UNIVERSIDADE DO ESTADO DO AMAZONAS

ESCOLA SUPERIOR DE TECNOLOGIA

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

KENNEDY MADSON FERREIRA DE AZEVEDO

DESENVOLVER UM SISTEMA DE MONITORAMENTO EM TEMPO REAL DE VAGAS DE ESTACIONAMENTO PRIVADO ATRAVÉS DE APLICATIVO ANDROID

MANAUS

2019

KENNEDY MADSON FERREIRA DE AZEVEDO

DESENVOLVER UM SISTEMA DE MONITORAMENTO EM TEMPO REAL DE VAGAS DE ESTACIONAMENTO PRIVADO ATRAVÉS DE APLICATIVO ANDROID

Projeto de Pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do Curso de Engenharia de Controle e Automação da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheira de Controle e Automação.

ORIENTADOR: JOSÉ RUBEN SICCHAR VILCHEZ, DR.

MANAUS

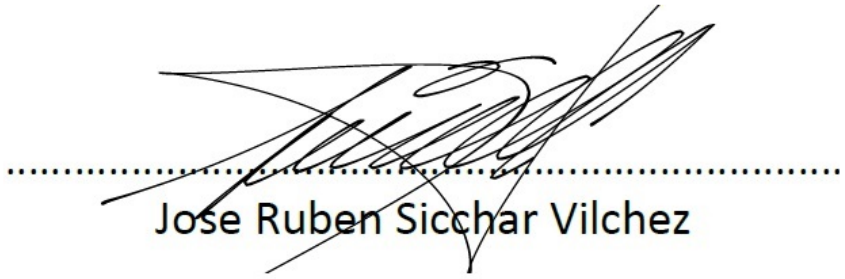
2019

KENNEDY MADSON FERREIRA DE AZEVEDO

DESENVOLVER UM SISTEMA DE MONITORAMENTO EM TEMPO REAL DE VAGAS DE ESTACIONAMENTO PRIVADO ATRAVÉS DE APLICATIVO ANDROID

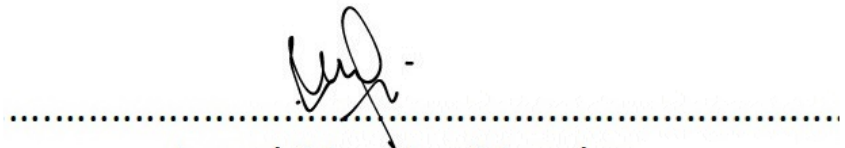
Projeto de Pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do Curso de Engenharia de Controle e Automação da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheira de Controle e Automação.

Orientador: José Ruben Sicchar Vilchez, Dr.



Jose Ruben Sicchar Vilchez

Moisés Pereira Bastos



Israel Mazaira Morales

Manaus

2019

AGRADECIMENTOS

Primeiramente, agradeço a Deus por ter me guiado até aqui e por ter me proporcionado adquirir um pouco de Sua sabedoria, pois tudo vem Dele.

Agradeço aos meus pais, Marcos e Alessandra, aos meus irmãos Sabrina e Vinícius e a minha namorada Emiliane Monteiro, por terem me dado todo o suporte necessário para concluir mais essa etapa da minha vida e os agradeço por isso.

Agradeço a todos os professores que colaboraram com o meu crescimento acadêmico e profissional, em especial aos professores: José Ruben Sicchar, meu orientador que teve bastante paciência e que não relevou as minhas falhas fazendo com que eu crescesse pessoalmente, ao Charles Melo, grande profissional e amigo, Marivan Gomes, o professor que me abriu portas do conhecimento e me ajudou para o crescimento profissional, ao professor Moisés Bastos, que foi bastante paciente por muitas vezes durante os TCC's, e ao Almir Kimura, que em suas aulas deu conselhos os quais foram muito importantes mim.

Também agradeço aos meus amigos que foram presentes e que contribuíram para a realização deste trabalho, e por isso agradeço a Rayssa Feitosa, Adriele Costa, Lucas Belido e Jailson Pereira.

RESUMO

A quantidade de carros circulantes, no Brasil, tem crescido ano após ano. E as maiorias das cidades não estão preparadas para receber essa grande frota de veículos, fato que gera inúmeros transtornos aos moradores, como: engarrafamentos, emissão de gases poluentes na atmosfera em grandes quantidades e a lotação de vagas em estacionamentos. Sendo este último, por não possuir sinalização adequada e disponibilização de informações a respeito das vagas do estacionamento, faz com que o usuário gaste vários minutos a procura de vagas. Pensando nisso, projetou-se um sistema para realizar o monitoramento inteligente das vagas o qual utiliza o ESP8266-01, que dispõe de tecnologia wifi, para receber e enviar as informações, coletadas pelos sensores de distância infravermelhos posicionados sobre as vagas, ao banco de dados utilizando um servidor web. E com isso, o usuário acessa essas informações através de um aplicativo android desenvolvido especificamente para este fim. A implementação e os testes do sistema foram realizados em uma representação miniaturizada de um estacionamento em maquete.

Palavras-chaves: Android Studio, ESP8266-01, Monitoramento Inteligente, LPC1768, Modelagem.

ABSTRACT

The amount of cars in Brazil has grown lately. And the majority of cities are not prepared to receive this large fleet of vehicles, a fact that generates innumerable inconveniences to the inhabitants, such as: traffic jams, polluting gases emission in the atmosphere in large quantities and lacking of parking spaces, which is justified by the it inadequate signaling and information available at the parking lots, and it reflects on a lot of time spent on looking for spots. With that in mind, it was designed a system to perform intelligent monitoring of parking spots using the ESP8266-01, which has Wi-Fi technology, to receive information collected by the infrared distance sensors positioned on the spots and send it to the bank through a web server. And by that, the user accesses this information through an android application specifically developed for that purpose. The implementation and testing of the system was performed in a miniaturized representation of a mock-up parking lot.

Key-words: Android Studio, ESP8266-01, Smart Monitoring, LPC1768, Modeling.

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

Segundo o (EMPRESÔMETRO, 2018) o que inclui: automóveis, comerciais leves, ônibus/micro-ônibus e caminhões, motocicletas/ciclomotores. Dados coletados pelo IBPT mostram que o número de veículos vem crescendo a cada ano, como é observado na Figura 1:

Figura 1 - Efetiva Frota Circulante

EFETIVA FROTA CIRCULANTE						
ANO	AUTOMÓVEIS	ÔNIBUS / MICRO-ÔNIBUS	MOTOCICLETAS / CICLOMOTOR	CAMINHÕES	COMERCIAIS LEVES	TOTAL
2010	30.826.816	297.462	11.116.478	1.485.225	4.739.579	48.465.559
2011	32.872.197	319.714	12.495.071	1.641.261	5.171.103	52.499.346
2012	35.048.017	335.583	13.367.437	1.747.324	5.602.506	56.100.868
2013	37.069.806	356.315	14.337.914	1.872.166	6.039.328	59.675.529
2014	38.847.587	371.210	14.813.790	1.968.428	6.476.101	62.477.116
2015	39.834.384	375.514	15.213.464	2.005.963	6.714.265	64.143.564
2016	40.377.833	375.302	15.290.268	2.016.726	6.885.910	64.946.039
2017	41.249.207	376.482	15.151.273	2.033.596	7.025.115	65.835.673

Fonte: (EMPRESÔMETRO, 2018).

Este crescimento efetuou-se até dezembro de 2015 com o fim da isenção do IPI (Imposto sobre Produtos Industrializados). Evento que fez com que o número de automóveis no país mais que dobrasse na última década. Dessa forma, o crescimento na quantidade de veículos gerou consequências negativas para a sociedade, como: o intenso tráfego de automóveis e pedestres, enormes engarrafamentos, e também a demora na procura por vagas em estacionamentos públicos e/ou privados, sendo este último, um dos problemas enfrentados constantemente por moradores de cidades com grande circulação de carros.

