
IEEE	*Instituto de Engenheiros Eletricistas e Eletrônicos*
SGBD	*Sistema Gerenciador de Banco de Dados*
GUI	*Graphical User Interface*
PHP	*Hyper-text Preprocessor*

SUMÁRIO

1 INTRODUÇÃO

A missão da busca por uma vaga para estacionar seu carro é uma tarefa do cotidiano que frequentemente incomoda. São muitas empresas que não possuem uma gestão para melhorar sua rotina, seja operacional ou até mesmo para seu estacionamento. Esta má gestão faz com que muitas vezes as organizações percam clientes ou tomem atitudes emergenciais como a alocação de pessoal de outras áreas para orientar os motoristas, seja para tranquilizá-los tanto para ajudá-los na localização de uma vaga para seu respectivo veículo.

As vias estreitas e a alta densidade de tráfego de veículos, especialmente nos centros das cidades, dificultam a condução e afetam a busca diária por um espaço de estacionamento. Isso custa tempo, dinheiro e também gera um desconforto ao motorista. Neste cenário, torna-se cada vez mais importante a necessidade do auxílio de sistemas automatizados, para facilitar situações do dia-a-dia como a citada acima.

Com o desenvolvimento deste projeto pode-se resolver os problemas levantados, sendo o sistema instalado na parte superior onde mantém-se posicionado o veículo, desde que a vaga de estacionamento seja em ambiente fechado. Para que o sistema atue são necessárias uma conexão à rede *Wi-Fi* e uma tomada.

Este projeto tem traçado alguns objetivos de desenvolvimento, tais como: o desenvolvimento de uma *dashboard* de gerenciamento do sistema de vagas de estacionamento, *dashboard* com informações relevantes sobre o sistema, contando com a lotação em tempo real, dispositivos instalados, sua localização e um sistema supervisório em tempo real do estacionamento. Além do *dashboard*, o projeto tem como objetivo ser compacto e de fácil instalação, e que as informações do sistema como um todo possam ser registradas em um banco de dados online, priorizando a estabilidade e segurança do sistema como um todo. Todo este projeto demandou de conhecimentos obtidos durante o curso de Engenharia para integração de *hardware* e *software*.

2 REVISÃO DA LITERATURA

Neste capítulo, será abordada uma revisão da literatura dos principais conceitos utilizados no desenvolvimento do Trabalho de conclusão de curso.

2.1 TRABALHOS RELACIONADOS

No meio acadêmico existem alguns trabalhos semelhantes a este, como o trabalho de Bandeira (2015), o qual utiliza o Arduino como controlador e sensores ultrassônicos. Diferente do trabalho do Bandeira, Victor (2019) utilizou outros componentes eletrônicos para a mesma finalidade, utilizou como controlador Raspberry Pi 3 Model B e resistores dependentes de luz LDR. Além destes trabalhos acadêmicos existem no mercado empresas que fornecem serviços semelhantes a este trabalho, onde as empresas utilizam sensores tanto na posição superior à vaga, quanto no piso da vaga de estacionamento. As principais empresas que trabalham com estes sistemas no mercado brasileiro são a AutomSolution, MFpark Estacionamentos e Technik. Estas empresas fornecem para seus clientes os indicadores de ocupação, como os da Figura 2.1, *dashboard* com as principais informações do estacionamento em tempo real. Similar à este projeto as empresas utilizam para detecção dos veículos sensores ultrassônicos, com emissor e receptor de sinal, ou utilizam sensores magnéticos.

Figura 2.1: Indicadores de ocupação de vaga da empresa AutomSolution

Fonte: AutomSolution.

2.2 SENSOR DE DISTÂNCIA ULTRASSÔNICO HC-SR04

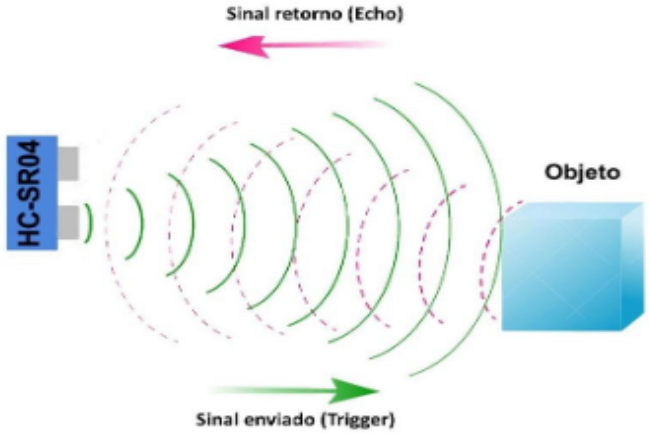
O Sensor Ultrassônico HC-SR04 (vide Figura 2.2) é um sensor amplamente utilizado para mapeamentos e localização de objetos em um ambiente. O sensor emite ondas sonoras

de alta frequência através do seu transdutor piezoelétrico e após detecta os pulsos de retorno através de outro transdutor, (vide Figura 2.3). Em seguida, o sensor converte estes dados em variação de tensão proporcional a distância (ADARSH et al., 2016).

Figura 2.2: Sensor de Distância Ultrassônico HC-SR04

Fonte: Retirado de FILIPEFLOP (2011).

Figura 2.3: Sinal do Retorno



Fonte: Retirado de FILIPEFLOP (2011).

O funcionamento consiste em duas etapas após o sinal de *output* estar no nível alto:

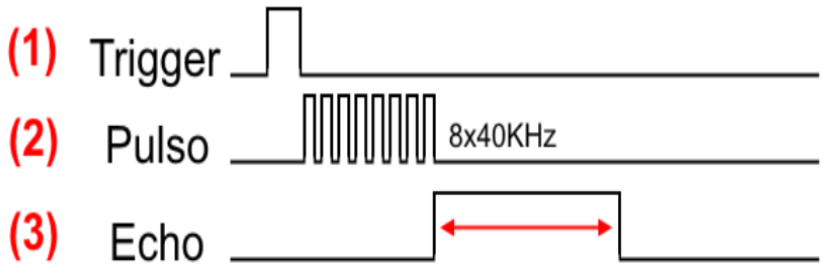
- Ao receber o sinal de *trigger*, o sensor envía 8 pulsos de 40 kHz e detecta se há algum sinal de retorno ou não.
- Se algum sinal de retorno for identificado pelo receptor, o sensor gera um sinal de nível alto no pino de saída cujo tempo de duração é igual ao tempo calculado entre o envio e o retorno do sinal ultrassônico.

Por meio desse pulso de saída, calcula-se que o tempo é igual a duração entre emissão e recepção, a distância entre o sensor e o objeto/obstáculo, em que:

- O circuito externo (microcontrolador) envia um pulso de *trigger* de pelo menos 10s em nível alto
- Velocidade do som = 340 m/s

Percebe-se que as unidades devem estar coerentes para o resultado da conta ser correto. Assim, se a velocidade do som é dada em metros/segundos, o tempo de duração do sinal de saída deve estar em segundos para que possa-se encontrar a distância em metros. Conforme Figura 2.4, pode-se verificar o funcionamento do sensor, onde o sinal do *trigger* em nível alto, os 8 sinais de pulso com a frequência de 40 kHz e o *echo* identificando a distância entre os pulsos.

Figura 2.4: Diagrama de tempo Sensor Ultrassônico HC-SR04



Fonte: Retirado de FILIPEFLOP (2011).

2.3 DIODO EMISSOR DE LUZ

Os Diodos Emissores de Luz (LEDs) são dispositivos semicondutores que surgiram por volta da década de 60. LEDs são muito similares aos diodos semicondutores tradicionais, permitindo a passagem de corrente em apenas um sentido. Quando polarizado desta forma resulta na emissão de luz, gerando assim seu nome Diodo Emissor de Luz. Os LEDs podem ser classificados em três categorias, conforme Figura 2.5: de sinalização, de alto brilho e de potência (BULLOUGH, 2003).

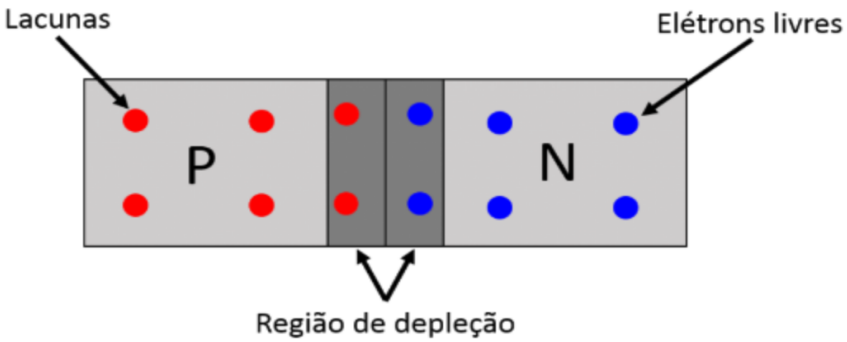
O funcionamento do LED ocorre pela sua composição, que é composta por dois materiais distintos formando uma junção P-N, conforme a Figura 2.6. Do lado P da junção existem pequenas lacunas que são resultantes da falta de elétrons, enquanto do lado N há o excesso

Figura 2.5: LEDs de sinalização, alto brilho e potência

Fonte: PINTO et al. (2012). .

de elétrons. Quando ambos são polarizados diretamente, as lacunas e os elétrons se atraem ao mesmo ponto, e com a combinação entre esses elementos resulta a emissão de fótons, transformando energia elétrica em luz (BULLOUGH, 2003).

Figura 2.6: Junção P-N



Fonte: UFRGS (2019).

2.4 MICROCONTROLADORES

De forma direta, pode-se dizer que o microcontrolador é um dispositivo que mistura *hardware* com *software*. Em um microcontrolador pode-se resolver problemas através de lógica de programação, conseguindo controlar dispositivos para fazer determinadas funções de uma maneira simples, fácil, flexível e poderosa, como é o caso deste trabalho. Numa abordagem mais técnica, pode-se dizer que um microcontrolador é um circuito integrado, assim como um microprocessador (BRASIL, 2014).

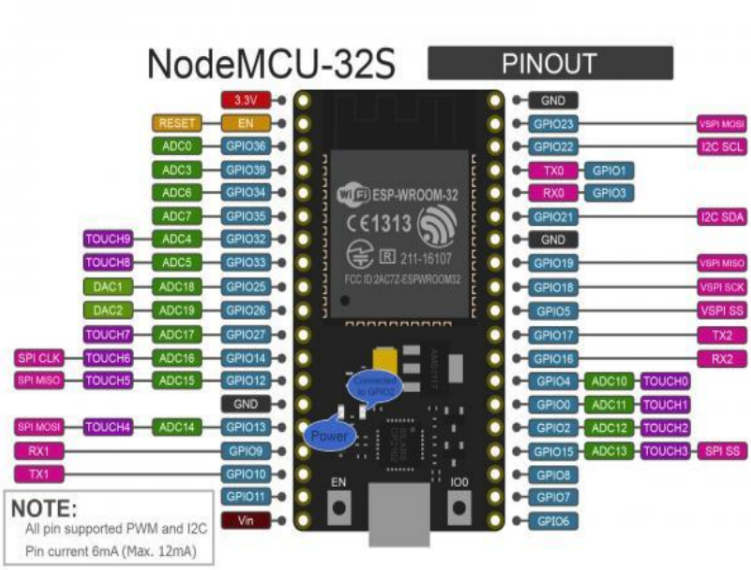
Os microcontroladores são muito relacionados a processadores, onde os microcontroladores são muito mais versáteis que uma unidade central de processamento CPU, do inglês *Central Processing Unit*. Este dispositivos possui periféricos que auxiliam na executação de muitas funções, o que o torna muito vantajoso em um projeto embarcado. Pode se dizer que um microcontrolador é uma espécie de computador, composto de um processador (CPU), memória de armazenamento de programa, que consiste em uma memória para armazenamento de variáveis. Alguns ainda possuem periféricos para comunicação, conversão analógico/digital etc - muito similar a um computador que é programado por meio das chamadas linguagens de programação, como a linguagem C. Existem muitos recursos que são limitados e que trazem alguns pontos negativos comparados a computadores, como frequência de *clock*, tamanho de memória de programa e memória RAM etc. Porém, existem vantagens como o baixo custo, baixo consumo energético e pequenas dimensões (FIOREZA, 2019).

2.4.1 NodeMCU-32S ESP32

Systems (2019), apresenta o NodeMCU-32S ESP32 como uma placa de desenvolvimento fundamentada no módulo ESP32S *WiFi*, um componente eletrônico demasiadamente tecnológico, desenvolvido especialmente para conectar projetos robóticos ou automação residencial a Rede Mundial de Computadores (Internet), com maior facilidade e baixo custo. Muito eficiente, possui o módulo ESP-WROOM-32, que além de conter o ESP32, conta com um cristal de 40 MHz, antena embutida e porta micro USB para alimentação e programação, 36 pinos GPIOS, Conversor Analógico Digital com 12 *bits* de resolução (até 18 canais), 2 DAC com 8 *bits* de resolução, além de antena embutida. Possui conexão *Wi-Fi* nativa, também com *Blueto-oth* V4.2 embutido e microprocessador *dual core 32-bit* LX, de forma a tornar seu projeto ainda mais prático e aumentar as possibilidades de uso.