O tempo para alocar um veículo em um estacionamento, principalmente em horários de pico, com poucas vagas disponíveis à vista do motorista, causa transtorno e desconforto aos usuários, fazendo com que fiquem impacientes e percam minutos a procura de espaços disponíveis para guardar o carro, também, averígua-se por isso um consumo contínuo elevado do combustível do transporte automotor e emissão de gases poluentes no ambiente (Reichert, et al., 2011).

A aplicação de tecnologias referentes à automação de pequenos prédios se torna efetiva quando trabalha em conjunto com monitoramento remoto via dispositivo móvel, onde este é uma das ferramentas digitais mais utilizadas pelo ser humano. Segundo a Anatel (2016), o

Brasil terminou fevereiro de 2016 com 258,1 milhões de celulares e densidade de 125,62 celular/100 hab.

O projeto em questão propõe o desenvolvimento e a implementação de um sistema inteligente de monitoramento em tempo real de vagas de um estacionamento privado, com a transmissão dessas informações para o usuário através da internet para um dispositivo móvel (*smartphone*). As mesmas serão visualizadas através de um aplicativo Android, sendo que, será elaborado um protótipo em uma maquete que simulará um estacionamento composto de sete vagas.

1.1. TEMA

Desenvolver um sistema de monitoramento em tempo real de vagas de estacionamento através de aplicativo Android.

1.2. FORMULAÇÃO DO PROBLEMA

Atualmente, há formas de se encontrar vagas disponíveis para estacionamento através de sinais luminosos produzidas por leds (nas cores vermelho, verde e azul) posicionados sobre as vagas do estacionamento ou o uso de displays que informam o número de vagas que estão disponíveis para guardar o veículo. Estes métodos, no entanto, não suprem a necessidade dos motoristas em sua totalidade, pois ainda assim os mesmos necessitam procurar as vagas, orientados apenas por sinais luminosos.

Portanto, há a necessidade de se implementar um sistema com monitoramento inteligente, no qual o usuário obtenha as informações referentes aos estados das vagas do estacionamento monitorado por um aplicativo desenvolvido para celular com comunicação via internet. O monitoramento, desta forma, é em tempo real. Assim, ocorre a redução de tempo na procura por vagas livres, economia de combustível, evita possíveis transtornos e inconveniências e traz comodidade ao condutor. O motorista aloca seu veículo de forma rápida e precisa.

1.3. HIPÓTESE

É concebível a elaboração e a implementação de um sistema para monitoramento inteligente de vagas em um estacionamento privado utilizando sensores infravermelhos de distância, posicionados sobre as vagas, os quais coletam e o enviam os dados para o microcontrolador, que realiza o processamento das informações e, através de comandos AT, envia as informações para a rede local utilizando um módulo *wifi*.

1.4. OBJETIVOS

1.4.1. Geral

Projetar e implementar um sistema para monitoramento de vagas em um ambiente simulando um estacionamento com sensores posicionados em cada uma das vagas. O microcontrolador receberá e processará as informações das vagas coletadas pelos sensores e enviará o status das vagas para uma rede local, para que o usuário tenha acesso a essas informações através de um aplicativo Android.

1.4.2. Específicos

- Levantar informações de projetos que tenha relação com o desenvolvimento e implementação deste plano;
- Realizar estudos de componentes, ferramentas e softwares necessários para elaboração do sistema;
- Modelar o sistema em UML;
- Projetar e desenvolver o sensoriamento do estacionamento para coleta de dados;
- Elaborar uma rede de comunicação para transmissão das informações coletadas para a internet;
- Desenvolver aplicativo Android para incorporar no projeto;
- Apresentar o status das vagas para o usuário através da aplicação mobile.

1.5. JUSTIFICATIVA

A implementação do projeto contribuirá para a sociedade ao auxiliar o condutor do veículo na identificação de vagas livres, informando a localização exata por meio do aplicativo em seu celular, resultando na redução no tempo pela procura por vagas, economia de combustível, rapidez na alocação do veículo e otimização do fluxo de veículos no estacionamento.

Para o desenvolvimento do protótipo, serão utilizados sensores, microcontroladores, servidor web, programação em linguagem C e linguagem orientada a objetos e para isso faz-se necessário a utilização de conhecimentos adquiridos ao longo do curso de Engenharia de Controle e Automação, como nas disciplinas: Eletrônica Digital e Analógica, Circuitos Elétricos, Redes de Computadores, Instrumentação Industrial, Linguagem de Programação e Sistemas Microprocessados.

1.6. METODOLOGIA

Pretende-se no desenvolvimento do protótipo do referido trabalho, a aplicação dos conhecimentos adquiridos através de pesquisas em artigos acadêmicos, revistas científicas, disciplinas cursadas na universidade e experiências realizadas em laboratório. Nas quais, os métodos necessários para a realização da proposta estão divididos em: pesquisas de assuntos relacionados em artigos e revistas, sensoriamento, sistema embarcado, comunicação entre redes e desenvolvimento de aplicativos *Android*.

Inicialmente, serão feitas pesquisas em literaturas para adquirir conhecimentos que auxiliem no desenvolvimento e aplicação do trabalho para que o projeto tenha embasamento teórico que fortaleçam a ideia de que é executável a realização deste plano assegurando a qualidade acadêmica do trabalho. Com a obtenção do material bibliográfico, faz-se possível a seleção de componentes e ferramentas necessárias para execução prática do plano. Como a escolha de: sensores, microcontrolador, tipos de periféricos específicos para o projeto, circuitos integrados, *software* para desenvolvimento do *firmware* para embarcar no microcontrolador e construção do protótipo.

Após o levantamento de sensores de distâncias que poderão ser utilizados, serão realizados estudos para identificar o posicionamento mais adequado dos sensores nas vagas para que possam trabalhar da melhor maneira possível, com eficiência e sem falhas.

Para a leitura dos sensores, é utilizado um kit de desenvolvimento *mbed* baseado no LPC1768 que utiliza o microcontrolador *arm* córtex M3, que será responsável pelo processamento das informações e pelo envio delas para a *internet* através de um módulo *wifi*, chamado de ESP8266-01. Este processo dependerá de um algoritmo desenvolvido em Linguagem C/C++, que deve ser desenvolvido em uma IDE *online* fornecida pelo *site* do fabricante.

O módulo wifi será responsável por fazer a comunicação das placas desenvolvidas com a Rede Local. E por meio dele, o envio das informações pelos sensores para o banco de dados coletadas é realizado. Permitindo assim o monitoramento remoto das vagas do estacionamento. O tipo de comunicação utilizado para o tráfego de dados da placa eletrônica para o servidor é o protocolo TCP/IP.

Um serviço *Web* desenvolvido disponibiliza serviços para que o aplicativo desenvolvido na plataforma *Android* possa ter acesso aos dados coletados dos sensores de distância.

Para o desenvolvimento do aplicativo na plataforma Android, é necessário ter conhecimento em linguagem JAVA *Script* e ser familiarizado com o *software Android Studio*.

A aplicação mobile interfaceará a usabilidade do sistema com o homem, fazendo com que o usuário compreenda as informações do estacionamento de maneira fácil e intuitiva.

Ao final, deve-se construir um protótipo de um estacionamento para que se integre todos os componentes descritos acima para simular uma aplicação real.

1.7. APRESENTAÇÃO DO TRABALHO

Este trabalho está organizado em 6 capítulos, os quais abordam o trabalho de maneiras específicas referentes ao desenvolvimento do projeto. O capítulo 1 contém vários tópicos, e estão dispostos de maneira que expõem a ideia principal do projeto e explicam o motivo da execução do plano.

E no capítulo 2 são mencionados e citados 3 (três) trabalhos parecidos e que utilizam tecnologias aplicáveis ao trabalho em questão e serão utilizadas como modelo para execução do protótipo.

No capítulo 3 é apresentado o embasamento teórico das ferramentas a serem utilizadas no desenvolvimento do projeto. O embasamento é feito de acordo com a visão e experiência de autores que relatam o assunto inerentes ao utilizados no projeto.

Em seguida, no capítulo 4, é apresentado o diagrama de funcionamento do sistema, máquina de estados do firmware, arquitetura de hardware, materiais e métodos utilizados na elaboração do projeto.

No capítulo 5 serão apresentados os resultados, as conclusões e possíveis propostas de melhoramento do projeto.

E por fim, no capítulo 6, mostra as referências bibliográficas do trabalho.

2. TRABALHOS RELACIONADOS

2.1. MONITORAMENTO DE EQUIPAMENTOS ELÉTRICOS INDUSTRIAIS UTILIZANDO IOT.

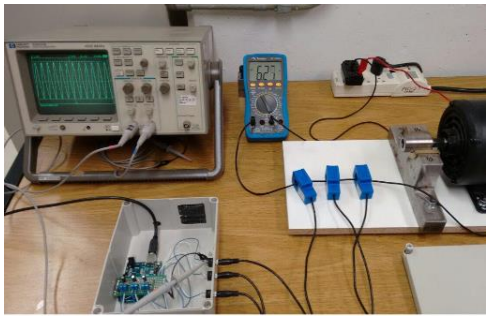
O trabalho descrito é uma dissertação de mestrado do Marcos Aurélio Fabrício da PUC de Campinas. De acordo com Fabrício (2018), o projeto em questão pretende realizar o monitoramento de corrente elétrica de uma máquina de uma linha de produção sem interferência humana. O processo citado era realizado manualmente, a Figura 2 mostra como a aferição era feita.

Figura 2 - Aferição Manual.

Fonte: (Fabrício, 2018).

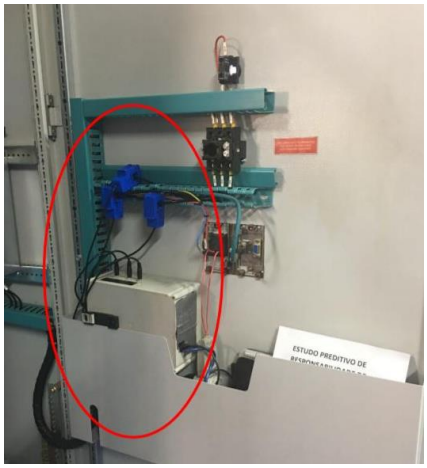
O projeto é constituído por sensores que enviam seus dados a um Arduino que possua vez se comunica com um computador, que conectado a Internet, envia as informações a um banco de dados. A figura da proposta e do resultado do projeto implementado é mostrado a seguir.

Figura 3 – Montagem do Projeto.



Fonte: (Fabrício, 2018).

Figura 4 - Implementação do Projeto.



Fonte: (Fabrício, 2018).

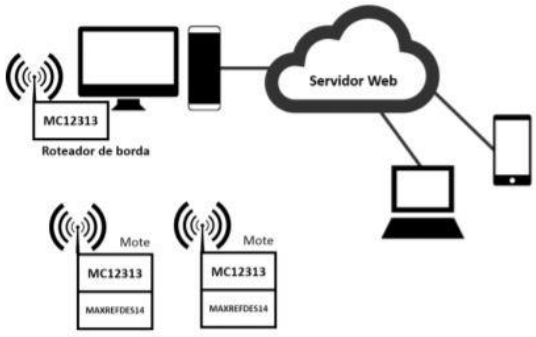
Os resultados se mostraram consistentes e demonstrou-se que o sistema de monitoramento proveu uma base para que uma aplicação recuperasse as informações geradas pelos sensores (FABRÍCIO, 2018).

2.2. PROPOSTA DE UM MEDIDOR DE CONSUMO UTILIZANDO TECNOLOGIA DE INTERNET DAS COISAS.

Segundo Cardozo e col. (2015), este projeto visa desenvolver um medidor de consumo de energia elétrica de aparelhos eletrônicos utilizando o princípio de IoT.

Foi desenvolvido um hardware capaz de realizar a leitura de corrente elétrica e enviar esses dados para uma rede local usando o wifi. As informações enviadas por esses dispositivos, serão armazenados no banco de dados fornecido pelo servidor web que também disponibilizará essas informações na Internet.

Figura 5 - Proposta do Projeto.



Fonte: (Cardoso, Renato, de Almeida, & Cunha, 2015)

O projeto em questão, ainda está em fase de testes e os resultados registrados eram preliminares. A condição de teste foi utilizada uma rede *Zigbee* para a comunicação entre PC, dois *motes* e o roteador de borda.

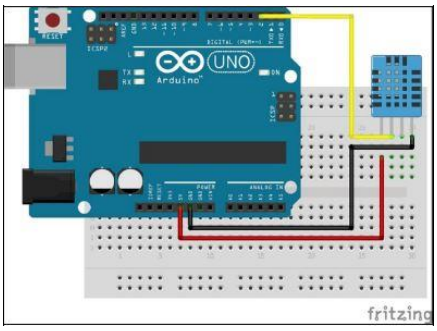
2.3. BANCO DE DADOS DISTRIBUÍDO PARA CONSULTA DE TEMPERATURA E UMIDADE UTILIZANDO ARDUINO E ANDROID.

Para o Scriptore e Júnior (2014), o objetivo deste trabalho é realizar a integração de duas ferramentas muito usadas nos dias atuais: o sistema operacional *android*, uma plataforma mobile com mais dispositivos no mercado e o Arduino, que é muito utilizado para realizar projetos de todos os tipos.

Para a concretização deste projeto, foi realizado pesquisar para embasar o trabalho e assim adquirir conhecimento para escolher as melhores ferramentas para auxiliar no desenvolvimento do projeto. Para realizar a leitura de temperatura e umidade do local estudado, foi escolhido o sensor DHT11. Este sensor será responsável por fazer o monitoramento em tempo real das variáveis escolhidas (temperatura e umidade) e os resultados serão enviados ao microcontrolador que se comunicará com o banco de dados armazenado nas nuvens pela internet via Ethernet Shield. Para que o usuário tenha acesso a estas informações em tempo real e de qualquer lugar do mundo, desenvolveu-se um aplicativo na plataforma android em que a pessoa consulte os dados em questão. A seguir serão apresentadas figuras que representam o sistema citado.

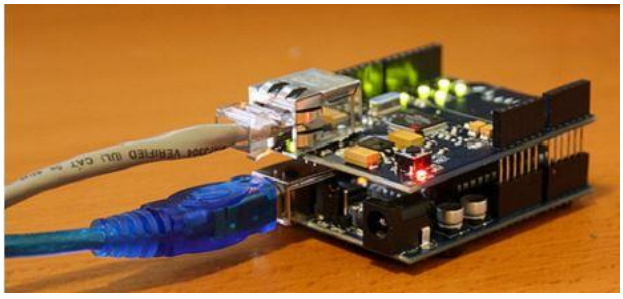
As propostas citadas no artigo foram cumpridas no qual culminou na comunicação entre os resultados levantados pelo sensor DTH11 e o usuário final, que monitorava a temperatura de umidade de um ambiente específico.

Figura 7 - Hardware Proposto.