Segundo FilipeFlop (2020), o ESP32 (vide Figura 2.7) reúne um apanhado de recursos que tornam seu uso em Internet das Coisas muito recorrente, sendo assim de fácil implementação utilizando os mais variados dispositivos e componentes. Dentre as interfaces de comunicação, ele possui suporte a SPI, UART e I2C, que são alguns dos protocolos mais comuns, como também tem suporte a infravermelho e SDIO que usa a interface com cartão de memória. Possui também CAN, *Ethernet*, DAC, sensor de toque, e I2S. As informações referentes as características técnicas do ESP32 pode-se verificar na Tabela 2.1.

Figura 2.7: NodeMCU-32S ESP32 *layout*



Fonte: 99TECH (2019).

2.5 PROGRAMAÇÃO

O módulo ESP32 é uma solução de baixo custo criada para conexões *Wi-Fi* permitindo que execute programas carregados em sua própria memória. Segundo Ricardo Rodriguez Oliveira (2017), o ESP32, assim como o ESP8266, podem funcionar de forma embarcada, sem precisar estar ligado a outro dispositivo fisicamente, podendo fazer a comunicação somente via *Wi-Fi* ou com sensores e saídas pelos pinos. Programação é a aplicação da lógica para criar algoritmos que possam executar uma determinada tarefa, e para transcrever essa lógica é utilizado

Tabela 2.1: Características ESP32

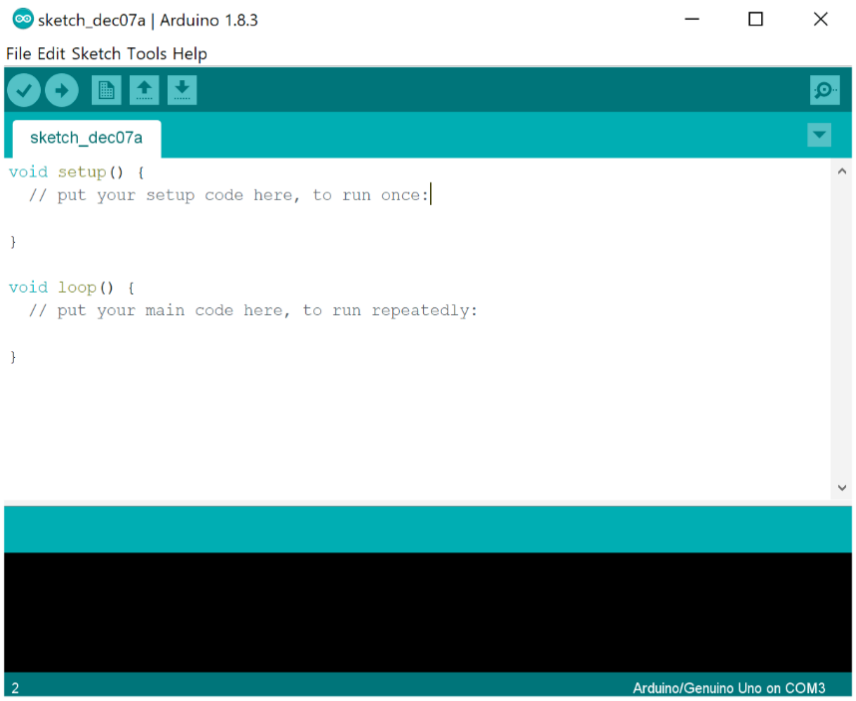
Nome	Características ESP32
Núcleos	2
Arquitetura	32 *bits*
Clock	160MHz
Wi-Fi	802.11 2.4GHz até 150Mbits/s
Bluetooth	V4.2 *Bluetooth Low Energy* (BLE)
RAM	512 kB
FLASH	16 MB
GPIO	36
Interfaces	SP1, I2C, UART, I2S e CAN
ADC	18
DAC	2

as mais diversas formas de programação seja para um ESP32 ou um computador.

2.5.1 Arduino IDE

Para programar o ESP32, pode-se utilizar o Arduino IDE, na Figura 2.8 pode-se visualizar a interface do Arduino IDE, onde o usuário pode preencher e compilar seus códigos. Arduino IDE é um *software* livre que lhe permite programar na linguagem que preferir. No caso do ESP32, a linguagem é baseada em C/C++ e pode até ser estendida por meio de bibliotecas C++. O IDE permite que se escreva um programa de computador, que é um conjunto de instruções passo a passo da lógica a qual deve ser seguida, a partir disto, deve-se fazer o *upload* para o ESP32. A seguir, o seu microcontrolador executa essas instruções e interage com o que estiver conectado a ele. No mundo dos embarcados, os programas são conhecidos como *sketches* ou esboços.

Figura 2.8: IDE (Ambiente de Desenvolvimento Integrado)



Fonte: ARDUINO (2015).

2.6 NODE-RED

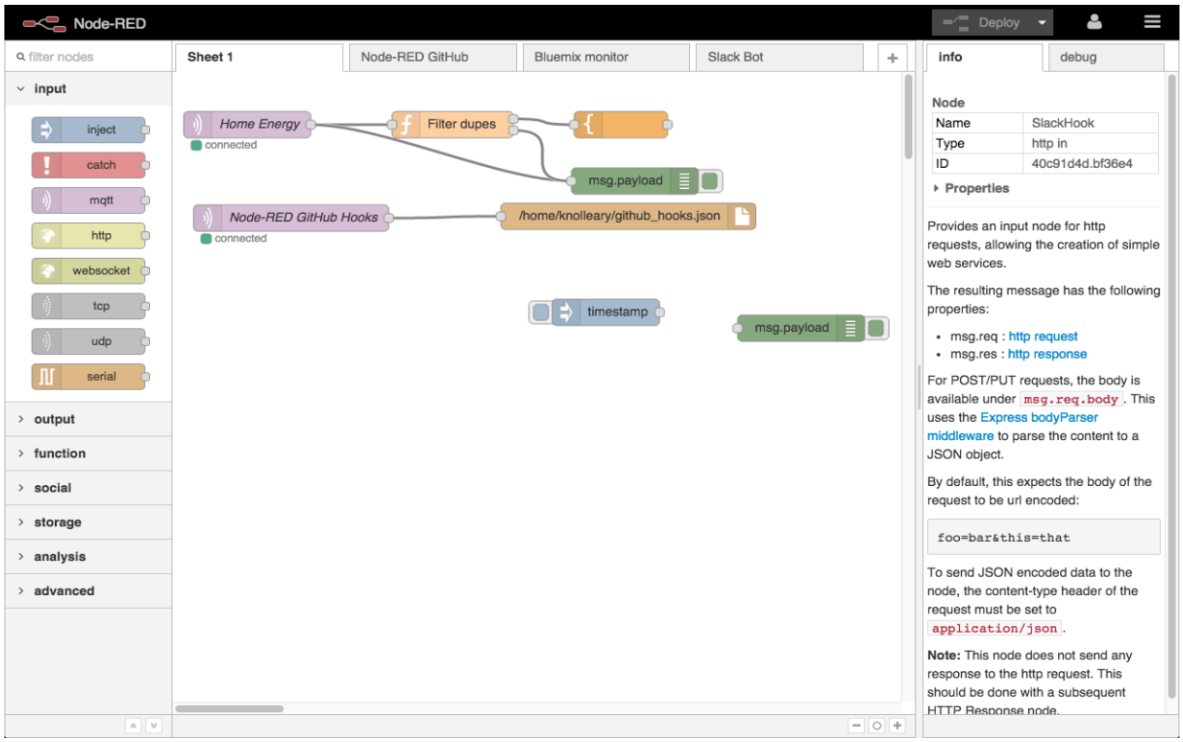
O Node-RED consiste em uma ferramenta bastante versátil, voltada principalmente para o desenvolvimento de aplicações relacionadas ao conceito de Internet de Coisas (IoT). Este recurso utiliza uma abordagem de programação gráfica, ou seja, torna-se possível elaborar uma aplicação por meio do estabelecimento de conexões entre blocos que possuem códigos predefinidos (conhecidos como nós) para a realização de determinadas tarefas.

Conforme a empresa Node-RED (2019), a programação é baseada em fluxo, sendo esta uma maneira de descrever o comportamento de um aplicativo, usando nós, como são chamados no Node-RED. Cada nó tem uma finalidade bem definida, recebendo dados, executando determinada tarefa e passando adiante. Por meio dos nodes ou nós é possível ler arquivos CSV, executar eventos *HTTP*, TCP, *websocket*, *twitter*, MQTT entre outros. Node-RED consiste em um *runtime* baseado em Node.js, o qual direciona a um navegador *Web*, acessando assim o editor de fluxo. No navegador, pode-se criar aplicativos arrastando nós para sua área de trabalho e conectandos, como pode-se verificar na Figura 2.9, onde na interface do Node-RED encontram-se os nós conectados. E assim como o protocolo MQTT (*Message Queue Telemetry Transport*), o Node-RED também foi criado pela IBM *Emerging Technology*.

2.6.1 Socket TCP

A principal razão da existência de redes é permitir ou oferecer subsídios para que dois computadores possam se comunicar independente da sua localização geográfica. Para que isso aconteça é preciso estabelecer alguns padrões para que os computadores possam se entender. Sendo este um pré-requisito mínimo para o funcionamento de uma rede entre dois computadores ou dispositivos, pois existe uma grande heterogeneidade entre equipamentos que serão conectados entre si. Desta forma, com estes pré-requisitos e alinhamentos de comunicação, nasce o *Socket* TCP, o qual prevê uma suíte de protocolos que estão preocupados com questões relacionadas desde a entrega do pacote até a maneira como essa informação é transportada. Assim como o modelo OSI, o TCP também está organizado em camadas e cada qual possui uma função específica dentro do objetivo macro de efetivar a comunicação entre dois *hosts* (nós). A característica mais básica e marcante deste modelo é a entrega da informação a qualquer custo. Isso significa que o pacote deve ser entregue ao destinatário mesmo que um caminho usual não esteja mais disponível. É através da operação do protocolo IP que a informação pode percorrer

Figura 2.9: Plataforma Node-Red



Fonte: NODE-RED (2019).

caminhos alternativos quando um meio não estiver operante (COMER, 2016).

2.7 WI-FI

Wi-Fi é a expressão usada atualmente para a tecnologia IEEE 802.11, que permite a conexão entre diversos dispositivos sem fio. Esta tecnologia amplamente utilizada na atualidade por dispositivos móveis, *notebooks* e em sistemas embarcados. As redes *Wi-Fi* funcionam por meio de ondas de rádio. Elas são transmitidas através de um adaptador chamado comumente de roteador, que recebe os sinais, decodifica e os emite a partir de uma antena. Segundo André Iasi Moura (2007), *Wi-Fi* (*Wireless Fidelity*) é um termo que identifica redes e dispositivos que implementam a especificação IEEE 802.11 para redes sem fio. Uma rede *Wi-Fi* estruturada, é composta por dispositivos que se comunicam por sinais de rádio-frequência dentro os quais um ou mais são Pontos de Acesso (PA). PAs são dispositivos que, por um lado, conectam-se à rede cabeada e, por outro, comunicam-se com os outros dispositivos *Wi-Fi* através de sinais de rádio-frequência, servindo como ponte para que tais dispositivos acessem a rede.

2.8 ALCANCE DE SINAL *WI-FI*

Este projeto apresenta um grande desafio quando o assunto é alcance de sinal de *Wi-Fi* dos dispositivos ESP32 instalados no estacionamento. Pensando em soluções, pode-se utilizar 2 dispositivos para aumentar esta distância do sinal do *Wi-Fi*, sendo eles o Roteador Access Point e o Repetidor de Sinal *Wi-Fi*. Ambos aparelhos podem utilizar o padrão *Power Over Ethernet* como meio de transmissão de energia e dados através de um único cabo, diminuindo assim a necessidade de infraestrutura elétrica para energizar os dispositivos.

2.8.1 Roteador Access Point

Roteador Access point é um dispositivo de rede usado para estender a cobertura de redes de *Wi-Fi*, onde seu funcionamento é conectado via cabo a um roteador conectado à Internet, distribuindo assim assim o sinal *Wi-Fi* para os dispositivos que estejam na área de alcance do roteador. O roteador AP, diferente de um repetidor de sinal, é mais direcionado ao uso empresarial, pois este possui maior desempenho e tolerância a um grande número de dispositivos conectados ao mesmo tempo.

2.8.2 Repetidor de Sinal *Wi-Fi*

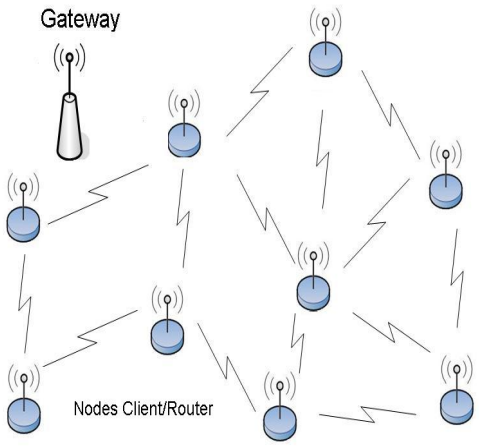
O repetidor de Sinal *Wi-Fi* é um dispositivo prático e compacto, o qual auxilia em projetos que necessitam de uma conexão com um maior alcance, ampliando assim sua área de cobertura para locais onde o sinal não é tão intenso. Este dispositivo pode ser instalado em qualquer ambiente. Este dispositivo como o próprio nome diz, ele repete o sinal, neste caso o sinal de *Wi-Fi*, estando este aparelho configurado à um roteador ou modem *Wi-Fi* conectado à Internet.

2.8.3 Arquitetura de Rede Mesh

As redes *mesh* sem fio utilizam o padrão 802.11s e permitem a interconexão de vários pontos de acesso entre si a fim de proporcionar uma ampla área de cobertura. Esse tipo de rede tem seu *backbone* muitas vezes estático e de conectividade sem fio. Uma rede *mesh* é composta por *gateways mesh* (WMG), roteadores *mesh* (WMR) e clientes *mesh*, exemplo de rede mesh na Figura 2.10 . Essa rede pode se tornar operável com apenas um nó *gateways* conectado à uma infraestrutura e compartilhando Internet com os outros nós, o que reforça um dos princípios da

rede, que é o baixo custo (SOUSA, 2016).

Figura 2.10: Rede Mesh



(SANTOS, 2009)

2.9 BANCO DE DADOS

Segundo Christopher J. Date (2004), um sistema de banco de dados é basicamente um sistema computadorizado de manutenção de registros; em outras palavras, é um sistema computadorizado cuja finalidade geral é armazenar informações e permitir que os usuários busquem e atualizem essas informações quando as solicitar. Estas informações em questão podem ser qualquer informação que tenha algum valor para determinado projeto, podendo assim, auxiliar no processo geral das atividades do projeto ao qual este banco de dados esta armazenando informações.

A implementação de um banco de dados neste trabalho possibilita armazenar as informações do sistema de gerenciamento de vagas de estacionamento. Pretende-se analisar esses dados para obtenção de informações relevantes baseado no histórico, assim como armazenamento de informações como a localização de cada equipamento e sua respectiva vaga instalada.

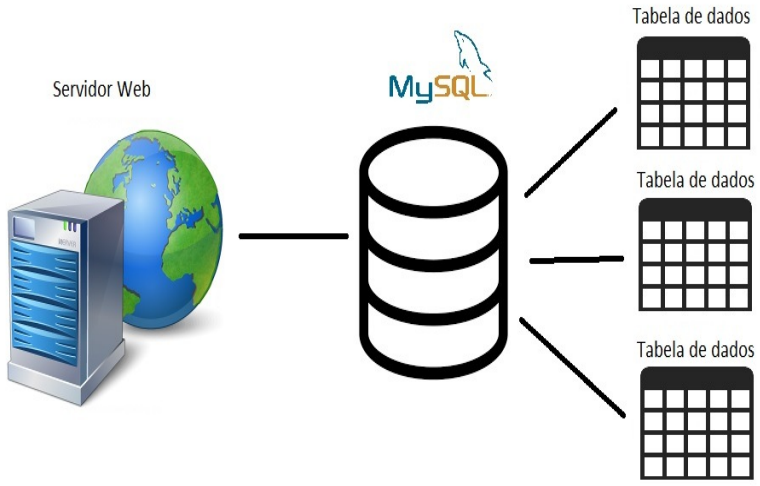
2.10 MYSQL

Conforme André Milani (2007, p.22), o MySQL é um servidor e gerenciador de banco de dados relacional, de licença dupla (sendo uma delas de *software* livre), projetado inicialmente para trabalhar com aplicações de pequeno e médio portes, mas hoje atendendo a aplicações de

grande porte e com mais vantagens do que seus concorrentes. Possui todas as características que um banco de dados de grande porte precisa, sendo reconhecido por algumas entidades como banco de dados *open source* com maior capacidade para concorrer com programas similares de código fechado, tais como SQL Server (da empresa Microsoft) e Oracle.

Segundo André Milani (2007, p.26), "o MySQL, além de banco de dados, contém todas as características de um SGBD (sistema gerenciador de banco de dados), que é o MySQL Server. Além de armazenar os dados, a ferramenta provê todas as características de multiacesso a estes, entre outras funcionalidade de um SGBD". Por exemplo, gerenciando o acesso, integridade dos dados de um sistema como deste trabalho.

Figura 2.11: Sistema Gerenciador de Banco de Dados



Fonte: Retirado de HOMEHOST (2017).

2.11 PHPMYADMIN

A empresa PhpMyAdmin (2013) descreve o PhpMyAdmin como uma ferramenta de software livre escrito na linguagem PHP (*Hypertext Preprocessor*), destinado a lidar com a administração do MySQL pela Internet, suporta uma ampla gama de operações em MySQL. As operações mais usadas (gerenciar bancos de dados, tabelas, colunas, relações, índices, usuários, permissões, etc) podem ser realizadas através da interface do usuário.

2.12 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O capítulo de revisão bibliográfica descreveu os principais elementos utilizados para a execução deste relatório. Havendo como objetivo a fundamentação teórica para esclarecer conceitos aprendidos durante a graduação de engenharia, clarificando assim o processo de construção de um sistema inteligente de gerenciamento e controle de vagas de estacionamento.

3 METODOLOGIA E MATERIAIS UTILIZADOS

3.1 INTRODUÇÃO

O objetivo fundamental deste capítulo tange a apresentação dos principais materiais empregados, assim como os circuitos do sistema embarcado, as conexões necessárias para acompanhamento em tempo real, bem como uma apresentação da lógica utilizada para a programação do microcontrolador quanto a plataforma Node-Red.

3.2 CAIXA PARA DETECÇÃO DE PRESENÇA

A caixa para detecção de presença surge da necessidade da identificação de vagas de estacionamento, pois a não identificação acarreta engarrafamentos e perda de tempo para os motoristas em shoppings, hospitais e outros estabelecimentos. Com o desenvolvimento do projeto é possível um maior controle quanto ao estado das vagas para o gestor e uma maior visibilidade para os motoristas na busca por uma vaga. Os circuitos de ligação a seguir, foram todos simulados no *software* Fritzing 2019 que é um *software* livre/*open source* que ajuda a modelar os circuitos de protótipos de uma maneira visualmente intuitiva.

3.2.1 Utilização do Sensor HC-SR04

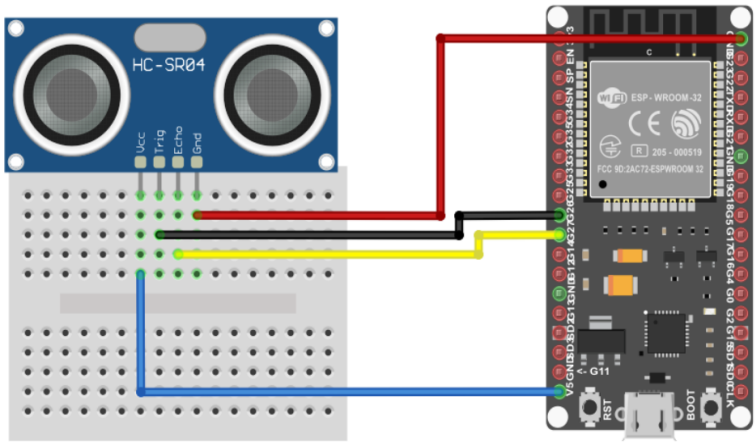
O sensor HC-SR04 é muito usado em produtos embarcados para detecção de presença, podendo também identificar a distância que um objeto ou pessoa está do sensor, abrindo assim uma grande gama de possibilidades de atuação no mercado. O emprego do sensor é muito simplificado, onde os pinos *Trig* e *Echo* são para leitura dos dados digitais e os pinos Vcc e Gnd para alimentação do sensor, conforme Figura 3.1.

O sensor é conectado diretamente no microcontrolador ESP32, onde os dados coletados pelo HC-SR04 serão analisados e processados. A leitura e processamento desses dados estão descritos na seção 3.3.5.

3.2.2 Sinalizador de presença

É de fundamental importância a visualização do estado das vagas de estacionamento para o motorista, principalmente no momento em que se busca uma vaga desocupada. E para

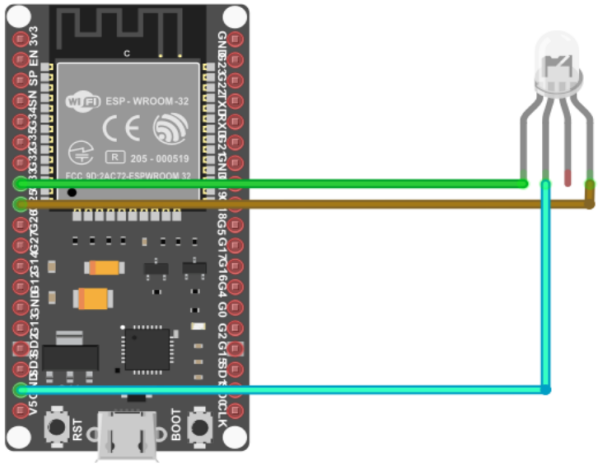
Figura 3.1: Circuito do sensor



Fonte: Autor.

melhor visualização foi implementado um LED RGB mostrando o *status* das vagas em duas cores, verde para desocupado e vermelho para ocupado. O sinalizador é interligado diretamente ao ESP32, conforme Figura 3.2, o qual é responsável pela troca dos *status* e consequentemente das cores.

Figura 3.2: SINALIZADOR DE PRESENÇA



Fonte: Autor.

3.3 PROGRAMAÇÃO DO NODE-RED

O Node-RED foi utilizado como uma ferramenta de programação para conectar os dispositivos do projeto assim como APIs e serviços *online*. O Node-RED é construído sobre o Node.js, valendo-se ao máximo do modelo sem empecilhos e orientado a eventos. Abstraindo protocolos e fazendo a interface de dados sem precisar explicitar código de programação, tudo por uma GUI de forma a tornar transparente e intuitiva a comunicação de diversas fontes de dados.

3.3.1 LÓGICA DE PROGRAMAÇÃO

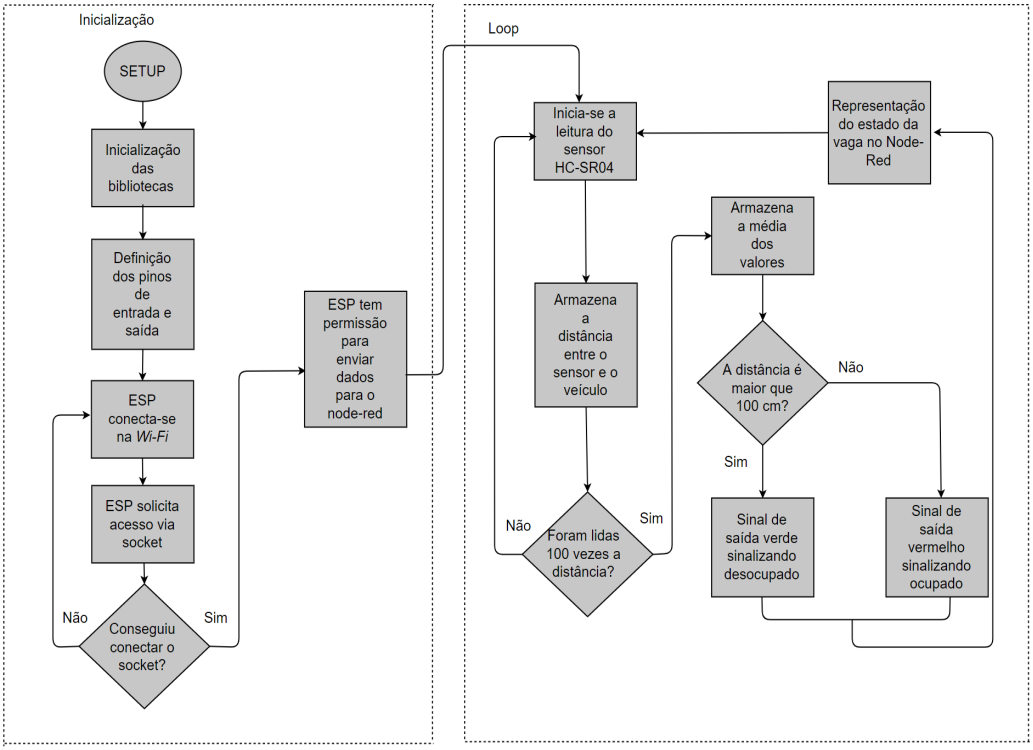
A lógica de programação implementada é representada por um fluxograma da Figura 3.3, sendo um diagrama que utiliza um conjunto padrão de símbolos gráficos que representam a ordem de instruções codificadas e introduzidas na memória do microcontrolador. Onde o microcontrolador será responsável pela execução das operações lógicas e aritméticas do projeto. A representação por fluxograma simplifica a visualização do programa como um todo, onde os símbolos básicos são o início, o processo onde os dados entram na lógica, os blocos de decisão e por fim o encerramento do programa.