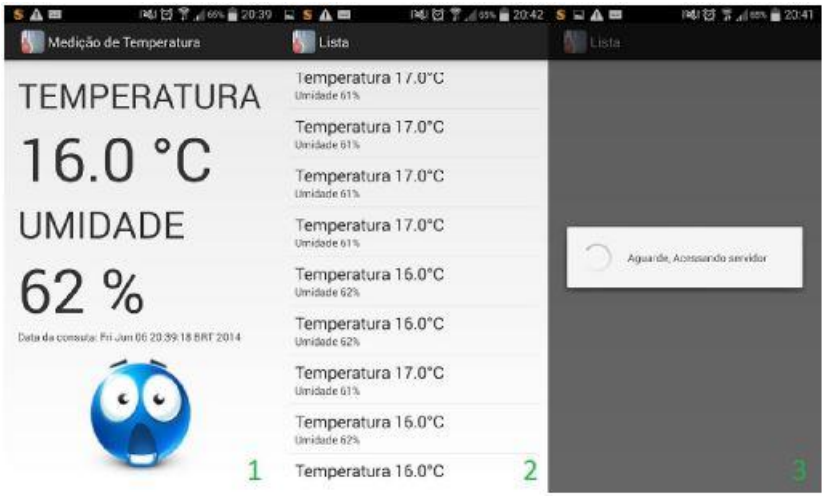
Fonte: (Scriptore & Júnior, 2014).

Figura 6 - Montagem do Hardware.



Fonte: (Scriptore & Júnior, 2014).

Figura 8 - Aplicativo para Leitura de Temperatura.



Fonte: (Scriptore & Júnior, 2014).

3. REFERENCIAL TEÓRICO

O desenvolvimento deste protótipo requer conhecimento específico em algumas áreas da engenharia, as quais englobam o tema proposto. Para isso realiza-se o embasamento teórico do trabalho e assim projetar e implementar o sistema em questão.

Os textos seguintes apresentarão a funcionalidade e aplicação de alguns elementos importantes para o funcionamento do protótipo.

3.1. SISTEMA EMBARCADO

Os sistemas embarcados referem-se a elementos eletrônicos que interagem com o ambiente por meio de sensores e atuadores os quais são controlados por conta de uma programação embutida (firmware) em um chip com orientações objetivas e específicas.

Segundo Otávio Chase (2007), os sistemas embarcados estão cada vez mais presentes na vida do homem. Eles são utilizados desde máquinas populares e menos sofisticadas, como lavadora de roupa, até as mais complexas como sistemas robotizados e autômatos, conforme vemos na figura 2. Esse crescimento se deu devido aos avanços das tecnologias, o barateamento de microcontroladores e a demanda de serviços variados. A Figura 9, vemos alguns exemplos.

Figura 9 - Dispositivos que possuem Sistema Embarcado.

Fonte:(Fernandes, 2018).

De acordo com Fernando Garcia (2018), o conceito de sistemas embarcados é um conjunto totalmente encapsulado e dedicado ao ambiente o qual deve-se controlar. É um sistema desenvolvido para realizar tarefas específicas e que foram pré-definidas no seu firmware (conjunto de instruções). O firmware é o software que está embarcado na memória flash do microcontrolador que compõe o sistema, o qual não está acessível ao usuário do produto. Ele é o responsável por realizar as tomadas de decisões, reagindo com atuadores em resposta aos dados lidos pelos sensores posicionados no ambiente.

Para a Equipe de Embarcados (2015), para desenvolver um sistema embarcado, o projetista precisa conhecer: o ambiente para o qual vai desenvolver o sistema, microcontroladores, linguagens de programação, sistemas digitais, noção de controle de processos, tecnologia de aquisição de dados, de atuadores e etc.

A característica essencial de um sistema embarcado é que eles são compostos basicamente por microcontrolador fixo em uma placa de circuito impresso e periféricos de entrada e saída que interagem com o meio o qual está embutido.

Por realizarem tarefas específicas, muitas vezes esses sistemas não têm flexibilidade (de software e de hardware) que lhes permita fazer outras tarefas quaisquer que não sejam aquelas para as quais foram projetadas e desenvolvidas.

3.1.1. Microcontrolador

De acordo com Marshall Brain (2014), um microcontrolador é um computador em escala menor. Pois as duas ferramentas possuem muitas características em comum, como por exemplo:

- CPU (unidade de processamento central);
- Memória;
- Portas de entradas e saídas;
- Timers;
- Contadores;
- Comunicação serial;
- Conversores analógico-digital, entre outros.

Os microcontroladores são dedicados ao sistema, ou seja, executam instruções de um programa específico, que está armazenado em sua memória ROM, para realizar tarefas às quais foram pré-programados.

As três principais diferenças do microcontrolador para o microprocessador são a arquitetura eletrônica e lógica, o tipo de aplicação e a frequência de operação. Na qual, os microcontroladores possuem elementos adicionais como: memória de leitura e escrita para armazenamento de dados, memória para gravar o firmware da aplicação, EEPROM e dispositivos periféricos como conversores ADC e DAC, portas de entrada e saída.

3.1.2. Periféricos

Periféricos são dispositivos que fazem a interface do microcontrolador com o mundo externo, os quais captam informações do meio para a unidade de processamento. Pode-se citar duas categorias de periféricos que são os sensores e atuadores. (Andre Luiz Delai,2013)

De acordo com André Luiz Delai (2013), os sensores são os responsáveis por realizar a aquisição de informações (entrada) do meio a ser controlado. Essas informações baseiam as tomadas de decisões da unidade de processamento. Na Figura 10, são mostrados alguns exemplos de sensores.

Figura 10 - Sensores.

Fonte: *AliExpress,* 2019.

Os atuadores interferem diretamente no ambiente que está envolvido, realizando ações para o processo em controle, como motores, ventiladores, luzes, aquecedores, resfriadores, chaveadores, etc. Esses são periféricos que enviam informação do Sistema Embarcado para o processo (Andre Luiz Delai,2013).

3.2. INTERNET DAS COISAS

A internet das coisas é considerada uma extensão da Internet a qual permite que dispositivos inteligentes com capacidade computacional e de comunicação troquem informações entre os elementos conectados na rede e assim compartilham serviços da Internet e permite que sejam monitorados e controlados remotamente por usuários. Fato que já traz diversos benefícios aos usuários. (Santos et al., 2016)

Para que o dispositivo possa ser considerado como um Smart Objects, ele deve apresentar em sua estrutura alguns requisitos mínimos de hardware, como: i) unidade(s) de processamento; ii) unidade(s) de memória; iii) unidade(s) de comunicação e; iv) unidade(s) de sensor(es) e/ou atuador(es). E estes objetos, ao estabelecerem comunicação com outros dispositivos, manifestam o conceito de estarem em rede. (Santos et al., 2016)

Conforme Santos et al., (2016), a IoT pode ser observada como um arranjo de diferentes tecnologias que se complementam no sentido de facilitar a integração dos dispositivos no ambiente físico ao mundo virtual. Os blocos básicos são:

Identificação: identifica o objeto para diferenciá-lo dos demais conectados na rede em que está atuando. Pode-se utilizar a tecnologia RFID, NFC e endereçamento IP.

Sensores/Atuadores: sensores responsáveis pelas coletas dos dados do ambiente ao qual está embutido e atuadores para manipular ou reagir ao meio de acordo com a programação embutida no chip microcontrolador.

Comunicação: está relacionado a tecnologias utilizadas para conectar os objetos inteligentes. Pode-se citar como exemplo: o *wifi*, *bluetooth*, *rfid*.

Computação: microcontrolador ou microprocessador com firmware embarcado o qual será responsável por tomar as decisões do processo.

Serviços: a IoT fornece diversos serviços, dentre eles, a identificação do sensor na rede, o armazenamento dos dados em nuvem ou data *warehouse* o qual pode possuir dados homogêneos e/ou heterogêneos obtidos dos outros dispositivos inteligentes.

Semântica: está relacionado à capacidade de extrair informações dos objetos inteligentes e utilizar essas informações para utilizá-las para prover outros serviços.