3.3.2 Bibliotecas

Para conexão do ESP32 com a Internet, utilizou-se a biblioteca *Wi-Fi Multi* do criador me-no-dev. O *download* pode ser feito no portal GitHub. A *Wi-Fi Multi*, é uma biblioteca *open-source* escrita sobre outra biblioteca muito utilizada, a Wi-Fi.h. Ambas têm o propósito de facilitar a conexão de dispositivos com a rede *wireless*. Com esta biblioteca basta fazer uma solicitação de conexão com a rede determinada e em seguida anexar a senha da respectiva *Wi-Fi*, é feita a partir desses dados, se alguma informação não esteja correta, o ESP32 entra em *loop* tentando efetivar conexão com a rede.

Além das bibliotecas mencionadas, foram utilizadas as bibliotecas, *WebServer*, *8266WebServer* e *WiFiManager*. As duas primeiras foram utilizadas para conexão cliente servidor com o *WebServer*, a última biblioteca mencionada foi utilizada para facilitar a conexão a uma *Wi-Fi* já existente.

Figura 3.3: Fluxograma



Fonte: Autor.

3.3.3 Integração Node-RED com ESP32

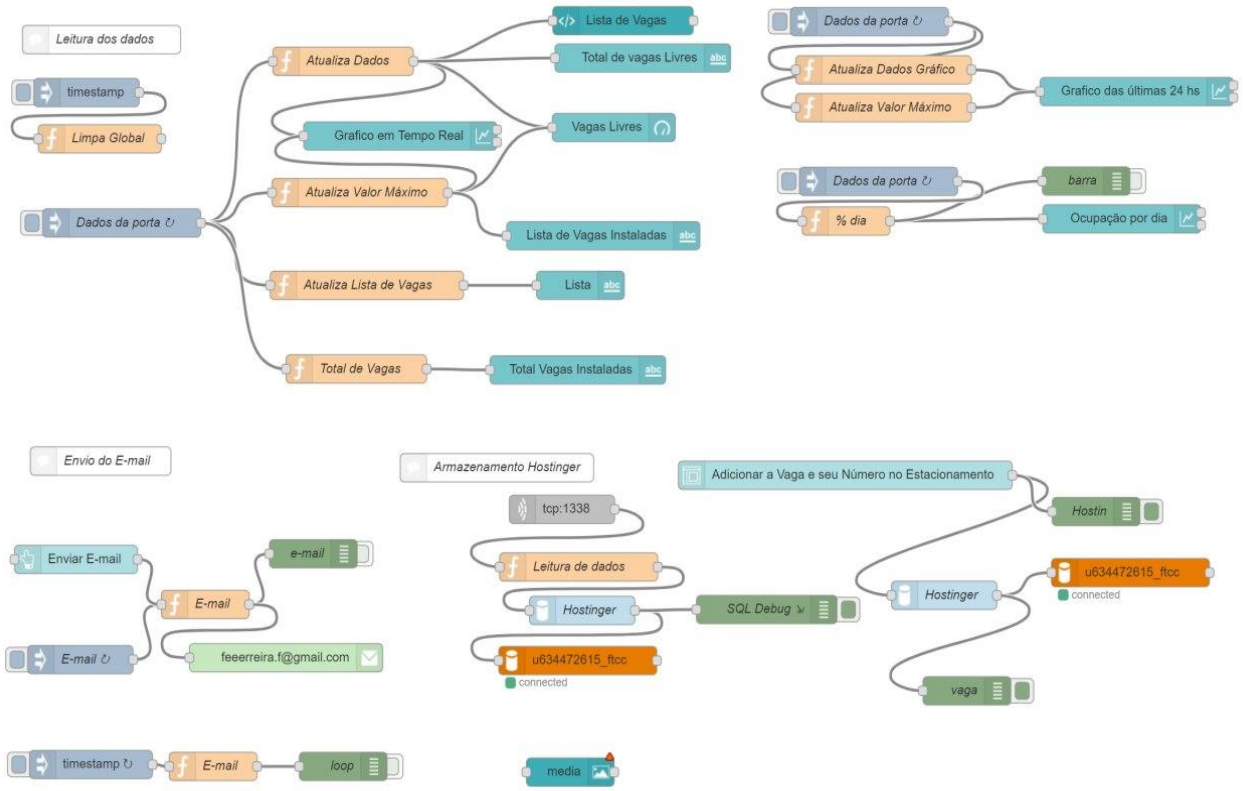
Utilizando a plataforma IDE do Arduino é possível programar o ESP32 para que possa enviar dados para os servidores do Node-RED usando o *socket* TCP. Com uma conexão estabelecida entre os dispositivos é possível controlar qualquer periférico remotamente, desde que conectado com o ESP32, através da sua própria interface *Web*. A interface é muito intuitiva e de fácil compreensão, podendo gerenciar todas suas entradas e saídas através das mais diversas formas, como visualização dos dados, alertas e etc.

Para que o sistema possa se integrar ao ESP32 é necessário a instalação e configuração dos *softwares* Node-RED e Node.js, e a configuração do Windows NT para suporte do Node-RED. Para que o sistema possa executar no Windows, antes de tudo é necessário configurar e após executar o Node-RED no *Prompt* de comando.

Neste projeto primeiramente é coletada a informação da vaga de estacionamento, verificando se ela esta ocupada ou livre, a partir disso é enviado para o servidor do Node-Red os dados coletados pelo sensor ESP32. Estes dados já estão convertidos para que possam servir

como informação para o usuário, chegando através de uma identificação de cada vaga, conforme Figura 3.5. Neste momento é constatado o ID e o estado da vaga. Para que haja uma sincronização entre o ESP32 e o servidor do Node-RED é pré-definida uma porta de comunicação do servidor local. Neste projeto foi definido a porta TCP 1338.

Figura 3.4: Programação em blocos na plataforma Node-Red



Fonte: Autor.

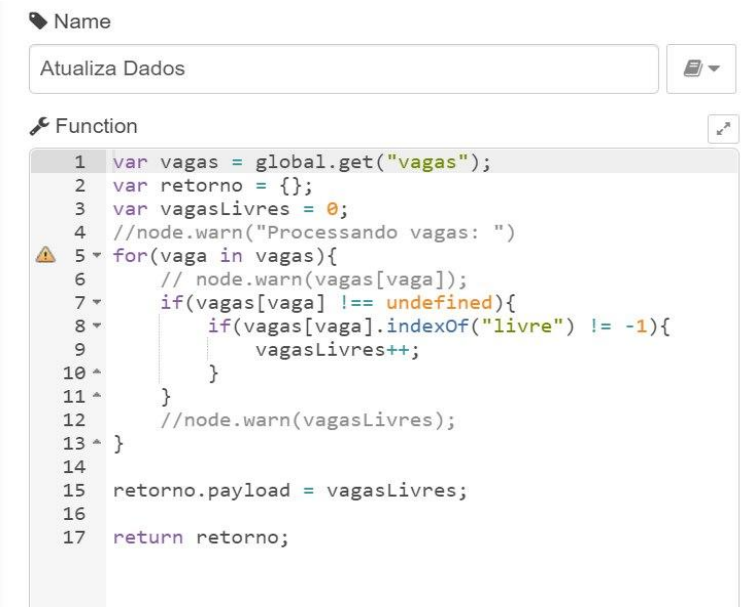
Com os dados coletados através da porta mencionada, pode-se então trabalhar com essas informações no Node-Red conforme Figura 4.20. Estas informações já filtradas podem então servir para amostragens no *dashboard*. A Figura 4.20 é o diagrama de blocos usado dentro da plataforma Node-Red, onde pode-se adicionar qualquer modelo de entrada ou de saída para dados. No caso específico deste projeto a entrada de dados acontece via protocolo TCP com o ESP32, coletando informações referentes a estado da vaga de estacionamento em tempo real. Com a coleta destes dados é possível manipulá-los dentro da Plataforma Node-RED das mais diversas formas, como por exemplo convertendo alguma informação para algum bloco ilustrativo como *template*, *delay*, *trigger* e etc. No caso deste projeto foi utilizado o modo de função, somando os valores binários de cada vaga (ocupado ou desocupado), e assim mostrar

a capacidade total em tempo real para o usuário. Com as informações filtradas e prontas para serem mostradas ao usuário, a plataforma Node-Red fornece os mais variados modos de saída de dados, neste caso foi utilizado a amostragem através de gráfico e *gauge*.

3.3.4 Leitura dos Dados

O sistema em seu pleno funcionamento envia os dados para o Node-RED em uma sequência, onde primeiro é identificado o *split* que neste caso é o caracter especial é a cerquilha. Este dividirá o conteúdo do dado em dois, onde a primeira parte é o ID da vaga e o segundo é o *status* da mesma, separados pelo símbolo de cerquilha, por exemplo "10#ocupado". Estes dois dados são armazenados de forma separada e enviados em forma de texto plano com codificação ASCII para o Node-RED via *Socket* TCP pela Porta 1338, sendo seus dados recebidos no Node-RED, (vide Figura 3.5), onde ocorre a interpretação dos mesmos, extraindo o número da vaga e o estado atual. Estes dois dados são registrados em uma variável global em forma de vetor. Com o dado organizado pode-se então utilizá-lo para as mais diversas formas de amostragem no *dashboard*.

Figura 3.5: Leitura de Dados na Plataforma Node-RED

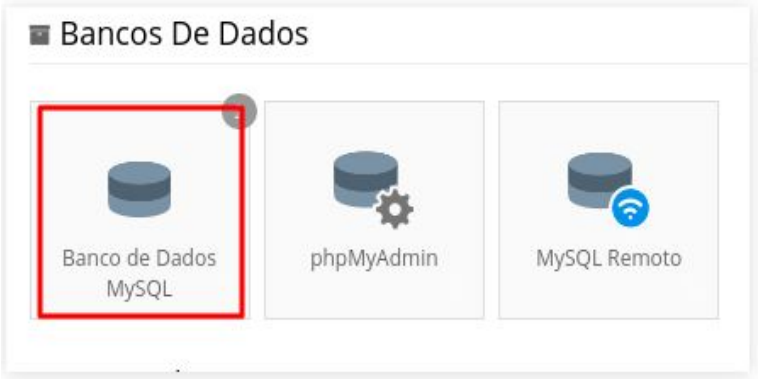


Fonte: Autor.

3.4 BANCO DE DADOS

Com o decorrer deste projeto seria inevitável o armazenamento das informações em algum banco de dados externo ao Node-RED. Foram estudadas então as empresas que fornecem o serviço de armazenagem de dados e optou-se pela empresa Hostinger. Foram comparadas questões como a valor, tamanho da armazenagem e a aplicabilidade do servidor para o projeto proposto. A empresa Hostinger após a comparação acabou sendo a mais adequeada a demanda. A vantagem da utilização de um servidor externo comparado ao servidor do Node-RED é a possibilidade de acessar de qualquer lugar do mundo, onde o *uptime* de um servidor externo é maior do que um servidor local, diminuindo as chances de queda do servidor. A desvantagem é a conexão com Internet, a qual não há necessidade em um servidor local.

Figura 3.6: Configuração MySQL



Fonte: Retirado de HOSTINGER (2020).

3.4.1 Configuração do Banco de Dados

Primeiramente, antes de configurar o banco de dados, foi definido que seria utilizado o MySQL para gerenciar o banco de dados. Além do banco de dados MySQL, a empresa hostinger possibilita também um banco de dados diretamente em phpMyAdmin ou MySQL remoto, como pode-se observar na Figura 3.6. A hospedagem da Hostinger possui a opção de um banco de dados MySQL, bastando apenas selecionar o mesmo como opção. Após a denominação e criação de usuários, são designados os privilégios do usuário, onde existem opções como por exemplo inserir, deletar e criar dados, dentro do próprio banco de dados.

4 RESULTADOS E PROTÓTIPO

4.1 CONFIGURAÇÃO DA REDE WI-FI NOS DISPOSITIVOS

Todos dispositivos para um funcionamento perfeito, devem estar conectados a uma rede *Wi-Fi*, para que possam enviar seus dados tanto para o servidor local, tanto para o banco de dados do projeto. E pensando em uma maior facilidade na hora de instalar o dispositivo, foi utilizada a biblioteca WiFiManager, a qual possibilita ao usuário configurar a rede e a senha, ao qual o ESP32 será conectado. Exclui-se assim a necessidade de programar estas informações direto no código fonte do ESP32.

Figura 4.1: Configuração Wi-Fi

Fonte: Autor.

Para que a configuração inicie, basta conectar o dispositivo a uma rede elétrica. No momento que o dispositivo estiver energizado ele automaticamente criara uma rede *Wi-Fi* de nome Config Vagas, conforme Figura 4.1. Ao se conectar nesta rede será habilitada a configuração

deste dispositivo, evitando assim a configuração através de programação, onde seria necessário acessar ao ESP32 e alterar as configurações de rede.

Conectado a rede Config Vagas, pode-se agora identificar as redes disponíveis no local e escolher a que possuir melhor sinal e acesso. Conforme a Figura 4.2, quando confirmado o acesso a rede, encontra-se a opção do botão Configure *Wi-Fi*, acessando assim à configuração do dispositivos e as redes disponíveis. Ao selecionar o botão consegue-se acesso a Figura 4.2, com todas redes à disposição e o nível do sinal da sua posição. Basta para configurar selecionar a rede, no caso Rede Estacionamento, e colocar a senha de acesso, aguardar alguns segundos e a configuração estará finalizada.

Figura 4.2: Acesso a rede Config Vagas

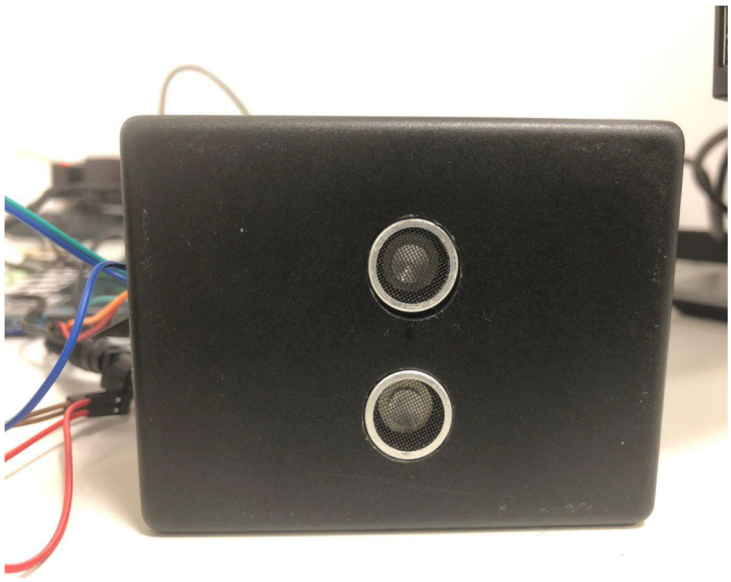
Fonte: Autor.

4.2 CAIXA PARA DETECÇÃO DE PRESENÇA

Neste projeto foi desenvolvida uma caixa para detecção de presença, esta caixa é instalada na parte superior de uma vaga de estacionamento para detecção de veículos estacionados. Foi instalado o sensor ultrassônico HC-SR04 na parte frontal da caixa, ficando expostos apenas os o emissor e receptor do sensor HC-SR04, conforme Figura 4.3.

Na parte interna da caixa consta o ESP32 para o processamento dos dados coletados pelo sensor ultrassônico HC-SR04, conforme já apresentado, o ESP32 transmitirá para o Node-

Figura 4.3: Caixa para detecção de presença



Fonte: Autor.

Red o estado da vaga em tempo real sem a necessidade de qualquer conexão física, sendo a transmissão dos dados através da rede *Wi-Fi*. A alimentação do sistema é através de um conversor de 220V AC para 5V DC.

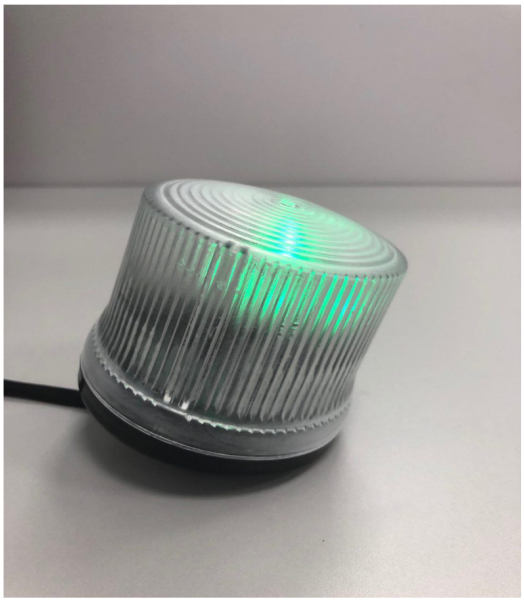
4.3 SINALIZADOR

De forma a complementar o sistema *Web* foi desenvolvido um sinalizador, (Figura 4.4), para maior identificação do motorista na hora de procurar uma vaga desocupada para seu veículo. Este sinalizador atua em dois estados, verde para vaga desocupada e vermelho para ocupada. O *status* da vaga é verificado pela caixa de detecção de presença apresentada acima que é instalada na parte superior da vaga, para que possa identificar quando algum veículo estacione abaixo do sensor ultrassônico HC-SR04.

4.4 INTERFACE GRÁFICA

Primeiramente a interface gráfica foi desenvolvida diretamente no *software* Node-Red, com o intuito de poder passar informações referentes à necessidade que o usuário possuía. Com o decorrer do projeto, pode-se notar mais informações de grande relevância, assim como pontos a serem melhorados, como alguns *bugs* no *dashboard* referentes a perda de sinal de uma das

Figura 4.4: Sinalizador em estado de vaga desocupada



Fonte: Autor.

caixas de detecção de presença. Para solucionar este problema foi ajustada a lógica dentro da *software* para que esperasse a conexão dos dois dispositivos para em seguida começar a mostrar as informações no *dashboard*, o qual é executado no computador do usuário, sendo assim uma plataforma local.

4.4.1 Visualização do *Dashboard Web*

Figura 4.5: Menu do Dashboard

Fonte: Autor.

O *dashboard* são painéis que mostram métricas e indicadores importantes para alcançar

objetivos e metas traçadas de forma visual. O *dashboard* facilita a compreensão do usuário no momento de analisar as informações geradas. Todas essas informações são via página *Web* (*World Wide Web*), apresentadas com o recurso de um navegador ou *browser*. A utilização de um navegador colabora pela praticidade, pois praticamente todos dispositivos hoje contam com um *browser* com acesso a Internet, e com isso, consigam ter acesso a estas informações. Dentro do *dashboard* tem um menu de abas do sistema Figura 4.5, onde existem 4 opções de *dashboards*, Estacionamento, Andares, Sistema Supervisório e Dispositivos. Essas opções serão abordadas detalhadamente.

4.4.1.1 Estacionamento

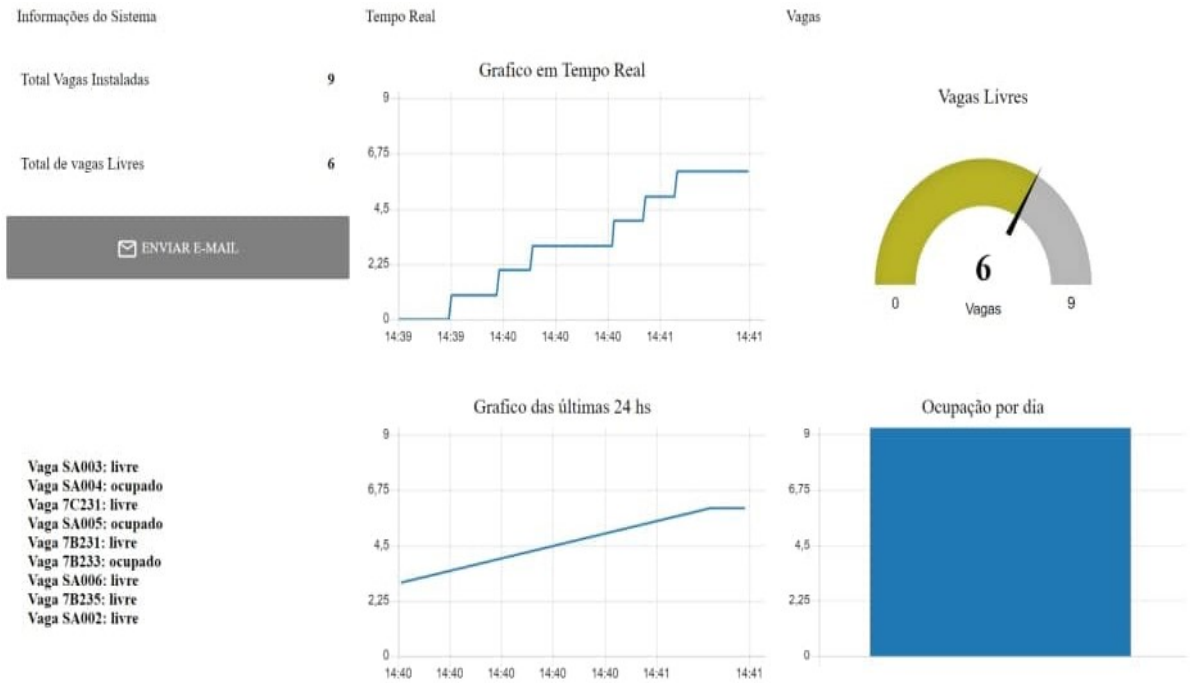
O *dashboard* Estacionamento, Figura 4.6, é a principal tela do sistema, onde pode-se observar as principais informações do sistema, no qual há informações já processadas, podendo assim apenas tomar alguma decisão. No primeiro terço desta tela estão as informações do sistema, informando dados como total de vagas instaladas, total de vagas livres e um acompanhamento dos dispositivos e seu respectivo estado. Foi incluido no sistema também a opção de envio da ocupação em tempo real para o e-mail cadastrado no sistema, seja este envio clicando no botão em cinza ou automaticamente a cada hora.

No segundo terço desta tela, está o tempo real, onde são gerados 2 gráficos em tempo real da ocupação das vagas em relação ao tempo. O primeiro gráfico tem a inclusão de dados a cada 5 segundos e o segundo gráfico a cada hora. E no último terço deste *dashboard*, encontra-se um *gauger* indicando as vagas livres e em seguida um gráfico diário pra acompanhamento à longo prazo, diferente dos gráficos do segundo terço desta tela.

Pensando na facilidade e utilização do usuário, neste projeto adicionou-se duas formas de enviar e-mail com o estado de ocupação do estacionamento. Pode-se enviar a mensagem ao acionar o botão de cor cinza à esquerda da Figura 4.6. Ao acionado, ele vai encaminhar o estado atual do estacionamento para o e-mail cadastrado.

Além do envio do estado atual via acionamento do botão, o e-mail enviado a cada hora automaticamente. Pode-se acompanhar de qualquer lugar por e-mail com uma informação simplificada conforme a Figura 4.7. Esta figura é do e-mail que chega ao usuário cadastrado no sistema.

Figura 4.6: Interface principal para o usuário



Fonte: Autor.

Figura 4.7: Envio de e-mail

Fonte: Autor.

4.4.1.2 Sistema Supervisório

Pensando na experiência do usuário e uma melhor visualização em tempo real do sistema, foi desenvolvido um sistema supervisório para que possa acompanhar em tempo real o estado das vagas de estacionamento. Foi inserida no *dashboard* do sistema supervisório a um exemplo hipotético de planta baixa de dois andares de estacionamento como pode-se visualizar na Figura 4.8. Neste exemplo de estacionamento estão alocadas as vagas e seus respectivos

nomes.

Figura 4.8: Sistema supervisório

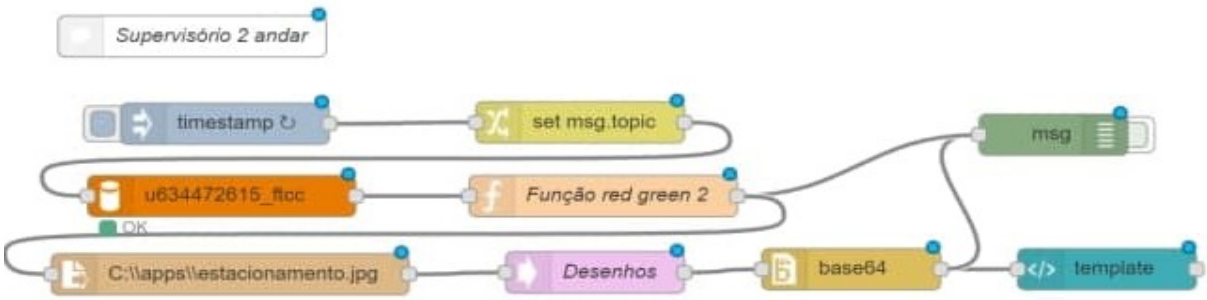
Fonte: Autor.

Nestas vagas, pode-se observar circulos nas cores vermelho e verde, estas cores representam a ocupação em tempo real das respectivas vagas, onde vermelho simboliza vaga ocupada e a cor verde caracteriza vaga livre. Estes circulos podem ser inseridos na aba Dispositivos, o qual será abordada na sequência.