3.3. SERVIDOR WEB

Web Services é uma tecnologia utilizada para disponibilizar serviços interativos na Web ou Internet para que diversos sistema de plataformas diferentes, também podem ser chamados de sistemas heterogêneos, os quais utilizam o software para manipular e acessar as informações da rede. Alguns dos principais serviços oferecidos por esta tecnologia é a inserção das informações coletadas de cada sistema e requisição de informações de outros sistemas disponíveis a todas as aplicações (Dantas, 2007).

Klabunde (2007) sugere ainda que um servidor web é um computador que processa as requisições *Hyper Text Transfer Protocol* (HTTP) que é um dos protocolos muito utilizado na Internet. E afirma ainda que o servidor web mais utilizado no mundo é o Apache pois ele oferece segurança, excelente desempenho e possui compatibilidade com diversas plataformas.

3.3.1. Apache

Conforme Marcelo (2005), o servidor web Apache teve seu início em 1995 quando um grupo de desenvolvedores que trabalhavam na *Nacional Center of Supercomputing Applications* (NCSA), que naquela época possuía o *Web Server* mais popular do mundo o qual

utilizava o protocolo Http, ficaram insatisfeitos com a empresa pelo fato de não querer desenvolver mais projetos. Sentindo-se estagnados, esse pequeno grupo saiu da empresa e se juntou e começaram a criar uma série de inovações em cima do código do *Web Server* da NCSA. E hoje, eles formam a Fundação Apache que é responsável por manter o *Web Server* mais usado no mercado atualmente.

Uma das grandes vantagens citadas por Marcelo (2005) é de este software ser gratuito, ser de código aberto - o que permite que outros profissionais desenvolvam softwares que se adaptem a este servidor, e possuir suporte ao protocolo http. Outra característica que deve ser ressaltada é a citada por Ford (2008) é de que o Apache possui uma arquitetura modular para ser eficaz e portátil.

3.3.2. Protocolo HTTP

Conforme Jaime (2014), o HTTP é o protocolo de trocas de informações, o qual é responsável pela especificação da mensagem que o navegador pode enviar e nas respostas que eles receberão. Ou seja, o protocolo utiliza o modelo de comunicação baseado em *request-response* na qual o navegador estabelece uma conexão com o servidor que por sua vez responde a cada requisição solicitada pelo navegador.

As requisições deste protocolo devem ser formadas de acordo obedecendo algumas regras, como é descrita na Figura 11 (Fonte Kurose e Ross, 2010).

A linha de requisição deve ser composta pelo método utilizado, objeto requisitado e versão do protocolo. O método especifica a ação a ser tomada no recurso informado (Kurose e Ross, 2010).

De acordo com Ferreira (2017), os principais métodos do protocolo HTTP são:

GET: obter os dados de um recurso;

POST: criar um novo recurso;

PUT: substituir os dados de um determinado recurso;

PATCH: atualizar parcialmente um determinado recurso;

DELETE: excluir um determinado recurso;

HEAD: similar ao GET, porém é utilizado para obter o cabeçalho de resposta.

Quanto a linha de cabeçalho deve possuir informações adicionais da mensagem, abaixo podem ser observados alguns cabeçalhos:

User-Agent (solicitação): informações sobre o navegador e sua plataforma;

Accept (solicitação): o tipo de página que o cliente pode manipular;

Accept-Charset (solicitação): os conjuntos de caracteres aceitáveis para o cliente;

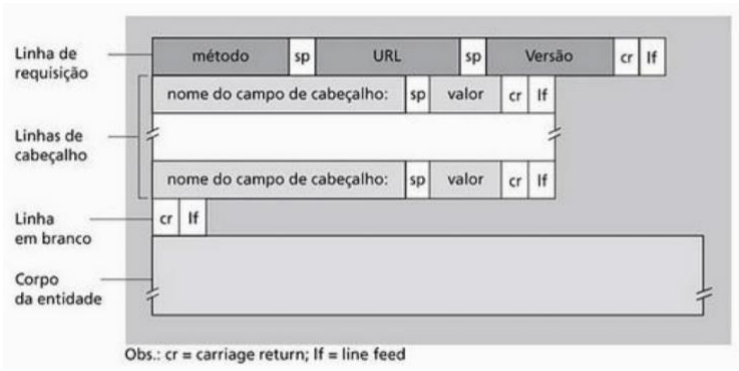
Accept-Language (solicitação): os idiomas com os quais os clientes podem lidar;

Content-Language (respota): o idioma utilizado na página;

Content-Length (respota): o comprimento da página em bytes;

Content-Type (respota): o tipo MIME da página.

Figura 11 - Estrutura da requisição HTTP.



Fonte: Caelum, 2017.

3.3.3. Arquitetura Cliente-Servidor

Conforme Hulquist (1997), as características do Cliente e do Servidor são descritas a seguir:

Cliente, que também pode ser chamado de "*front-end*" e "*workstation*": é um processo que interage com o usuário por meio de uma interface gráfica ou não, permitindo consultas ou comandos de recuperação de dados e análise. Além disso o cliente é um agente ativo nesta arquitetura pois ele inicia e termina a conversação com o servidor e ele não interage diretamente com outros cliente.

Servidor ou "*back-end*": fornece serviços e fica disponível para os seus clientes. Ele é um agente reativo na arquitetura, pois ele depende das requisições dos clientes e assim oferecer serviços solicitados.

3.4. BANCO DE DADOS

Conforme Thomé (2015), banco de dados (BD) é um projeto utilizado para armazenar grandes quantidades de informações. As informações armazenadas devem ter um significado efetivo que atenda um propósito específico (FRANÇA, 2011). Esses dados são organizados dentro desse sistema o que permite que essas informações sejam recuperadas de forma rápida e ágil. Para ter acesso e manipular essas informações, faz-se necessário o uso de linguagens padrões que é a SQL (Linguagem de Consulta Estruturada). Essa linguagem permite a inserção, busca, atualização, criação de tabelas de dados, entre outros.

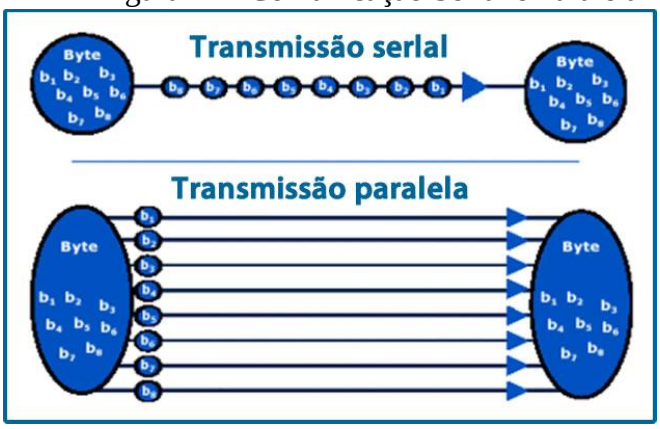
Conforme FRANÇA (2011), há especialistas que define banco de dados como um Sistema Gerenciador de Banco de Dados que é um pacote de softwares que permite a criação e manipulação do banco de dados.

3.5. PROTOCOLO DE COMUNICAÇÃO SERIAL

Conforme Stallings (2005), os protocolos de comunicação são conjuntos de regras e convenções que devem ser obedecidos para que o transporte e a troca de dados ocorram de maneira eficiente entre os dispositivos. Cada protocolo possui suas regras, limitações e características, o que permite ao desenvolvedor utilizar o protocolo mais adequado para a sua aplicação.

Há dois tipos de formatos que um protocolo de comunicação pode seguir, o serial e paralelo. A comunicação serial envia e recebe os dados da informação de maneira sequencial o que permite que o número de fios para a comunicação seja menor. Enquanto a comunicação paralela ocorre de forma a enviar e receber os dados por vários fios simultaneamente, sendo assim, este método transmitirá a informação com maior rapidez. A Figura 12 representa a forma como ocorre a troca de bytes.