Para o desenvolvimento deste sistema supervisório foram utilizados os *nodes Change* e *mysql*, para que se possa fazer o *request* no banco de dados das informações que serão utilizadas no sistema. Neste caso são a vaga e seu último estado. Foi utilizado para a parte gráfica o *node function*, *base64*, *file* e *template*, onde está a imagem da planta do estacionamento utilizando o *file*, *function* para a lógica dos estados dos circulos vermelho e verde e por último o *base64* e *template* para inserção e conversão das imagens para o *dashboard*. Na Figura 4.9 pode-se observar como foi feita a alocação dos nodes para a realização da aba sistema supervisório. Para o desenvolvimento da aba foi necessário a requisição dos *status* das vagas do segundo andar do banco de dados, com as informações coletadas, foi programado uma lógica que ilustrasse

o *status* da vaga sobreposta ao desenho da planta baixa do estacionamento. Nesta lógica as vagas ocupadas são desenhadas com círculos vermelhos, e para as vagas livres são desenhados círculos verdes, vide Figura 4.10 .

Figura 4.9: Nodes do sistema supervisório no Node-RED



Fonte: Autor.

Figura 4.10: Lógica do sistema supervisório no Node-RED

```
msg.annotations = [];

var yAtual = 150;
for(var itemVaga in msg.payload){
    var vaga = msg.payload[itemVaga];

    var dadosVaga = {};
    dadosVaga.type = "circle";
    dadosVaga.r = 10;
    dadosVaga.lineWidth = 2;
    dadosVaga.fontColor = "#000000";

    // O que muda é a poisção e a cor
    dadosVaga.x = vaga.posicaoX;
    dadosVaga.y = vaga.posicaoY;
    dadosVaga.stroke = "green";
    //dadosVaga.label = vaga.idVaga;
    if(vaga.estado == 0){
        dadosVaga.stroke = "red";
    }


    msg.annotations.push(dadosVaga);
    yAtual += 30;
}


return msg;
```

Fonte: Autor.

4.4.1.3 Registro da Localização dos dispositivos

Na aba Dispositivos, Figura 4.11, foi desenvolvida uma forma para relacionar os dispositivos instalados nas suas respectivas vagas, mas de uma forma que não fosse programável e sim adicionada pela interface do sistema. Dessa forma foi implementado o *node form*, onde pode-se adicionar os seguintes dados: Dispositivo; Vaga; Andar e as coordenadas da vaga X e Y. Estas posições, podem ser adquiridas na planta ou até mesmo com *softwares* de desenho. Estas coordenadas vão auxiliar para o posicionamento dos circulos vermelhos e verdes do sistema supervisório. Com as informações inseridas neste formulário, elas são salvas diretamente no banco de dados.

Figura 4.11: Dashboard para relacionar a posição de um dispositivo à uma vaga

Fonte: Autor.

4.4.1.4 Andares

Pensando em uma futura melhoria para o sistema, foi criada a aba Andares, onde pode-se adicionar indicadores de vagas disponíveis em cada andar, e dispondo estes indicadores em monitores pelo estacionamento. O banco de dados já esta estruturado para um *request* dessas informações, bastando apenas a criação de um *node request, mySQL* e *template*, para requisição dos dados necessários e incrementar na parte visual para visualização no *dashboard*. Pode-se visualizar na Figura 4.12 um exemplo dessa futura melhoria. Pode-se observar os Níveis A, B e C, nomes dos andares e ao lado direito em vermelho as vagas disponíveis nos respectivos

andares. Com esta melhoria pode facilitar ainda mais o acesso dos motoristas à uma vaga disponível para estacionarem seu veículo.

Figura 4.12: Display sinalizando as vagas livres em cada andar do estacionamento

(TECHPARKING, 2017)

4.5 CONFIGURAÇÃO BANCO DE DADOS

Para a escolha do serviço de hospedagem deste projeto analisou-se inúmeras empresas que fornecem esta solução. Primeiramente foi analisado aquela que tivesse compatibilidade com o Node-Red. Outro fator foi a simplicidade para configuração tanto para o Node-Red, quanto para o phpMyAdmin, pois estes itens devem ter a maior compatibilidade possível para que não haja problemas futuros quanto armazenagem dos dados no banco. Com estes itens de acordo, analizou-se o preço mensal dos serviço.

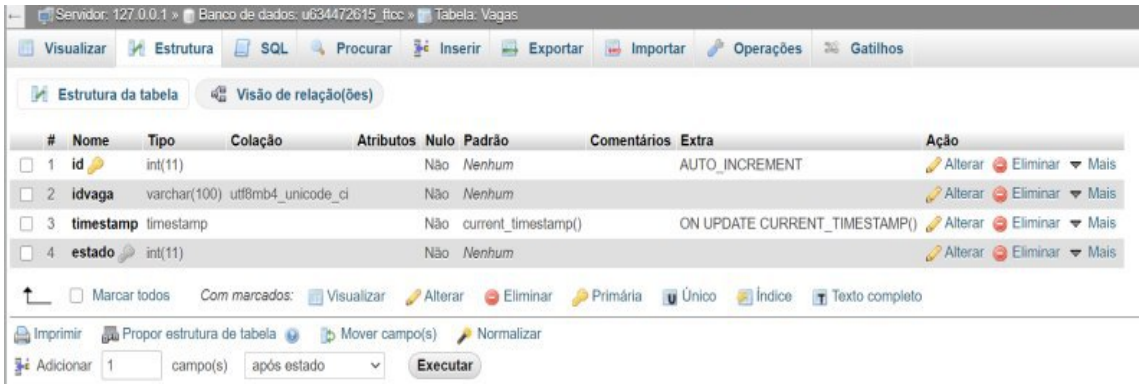
Para este projeto foram criadas duas tabelas. A tabela Vagas e a tabela NumeroVagas. A primeira serve para a entrada de dados dos dispositivos de detecção de presença, onde os *status* de cada vaga vão ser armazenados. A segunda tabela serve para referenciamento do posicionamento dos equipamentos instalados e suas devidas vagas.

4.5.1 Criação da Tabela Vagas pelo phpMyAdmin

Para criação desta primeira tabela foram definidos os seguintes dados, id, dado no formato *int* (inteiro, podendo apenas constar número inteiros) e com auto incremento (onde cada entrada de dado ele vai se auto incrementar). Este dado serve principalmente para saber a quantia de dados que vão entrar e assim ter um contagem. O segundo dado é o idvaga, dado no formato *varchar* (pode conter letras ou números). Nesta coluna é adicionado o *MAC adress* do dispositivo de detecção de presença. A terceira coluna é do *timestamp* e tem como finalidade a demarcação do tempo quando o dado entrou no banco de dados. E por último, o estado. Esta coluna é para identificar o estado ou *status* da vaga na hora que foi submetido o dado dentro do banco de dados. A Figura 4.13 mostra a contrução da tabela de dados do banco de dados.

Esta tabela tem a exclusiva entrada de dados que é dos dispositivos instalados e conectados ao sistema pela porta 1338. Então toda essa tabela é autopreenchida pelos dispositivos, bastando apenas acompanhar e trabalhar com as informações já existentes como pode-se observar na Figura 4.14.

Figura 4.13: Configuração da tabela de dados Vagas

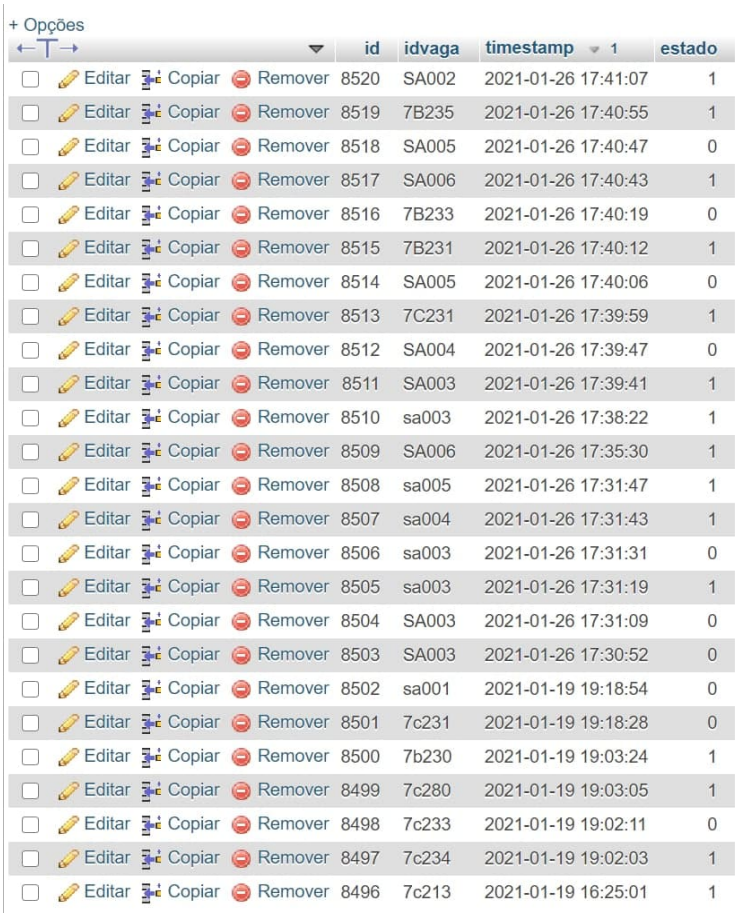


Fonte: Autor.

4.5.2 Criação da Tabela NumeroVagas pelo phpMyAdmin

A finalidade desta tabela, Figura 4.15 é para a localização dos dispositivos em uma instalação. O usuário registra 5 dados de grande valia para o sistema, como o número do *MAC adress* do dispositivo, o número da vaga que foi instalado, o andar o qual foi inserido e por último as coordenadas X e Y do posicionamento da vaga na planta baixa do projeto do estacionamento. Esta informação pode ser adquirida utilizando qualquer programa de edição

Figura 4.14: Tabela de dados Vagas



+ Opções

	id	idvaga	timestamp	estado
Editar Copiar Remover	8520	SA002	2021-01-26 17:41:07	1
Editar Copiar Remover	8519	7B235	2021-01-26 17:40:55	1
Editar Copiar Remover	8518	SA005	2021-01-26 17:40:47	0
Editar Copiar Remover	8517	SA006	2021-01-26 17:40:43	1
Editar Copiar Remover	8516	7B233	2021-01-26 17:40:19	0
Editar Copiar Remover	8515	7B231	2021-01-26 17:40:12	1
Editar Copiar Remover	8514	SA005	2021-01-26 17:40:06	0
Editar Copiar Remover	8513	7C231	2021-01-26 17:39:59	1
Editar Copiar Remover	8512	SA004	2021-01-26 17:39:47	0
Editar Copiar Remover	8511	SA003	2021-01-26 17:39:41	1
Editar Copiar Remover	8510	sa003	2021-01-26 17:38:22	1
Editar Copiar Remover	8509	SA006	2021-01-26 17:35:30	1
Editar Copiar Remover	8508	sa005	2021-01-26 17:31:47	1
Editar Copiar Remover	8507	sa004	2021-01-26 17:31:43	1
Editar Copiar Remover	8506	sa003	2021-01-26 17:31:31	0
Editar Copiar Remover	8505	sa003	2021-01-26 17:31:19	1
Editar Copiar Remover	8504	SA003	2021-01-26 17:31:09	0
Editar Copiar Remover	8503	SA003	2021-01-26 17:30:52	0
Editar Copiar Remover	8502	sa001	2021-01-19 19:18:54	0
Editar Copiar Remover	8501	7c231	2021-01-19 19:18:28	0
Editar Copiar Remover	8500	7b230	2021-01-19 19:03:24	1
Editar Copiar Remover	8499	7c280	2021-01-19 19:03:05	1
Editar Copiar Remover	8498	7c233	2021-01-19 19:02:11	0
Editar Copiar Remover	8497	7c234	2021-01-19 19:02:03	1
Editar Copiar Remover	8496	7c213	2021-01-19 16:25:01	1

Fonte: Autor.

de imagens como na Figura 4.16, posicionando o indicador do programa em cima da vaga do estacionamento, obtém-se suas coordenadas X e Y no canto inferior esquerdo da Figura 4.16. Situa-se com estas informações a vaga e seu *status* no sistema supervisório.

Figura 4.15: Tabela de dados NumeroVagas

Estrutura da tabela | Visão de relação(ões)

#	Nome	Tipo	Colação	Atributos	Nulo	Padrão	Comentários	Extra	Ação
1	id	int(11)			Não	*Nenhum*		AUTO_INCREMENT	Alterar Eliminar Mais
2	vaga	varchar(100)	utf8mb4_unicode_ci		Não	*Nenhum*			Alterar Eliminar Mais
3	numvaga	varchar(11)	utf8mb4_unicode_ci		Não	*Nenhum*			Alterar Eliminar Mais
4	andar	int(11)			Não	*Nenhum*			Alterar Eliminar Mais
5	posicaoX	int(11)			Não	0			Alterar Eliminar Mais
6	posicaoY	int(11)			Não	0			Alterar Eliminar Mais
7	timestamp	timestamp			Não	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Alterar Eliminar Mais

Fonte: Autor.

Figura 4.16: Aquisição das coordenadas X e Y

Fonte: Autor.