Figura 12 - Comunicação Serial e Paralela



Fonte: http://www.bloghardwaremicrocamp.com.br/wp-content/uploads/2015/01/serial-01.jpg

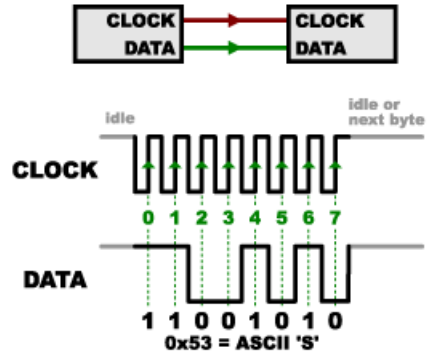
Apesar de a transmissão paralela possuir mais fios de comunicação, essa configuração não é mais rápida que a transmissão serial. Pois, para manter a sincronia e diminuir interferência (por ter muitos cabos) é necessário diminuir a frequência das paralelas o que diminui consideravelmente a sua velocidade (Lawrance, 2015).

Os protocolos possuem outras características, como: taxa de transmissão, método de comunicação e sentido de transmissão. A taxa de transmissão é a quantidade de bits que são transferidos por segundo e é representada por bps. O nome desta taxa pode mudar dependendo

da comunicação. Em caso de comunicação síncrona é chamada de clock e em comunicação assíncrona de "*Baud Rate*". É importante que durante a comunicação, os dispositivos trabalhem com a mesma taxa de transmissão (Schuncke, 2013).

A característica principal da comunicação serial síncrona depende do seu sinal de clock, isso quer dizer que a cada bit ou conjunto de bits enviados dependem do clock. A Figura 13 representa a estrutura desse tipo de comunicação.

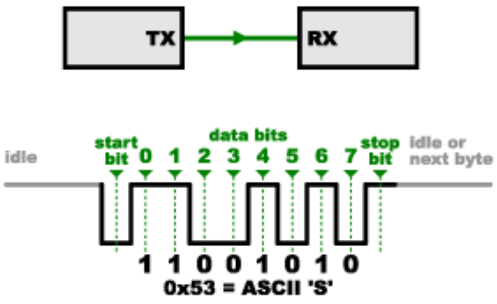
Figura 13 - Estrutura da Comunicação Síncrona.



Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi Acesso em ago. 2016.

Diferente do método acima, a comunicação serial assíncrona não precisa do sinal de clock para funcionar e, portanto, não se faz necessário o uso de um fio adicional, conforme é apresentado na Figura 14.

Figura 14 - Estrutura da Comunicação Assíncrona



Fonte: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi Acesso em ago. 2016.

O sentido de transmissão pode ser full duplex, onde os dados podem ser enviados e recebidos simultaneamente pelo barramento utilizado, o que reduz bastante o tempo de espera entre as transmissões. Tem-se também, o sentido de transmissão half duplex, é quando o dispositivo recebe ou envia dados e não executa essas funções ao mesmo tempo, e o temos

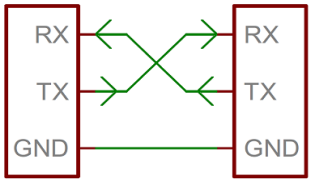
também o simplex que são dispositivos que se comunicam em apenas uma direção. A seguir serão apresentados dois protocolos: o I2C e UART.

3.5.1. UART

As vantagens de se utilizar esse tipo de comunicação está na simplicidade do protocolo e possui o sentido de transmissão de dados como o full duplex. Tornando a transmissão entre os dispositivos mais rápida.

Este tipo de comunicação utiliza o pino de transmissão (Tx) do protocolo o qual envia um pacote de bits que será interpretado bit a bit pelo pino receptor. Cada pacote enviado contém 1 start bit que indica o início da mensagem, 1 ou 2 stop bits para indicar o final da mensagem, 5 a 9 bits de informação e 1 bit de paridade para evitar a recepção de erros. A ligação é realizada conforme a Figura 15 abaixo:

Figura 15 - Estrutura de comunicação serial UART.



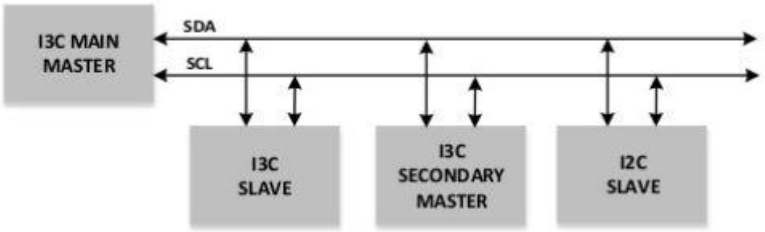
Fonte: https://learn.sparkfun.com/tutorials/serial-communication. Acesso em ago. 2016.

Este é um protocolo muito utilizado por diversos tipos de microcontroladores.

3.5.2. I²C

Essa comunicação foi desenvolvida pela Philips para realizar a comunicação entre periféricos. Este protocolo permite a conexão entre diversos dispositivos num mesmo barramento, além de ter a possibilidade de existir mais de um mestre controlando diversos escravos. O barramento em questão utiliza apenas 2 fios: o SDA (Serial Data) e SCL (Serial Clock). O pino SCL o caracteriza como comunicação síncrona e sua velocidade pode variar de 100kbit/s a 400 k bit/s. A Figura 16 caracteriza a comunicação I2C.

Figura 16 - Comunicação I²C

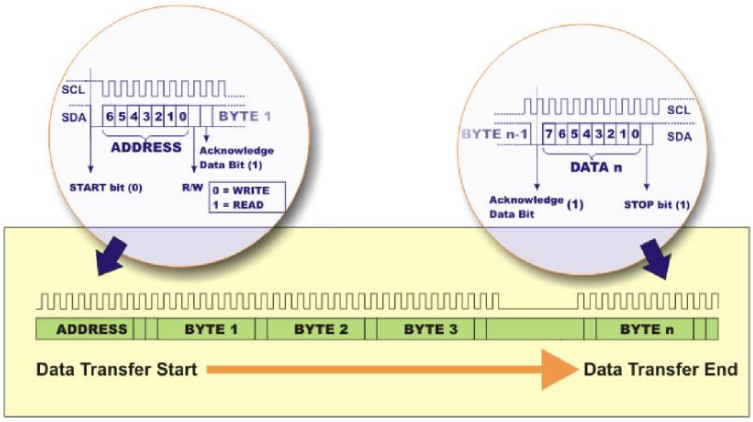


Fonte: Mikroe Embedded Tools.

Os elementos que compõem esse protocolo são chamados de mestre e escravo. O dispositivo mestre é responsável pela linha do clock e início da comunicação. O escravo possui um endereço físico único que os diferencia dos demais dispositivos conectados ao barramento.

Segundo a NXP, a comunicação inicia quando o mestre envia o start bit para o barramento. Nesta condição, os escravos estão prontos para receber a primeira informação que é o endereço do escravo o qual o mestre deseja comunicar. Depois que o byte de endereçamento foi enviado, caso haja dispositivo com este endereço, o escravo correspondente envia um ACK para confirmar que está apto para o recebimento das informações seguintes. A finalização da comunicação ocorrerá quando o mestre enviar um stop bit. A forma que ocorre a comunicação é representada na Figura 17.

Figura 17 - Transferência de dados do protocolo I2C.



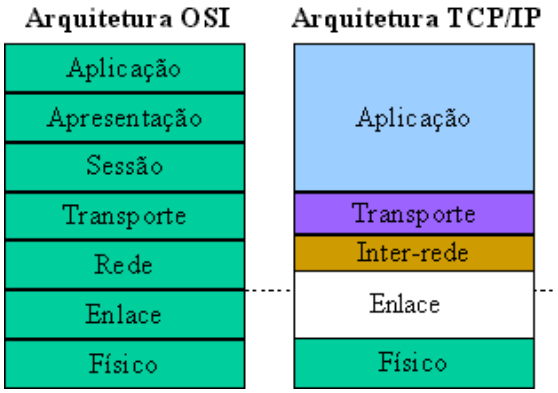
Fonte: Mikroe Embedded Tools.

3.6. PROTOCOLO TCP/IP

O TCP/IP (*Transmission Control Protocol/Internet Protocol*) é um protocolo que permite que equipamentos de uma rede possam comunicar entre si. O TCP/IP estabelece uma conexão fim a fim entre os usuários, isto significa o envio da mensagem com segurança entre o remetente e o destinatário.

Este protocolo pode ser analisado como um modelo em camadas similar à camada OSI, como observado na Figura 18, onde cada camada é responsável por realizar uma função específica que fornece um conjunto de serviços bem definidos.

Figura 18 - Arquitetura do Protocolo TCP/IP

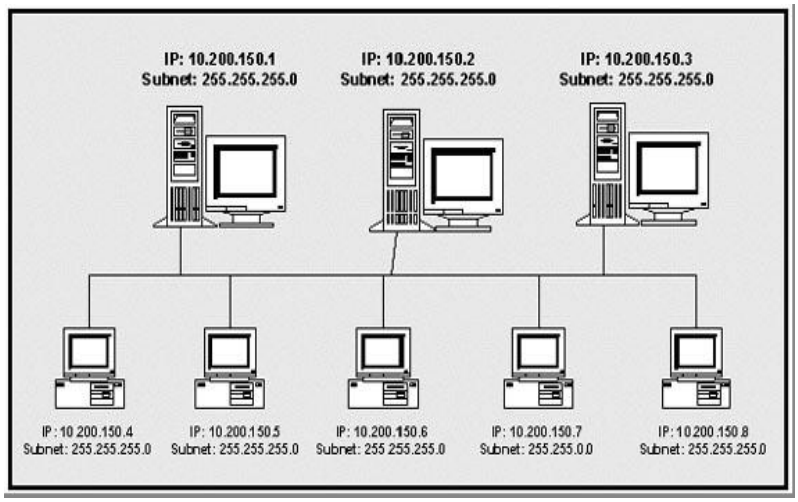


Fonte: http://www.gta.ufrj.br/grad/03_1/ip-security/paginas/introducao.html

Temos a camada de aplicação que é a camada mais alta e mais próximas do usuário. É a parte que lidam com dados mais abstratos. Abaixo da camada de aplicação, temos a camada de rede que é responsável por resolver o problema de se obter pacotes por meio de uma rede simples. A camada de enlace não faz parte do protocolo TCP/IP, mas é a maneira encontrada para passar quadros de rede de um dispositivo para a camada de internet de outro. A camada física trata das características elétricas e mecânicas do meio, como tipos de conectores e cabos usados para estabelecer uma comunicação (PORTAL EDUCAÇÃO, 2013).

Na Figura 19 temos um exemplo de uma pequena rede baseada no protocolo TCP/IP na qual está desconectada da Internet, e uma explicação básica de como funciona.

Figura 19 - Uma rede baseada no protocolo TCP/IP.



Fonte: http://www.linhadecodigo.com.br/artigo/134/tutorial-de-tcp_ip-parte-1.aspx

Segundo a visão de Battisti (2016), cada computador da rede precisa de, pelo menos, dois parâmetros configurados: número IP e máscara de sub-rede.

O Número IP é um número no seguinte formato: x.y.z.w, ou seja, são quatro números separados por ponto. Não podem existir duas máquinas, com o mesmo número IP, dentro da mesma rede. Caso seja configurado um novo equipamento com o mesmo número IP de uma máquina já existente, será gerado um conflito de Número IP e um dos equipamentos não conseguirá se comunicar com a rede.

Uma parte do Número IP (1, 2 ou 3 dos 4 números) é a identificação da rede, a outra parte é a identificação da máquina dentro da rede. O que define quantos dos quatro números fazem parte da identificação da rede e quantos fazem parte da identificação da máquina é a máscara de sub-rede (subnet mask). Vamos considerar o exemplo de um dos computadores da rede da Figura 12:

Número IP: 10.200.150.1

Subrede: 255.255.255.0

As três primeiras partes da máscara de sub-rede (subnet) iguais a 255 indicam que os três primeiros números representam a identificação da rede e o último número é a identificação do equipamento dentro da rede.

Quando a rede está isolada, ou seja, não está conectada à Internet ou a outras redes externas, através de links de comunicação de dados, apenas o número IP e a máscara de sub-rede são suficientes para que os computadores possam se comunicar e trocar informações.

3.7. APLICATIVO ANDROID

Segundo Tadewald (2014), conforme evoluía a capacidade de processamento de dados dos celulares a sua popularidade também crescia. Pois isto dava mais liberdade ao desenvolvedor de criar aplicações mais robustas e que são importantes hoje em dia. A competitividade das empresas que fornecem processadores e as que desenvolvem sistemas operacionais para esses dispositivos também cresceram entre si. Hoje as empresas dominantes neste ramo são: a ARM nos processadores e Google (através do Android) nos sistemas operacionais. O sistema operacional é o programa responsável por fazer com que todas as partes do smartphone funcionem e permite que todos os outros programas possam ser executados.

Esse sistema operacional já está bastante difundido. Muitos dispositivos como TV, tablets e smartphones, o tem como sistema operacional. O Android é desenvolvido tendo como base o sistema operacional Linux, conhecido por ser um sistema flexível, adaptável a várias arquiteturas de processador, seguro e eficiente. Um dos grandes diferenciais do Android e que

contribui para sua popularidade, é o fato de que o sistema é compatível com vários hardwares e pode estar disponível em smartphones de diversos fabricantes. Android é a plataforma ideal para usuários que não querem ficar restritos (ou dependentes) de uma única fabricante de hardware.

3.8. SQL – STRUCTURED QUERY LANGUAGE

Conforme Souza (2008), SQL é uma linguagem de pesquisa muito utilizada em banco de dados. Isso é decorrente de ser ela ser uma linguagem simples e fácil de usar. A seguir, serão citados dois de seus comandos, o UPDATE e o SELECT.

Com o comando SELECT é possível selecionar o registro que se quer recuperar em uma ou mais tabelas. A sua forma básica é:

SELECT <lista de atributos> FROM <lista de tabelas> WHERE <condições>;

Um exemplo citado por Souza (2008) para melhor entendimento é “Para selecionar o nome e o rg dos funcionários que trabalham no departamento 2 e que ganham mais do que R$2500,00 na tabela empregados, utiliza-se o seguinte comando: SELECT nome, rg FROM empregados WHERE departamento = 2 AND salário > 2500;”

O comando UPDATE permite realizar atualizações de valores dos componentes das tabelas. E a estrutura da requisição desse comando é:

UPDATE <tabela> SET <coluna>=<expressão> WHERE <condição>;

Para exemplificar o uso desta requisição, Souza (2008) citou a seguinte situação “Para atualizar o salário de todos os empregados que trabalham no departamento 2 para R$3000,00 o seguinte código é utilizado: UPDATE empregados SET salaário = 3000 WHERE departamento = 2”

3.9. JSON

O JSON, significa JavaScript Object Notation, é uma maneira leve de representar e trocar dados. JSON é formatado textualmente e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C, C++, C#, Java, JavaScript, Perl, Python e muitas outras. Essa propriedade faz com que o JSON seja um formato ideal para troca de dados. O JSON é constituído de duas estruturas: uma coleção de pares nome/valor, chamado de objeto, e uma lista ordenada de valores, os quais são caracterizados como vetores. (NUNES, 2014)