4.6 REDE WI-FI COM ARQUITETURA MESH

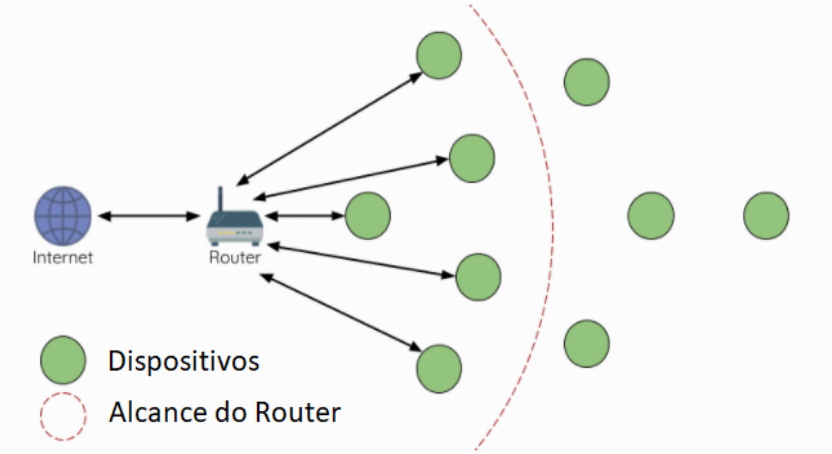
Este projeto por abranger uma área muitas vezes de grandes dimensões, necessita que a rede *Wi-Fi* tenha o alcance de sinal suficiente em todos dispositivos instalados, para que não ocorra a perda de dados e consequentemente falhas no projeto. Pensando neste possível caso, pode-se utilizar a arquitetura de rede *mesh* para melhorar o alcance e desempenho da rede *Wi-Fi*.

Com a implementação deste projeto pode-se utilizar a rede mesh sem fio para estacionamentos de maior porte, onde apenas um roteador *wireless* não conseguiria ter alcance para todos dispositivos. Para isso pode-se utilizar roteadores AP com repetidores de sinal *Wi-Fi*, onde este conjunto de equipamentos conseguem obter um alto alcance do sinal do *Wi-Fi*, concebendo assim uma área maior de cobertura da rede comparado a um roteador *wireless*, mantendo desse modo a conexão a Internet dos ESP32 instalados.

4.6.1 Rede Mesh utilizando ESP32

A rede mesh funciona diferente de uma rede *Wi-Fi* tradicional, Figura 4.17, onde a rede é ponto a multipontos, onde um único ponto de acesso é conectado diretamente a todos os outros nós conhecidos, como por exemplo um roteador e os dispositivos conectados neste aparelho. Como este projeto pode abranger áreas muito grandes, sendo esta uma das desvantagens do *Wi-Fi*, é justamente a cobertura de rede limitada devido ao requisito de que cada dispositivo deve estar dentro do alcance do roteador. Outra desvantagem é a limitação dos roteadores em dispositivos conectados ao mesmo tempo, podendo assim ter a desconexão de dispositivos pelo excesso de aparelhos conectados.

Figura 4.17: Arquitetura tradicional de rede *Wi-Fi*

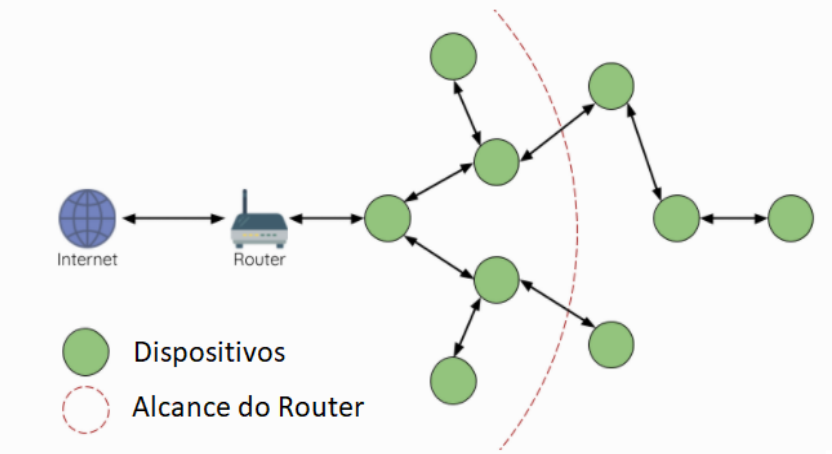


(ESPRESSIF, 2020)

Analisando todos impasses que podem ocorrer devido a conexão dos dispositivos ESP32 a rede *Wi-Fi*, opta-se por utilizar então a topologia de rede *mesh*, como já abordado é uma comunicação sem fio com nós organizados em uma topologia de malha usando o recurso do ESP32 o AP-STA, desenvolvido pela fabricante do ESP32, a Espressif. Com essa topologia de rede o ESP-WIFI-MESH, nome declarado pela fabricante, pode escalar até 1000 nós em grandes áreas, não necessitando de qualquer auxílio de infraestrutura *Wi-Fi*. Este metódo pode auxiliar a cobrir pontos de longo alcance e até mesmo pontos onde o *Wi-Fi* não tem alcance, conforme Figura 4.18.

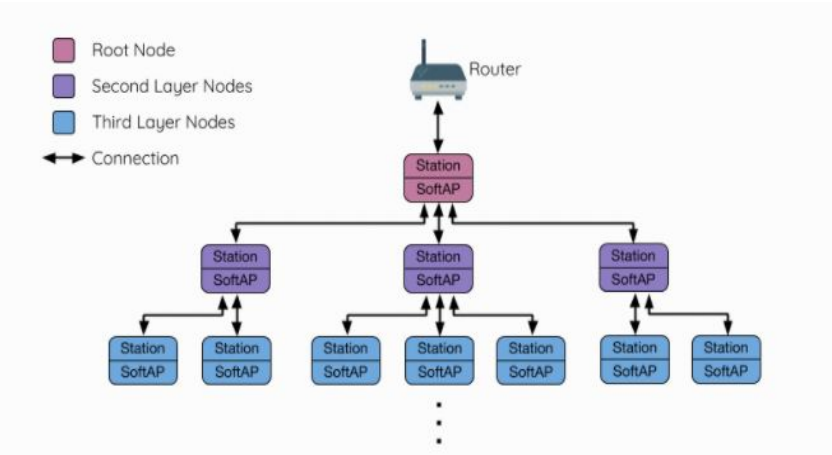
O ESP-MESH é arquitetado usando a topologia de árvore, Figura 4.19, construído sobre o protocolo de infraestrutura *Wi-Fi* e pode ser um protocolo de rede que combina muitas rede *Wi-Fi* individuais em uma única *WLAN*. No *Wi-Fi*, as conexões são limitadas a uma única liga-

Figura 4.18: Arquitetura de rede ESP-MESH



(ESPRESSIF, 2020)

Figura 4.19: Topologia de árvore ESP-MESH



(ESPRESSIF, 2020)

ção com um AP (conexão *upstream*) a qualquer momento, enquanto um AP pode ser conectado simultaneamente a várias estações (conexões *downstream*). No entanto, o ESP-MESH permite que os nós atuem ao mesmo tempo como uma estação e um AP. Portanto, um nó ESP-MESH pode ter várias conexões *downstream* usando sua interface *softAP*, embora tenha simultaneamente uma única conexão *upstream* usando sua interface de estação. Isso naturalmente resulta em uma topologia de rede em árvore com uma hierarquia pai-filho consistindo em várias camadas (ESPRESSIF, 2020).

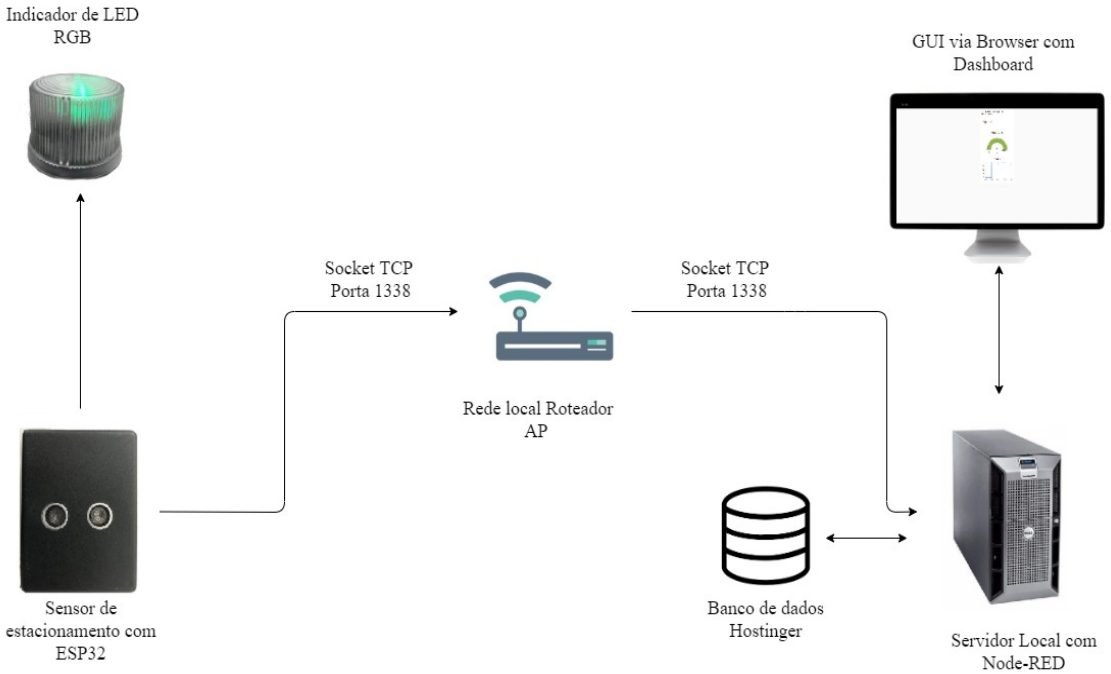
Na Figura 4.18 encontra-se a representação da Rede *mesh* que se enquadra neste projeto, onde pode-se utilizar um roteador AP com o auxílio de repetidores de sinal *Wi-Fi* para aumentar

o alncance da rede *Wi-Fi*. Com este conjunto de equipamentos pode-se aumentar o alcance da rede e com a implementação da topologia *mesh*, garantir que nenhum dado seja perdido do projeto.

4.7 IMPLEMENTAÇÃO

Para a execução deste sistema primeiramente foi analisada a infraestrutura dos estacionamentos e como o processo de procura de vagas acontece nestes estabelecimentos. A partir disso foi detectado a falta de visibilidade e localização de um lugar desocupado e assim partiu-se para o desenvolvimento do sinalizador em conjunto com a caixa de detecção de presença. A detecção desde o início se mostrou muito estável quanto a leitura da distância, principalmente se instalada longe do veículo a ser detectado. Porém para maior eficácia das leituras é feita a coleta destes dados 100 vezes e após é realizada uma média destes valores e assim repassada a informação para o atuador, no caso o sinalizador.

Figura 4.20: Diagrama de implementação



Fonte: Autor.

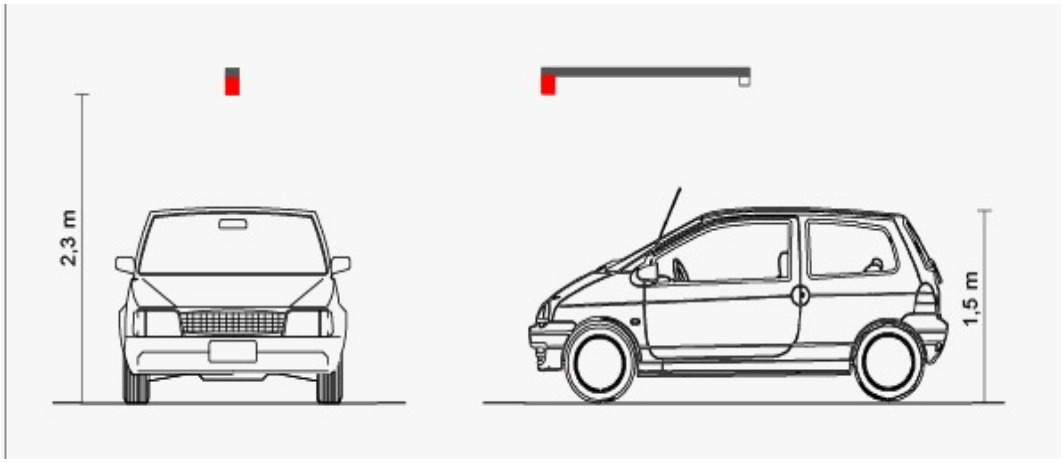
Foram realizados 3 ensaios para aferição da caixa detectora e do sistema como um todo, no primeiro ensaio foi para averiguar a precisão de leitura do sensor ultrassônico, com a intenção de obter a menor taxa de variância entre os dados coletados, foi verificado que 1 metro era um

valor aceitável e de bom posicionamento acima de um veículo. Com todas estas aferições o sistema final pode ser conferido no diagrama da Figura 4.20.

Na Figura 4.21 está representada a arquitetura física do sistema, onde a figura de um veículo estacionado representando a vaga de estacionamento, e acima dele o sistema descrito neste trabalho. A caixa de cor vermelha representa o indicador de LED do *status* da vaga, que no caso esta ocupada e por esse motivo a indicação vermelha. Ao lado a caixa de cor branca, representando a caixa de detecção de presença, localizada acima do veículo. Ambas as caixas localizadas a 2,3 metros de altura, para melhor visualização do motorista que esta buscando uma vaga, no caso do sinalizador, e para uma melhor leitura do *status* da vaga de estacionamento, no caso da caixa de detecção. E para o cliente/gestor do estacionamento é passado o *dashboard*, o qual fica localizado localmente no computador do cliente, onde também esta localizado o servidor.

Partindo para uma aprimoração do sistema foi implementado um banco de dados para armazenamento das informações do sistema, o qual nos possibilita acessar de qualquer lugar do mundo, obtendo um uptime maior do que por exemplo um sistema de armazenagem local, aumentar a segurança dos dados armazenados e diminuir a chance de cair o server. Neste banco de dados são armazenados importantes dados do projeto, como os dispositivo instalados, estado atual, mapa do estacionamento, quantidade de dispositivos instalados. Foi implementado um sistema supervisório do estacionamento, no qual o usuário poderá ter uma visão mais ampla da ocupação do estacionamento em tempo real, facilitando assim a compreensão da situação do estacionamento no exato momento, assim como auxiliando em possíveis tomadas de decisão. Com as aprimorações o sistema fica apto a futuras melhorias, como citado no trabalho, a criação de placas para indicação das vagas livres e seu respectivo andar. E pensando em projetos de grandes proporções pode-se utilizar o novo protocolo de comunicação, o ESPMESH, o qual pode alcancer maiores distancias comparado com redes *Wi-Fi*, bastando atualizar o sistema no ESP32 já instalado.

Figura 4.21: Arquitetura física do sistema



Fonte: Autor.

5 CONCLUSÃO

O estudo realizado por este trabalho mostrou uma aplicabilidade para o sistema inteligente de gerenciamento de vagas de estacionamento, viu-se a necessidade de gerenciar e supervisionar as vagas de estacionamento de forma remota e precisa. Além disso poder contar com um sistema supervisório que possa auxíliar numa melhor visualização da ocupação das vagas em tempo real.

No que compete a interface gráfica, o objetivo proposto era de uma plataforma de simples visualização que pudesse simplificar as informações relevantes do estacionamento em alguns indicadores. Além destes indicadores, a plataforma prioriza a simplicidade e uma usabilidade fácil, e pensando nisso o sistema conta com um sistema supervisório para verificação das vagas em tempo real.

A utilização de bibliotecas e da plataforma Node-RED facilitou a implementação deste projeto, principalmente para a conexão TCP/IP entre o ESP32 com o servidor. O desenvolvimento do *dashboard*, sistema supervisório e automações do sistema são todas feitas através da plataforma Node-RED.

No quesito à parte de *hardware* do sistema, foram utilizados componentes com a maior assertividade e que pudessem ter compatibilidade com o controlador ESP32, e que fossem compactos para uma fácil instalação. Este projeto tem muito potencial comercialmente, se tornando escalável quanto a instalação e produção do sistema. No que se refere ao custo, o sistema custa aproximadamente 200 reais, variando conforme a cotação do dólar, podendo este preço ser reduzido na compra em maiores escalas.

Com bastante estudo sobre a plataforma Node-RED e os *hardwares*, tornou-se possível desenvolver uma interface gráfica e a caixa de detecção de presença. O desenvolvimento destes itens foi de grande importância para alcançar os objetivos propostos neste projeto. Todos objetivos traçados foram atingidos, foi possível desenvolver a interface gráfica de fácil utilização e compreensão para o usuário, e um *hardware* de simples instalação, podendo a configuração de *Wi-Fi* do *hardware* ser configurado através de um dispositivo celular. E a alimentação do dispositivo sendo através de uma tomada de alimentação. Este *hardware* também teve êxito no que se refere à leitura da ocupação da vaga.

Projetando melhorias futuras, um dos aspectos que podem ser melhorados primeiramente é quanto ao desenvolvimento de telas de auxílio para os motoristas que adentram o es-

tacionamento, indicando onde há vagas disponíveis e seus respectivos andares. Outra melhoria que pode ser feita é quanto a implementação da arquitetura de rede *mesh*, aumentando o alcance do projeto e possibilitando assim uma maior quantia de dispositivos instalados em um estacionamento.

REFERÊNCIAS

99TECH. NodeMCU-32S ESP32 WiFi, Bluetooth, USB, PCB Antenna, ESP-32S, 38 pins. , [S.l.], 2019. Acessado em Agosto 2019. Disponível em: <https://99tech.com.au/product/nodemcu-32s/>.

ADARSH, S. et al. Performance comparison of Infrared and Ultrasonic sensors for obstacles of different materials in vehicle/robot navigation applications. In: IOP CONFERENCE SERIES: MATERIALS SCIENCE AND ENGINEERING. **Anais...** [S.l.: s.n.], 2016. v.149, n.1, p.012141.

ARDUINO. Arduino. , [S.l.], 2015. Acessado em Agosto 2019. Disponível em: <https://www.arduino.cc/>.

BANDEIRA, T. B. et al. Protótipo de estacionamento automatizado utilizando modelo computacional matricial e microcontrolador arduino. **Blucher Mathematical Proceedings**, [S.l.], v.1, n.1, p.817–824, 2015.

BRASIL, R. F. Eletrônica Progressiva. , [S.l.], 2014. Acessado em Agosto 2019. Disponível em: <https://www.eletronicaprogressiva.net/2014/08/Microcontroladores-O-que-sao-Para-que-servem-Onde-sao-usados.html>.

BULLOUGH, J. Lighting answers: led lighting systems. **National Lighting Product Information Program, Lighting Research Center**, [S.l.], v.7, n.3, 2003.

COMER, D. **Interligação de Redes com TCP/IP–**: princípios, protocolos e arquitetura. [S.l.]: Elsevier Brasil, 2016. v.1.

DATE, C. J. **Introdução a sistemas de bancos de dados**. [S.l.]: Elsevier Brasil, 2004.

ESPRESSIF. Espressif. , [S.l.], 2020. Acessado em novembro 2020. Disponível em: <https://www.espressif.com/en/products/sdks/esp-wifi-mesh/overview>.

FILIPEFLOP. Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino. , [S.l.], 2011. Acessado em Agosto 2019. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino>.

FILIPEFLOP. espfilipe. , [S.l.], 2020. Acessado em julho 2021. Disponível em: <https://www.filipeflop.com/blog/esp32-um-grande-aliado-para-o-maker-iot/>.

FIOREZA. Microcontroladores. , [S.l.], 2019. Acessado em Agosto 2019. Disponível em: <https://fiozera.com.br/microcontroladores-914a59cbf7de>.

FRITZING. **Software Fritzing**. Acessado em Agosto 2019. Disponível em: <https://fritzing.org/home/>.

HOMEHOST. Sistema Gerenciador de Banco de Dados. , [S.l.], 2017. Acessado em setembro de 2020. Disponível em: <https://www.homehost.com.br/blog/tutoriais/mysql/o-que-e-um-banco-de-dados/>.

HOSTINGER. Hostinger. , [S.l.], 2020. Acessado em setembro 2020. Disponível em: <https://www.hostinger.com.br/tutoriais/como-criar-banco-de-dados-mysql/>.

MILANI, A. **MySQL-guia do programador**. [S.l.]: Novatec Editora, 2007.

MOURA, A. I. **WBLS**: um sistema de localização de dispositivos móveis em redes wi-fi. 2007. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.

NODE-RED. Plataforma Node-Red. , [S.l.], 2019. Acessado em Agosto 2019. Disponível em: <https://nodered.org/>.

OLIVEIRA, R. R. Uso do microcontrolador ESP8266 para automação residencial. **Rio de Janeiro: UFRJ Escola Politécnica**, [S.l.], 2017.

PHPMYADMIN. phpMyAdmin. , [S.l.], 2013. Acessado em Agosto 2020. Disponível em: <https://www.phpmyadmin.net/.

PINTO, R. A. et al. **Sistemas eletrônicos para iluminação de exteriores empregando diodos emissores de luz (leds) alimentados pela rede elétrica e por baterias**. 2012. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Santa Maria.

ROSáRIO, V. A. do. Automação do controle e monitoramento de veículos de um estacionamento. , [S.l.], 2019. Acessado em Agosto 2021. Disponível em: <https://www.demdv.cefetmg.br/wp-content/uploads/sites/54/2019/04/Victor-Rosario.pdf>.

SANTOS, H. rede mesh. , [S.l.], 2009. Acessado em novembro 2020. Disponível em: <https://sites.google.com/site/redemesh/Home/projeto-tcc/>.

SOUSA, F. P. O. d. Análise de Viabilidade de Implantação de uma Rede Mesh sem Fios. , [S.l.], 2016.

SYSTEMS, E. ESP32 Series Datasheet. , [S.l.], 2019. Acessado em Agosto 2019. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32$_datasheet_en.pdf$ > .

TECHPARKING. Monitor. , [S.l.], 2017. Acessado em janeiro 2021. Disponível em: <http://www.techadvanced.com.br/site/techparking/>.

UFRGS. Junção P-N. , [S.l.], 2019. Acessado em Agosto 2019. Disponível em: <https://www.ufrgs.br/enfitecjunior/2018/04/19/diodo-emissor-de-luz-led/>.

APÊNDICES

APÊNDICE A – APÊNDICE A - Código completo utilizado no ESP32

```
#include <WiFi.h>
#include <WiFiMulti.h>
#include <Ultrasonic.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WebServer.h>
#include <WiFiManager.h>


// Número dos Pinos no ESP32

const int trigPin = 26;
const int echoPin = 27;
const int verde = 25;
const int vermelho = 33;

// Definição das variáveis

long duration;
int distance;
String IDVAGA = "???";
const uint16_t port = 1338;
const char * host = "192.168.137.1"; // IP ou DNS



// Setup

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

```
// Definição dos pinos de entrada e saída

  pinMode(trigPin, OUTPUT); // Set pino de saída do Trigger
  pinMode(verde, OUTPUT);
  pinMode(vermelho, OUTPUT);
  pinMode(echoPin, INPUT); // Set entrada do pino do Echo

  Serial.begin(115200);
  delay(10);

  // Conexão com a internet

  WiFiManager wifiManager;

  wifiManager.autoConnect("Config Vagas");

  Serial.println("Conectado");

  delay(1000);

  IDVAGA = getIDLegal();
  Serial.println("ID DA VAGA: " + IDVAGA);

}

String getIDLegal() {
  // Leitura do ChipID do ESP32
  char *aux;
  aux = (char *) malloc(sizeof(char) * 14);
  uint32_t chipid = ESP.getEfuseMac(); //The chip ID is essentially its
  sprintf(aux, "%04X", (uint16_t)(chipid >> 32)); //print High 2 bytes
```

```
  sprintf(aux, "%08X", (uint32_t)chipid);
  String macString(aux);

  return macString;
}

boolean vaga = false;

// Leitura do sensor ultrassônico

void loop() {

  // Baixar o sinal de entrada do Trigger

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Ajustar o pino de entrada do Trigger em nível HIGH por 10 microseg

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

// Leitura do pino echo, retornando o tempo da onda de som em microsegu

  duration = pulseIn(echoPin, HIGH);

// Calculo da distancia

  distance = duration * 0.034 / 2;

  byte i = 0;                    //  Variável para usar no for
```

```
  long acumHCSR = 0;              //  Variável para guardar o valor acu
  long  HCSR = 0;                 //  Variável para guardar o valor lid
  long HCSRM = 0;
  for ( i = 0; i < 100; i++)      // for de 5 vezes:  de 0 a 5 = 5 veze
  {
    HCSR = distance;      // le o valor ADC
    acumHCSR = acumHCSR + HCSR;   // Acumula 100 vezes HCSR lido
  }
  HCSRM = acumHCSR / 100;         //  calcula a ADC média das ultimas 1
  Serial.print("Distance: ");
  Serial.println(HCSRM);
  delay(2000);

  if (HCSRM > 100) {
    vaga = true;
    digitalWrite(verde, HIGH);
    digitalWrite(vermelho, LOW);
  }
  if (HCSRM < 100) {
    vaga = false;
    digitalWrite(vermelho, HIGH);
    digitalWrite(verde, LOW);
  }

  enviarStatusVaga();
  delay(1000);
}

void enviarStatusVaga() {
  WiFiClient client;

  if (!client.connect(host, port)) {
```

```
    Serial.println("Falha na conexão");
    Serial.println("esperando 5 segundos...");
    delay(2000);
    return;
  }

  client.print(IDVAGA);
  if (vaga == true)
  {

    client.print("#livre");
  }

  if (vaga == false)
  {
    client.print("#ocupado");
  }
  Serial.println("closing connection");
  client.stop();
}
```