De acordo com Eduardo (2012), é possível a representação de mais de objeto em JSON e cita o exemplo a seguir:

```
[
  {
   “titulo”: “JSON x XML”,
   “resumo”: “o duelo de dois modelos de representação de informações”,
   “ano”: 2012,
   “genero”: [“aventura”, “ação”, “ficção”]
  },
  {
   “titulo”: “JSON James”,
   “resumo”: “a história de uma lenda do velho oeste”,
   “ano”: 2012,
   “genero”: [“western”]
  }
]
```

Os objetos são especificados entre chaves e podem ser compostos por vários pares nome/valor e os objetos estão entre colchetes e separados por vírgula. (EDUARDO, 2012)

3.10. MODELAGEM DE SISTEMAS

3.10.1. Linguagem de Modelagem Unificada

A UML ou *Unified Modeling Language* é uma linguagem visual usada para modelar e documentar softwares. Esta linguagem auxilia engenheiros de software a definir as características do seu sistema, tais como seus requisitos, comportamento, estrutura lógica, dinâmica de seus processos e até mesmo suas necessidades físicas em relação ao equipamento, tudo isso por meio de diagramas. (THANIA VARGAS, 2007)

Um diagrama é uma representação gráfica de um conjunto de elementos (classes, interfaces, colaboração, componentes, nós, etc) e é muito utilizado para enxergar a estrutura do software de diversas perspectivas. (THANIA VARGAS, 2007)

Na linguagem UML há diversos diagramas que podem ser utilizados pra representar a estrutura de um software e são elas: diagrama de classes, diagrama de objetos, diagrama de pacotes, diagrama de estrutura composta, diagrama de componentes, diagrama de implantação, diagramas de caso de uso, diagramas de sequência, diagrama de comunicação, diagrama de atividades, diagramas de visão de geral de integração, diagrama de temporização e diagrama de máquina de estados. Sendo o último utilizado neste projeto. (THANIA VARGAS, 2007)

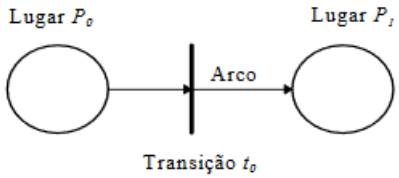
O diagrama de máquina de estados tem como principal elemento o estado. Esse tipo de diagrama informa o estado o qual o software pode estar ao longo de sua existência. E o elemento que muda o estado do software é chamado de transição que é o evento que ocorre durante o funcionamento do software (THANIA VARGAS, 2007).

3.10.2. Rede de Petri

Rede de Petri também é uma técnica de modelagem que permite a representação de sistemas e permite modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos (Rodovalho, 2012).

Esse tipo de técnica comporta dois tipos de componentes: um ativo chamado de transição e um passivo chamado de lugar. Os lugares equivalem ao estado do sistema e as transições de ações realizadas pelo sistema. Esses componentes se conectam através de arcos dirigidos. A Figura 20 mostra como os elementos citados se associam (Rodovalho, 2012).

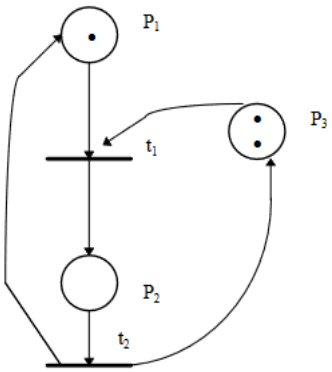
Figura 20 - Exemplo básico de Rede de Petri.



Fonte:(Rodovalho, 2012).

Os lugares podem ser “marcados” com tokens. Os tokens são informações atribuídas aos lugares para representar o estado do sistema em um determinado momento. Para simular o comportamento dinâmico dos sistemas onde a marcação da rede de petri é modificado a cada evento ocorrido. A Figura 21 representa uma rede marcada (Rodovalho, 2012).

Figura 21 - Rede de Petri Marcada.



Fonte: (Rodovalho, 2012).

A rede de petri pode seguir algumas estruturas, como: a sequencial, concorrente, paralelismo, rota alternativa e interativa. A estrutura sequencial é onde uma tarefa é executada após a outra e há dependência entre elas. A concorrente ocorre onde os lugares disputam uma mesma transição. No paralelismo, os eventos podem ocorrer simultaneamente e não há dependência da ocorrência do outro. A rota alternativa escolhe entre duas ou mais tarefas. A interativa é onde a mesma tarefa pode executar várias vezes (Rodovalho, 2012).

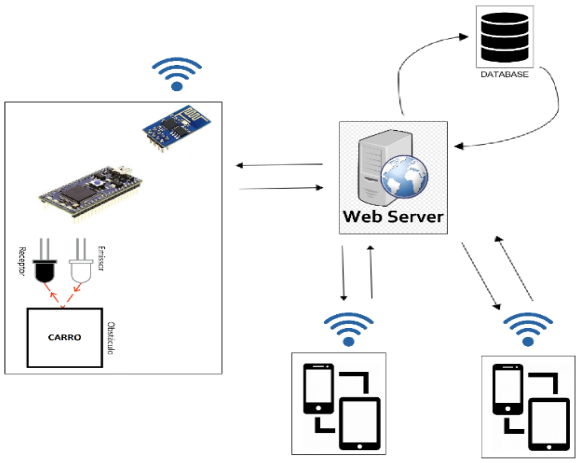
4. MATERIAIS E MÉTODOS

Para desenvolver o sistema de monitoramento inteligente de vagas para um estacionamento, projetou-se um hardware que será responsável por captar o status das vagas através de sensores infravermelho de distância que estarão situados sobre as mesmas, e enviar estas informações para o servidor usando um módulo wifi, o ESP8266-01. Além disso, compondo o sistema, foi desenvolvido também o firmware que é o responsável por realizar o controle e tomadas de decisão do processo. Para armazenar os dados enviados pelo hardware e disponibilizá-los para os demais dispositivos, um servidor web foi projetado. E uma aplicação Android foi criada para apresentar as informações a respeito das vagas para que o usuário final possa acessar esses dados. Para simulação de um estacionamento, projetou-se uma maquete para comportar carros em miniaturas. A seguir será descrito com mais detalhamento os materiais e métodos utilizados no desenvolvimento do sistema.

4.1. ARQUITETURA DO SISTEMA PROPOSTO

O sistema proposto é composto por Hardware, Firmware, Servidor Web e Aplicação Android. O hardware possui um microcontrolador que processa os sinais captados pelos sensores infravermelhos, tomadas de decisão e controle de leds para sinalização das vagas. O microcontrolador também se comunica com o módulo wifi o qual envia os dados para o servidor web. Quanto ao Servidor Web, é composto por um notebook que utiliza o sistema operacional Windows e comportará o software que disponibilizará serviços para os usuários. Além disso, há os usuários que utilizarão smartphones que serão os clientes do sistema. Na Figura 22, pode-se observar a arquitetura do sistema proposto:

Figura 22 - Arquitetura do Sistema



Fonte: Autor.

4.2. HARDWARE

Para desenvolver o hardware é necessário conhecer a aplicação e o ambiente o qual ele estará inserido, montar uma arquitetura de hardware e selecionar componentes para atender as necessidades do projeto.

Para definir os materiais, levantou-se alguns pontos principais, como: ambiente ao qual serão implementado o sistema, definição de sensores em relação a quantidade de vagas e quantidade de pinos do microcontrolador, o microcontrolador e o envio das informações para o servidor.

4.2.1. Sensor

A princípio, levantou-se a possibilidade de se utilizar o sensor de distância ultrassônico HC-SR04. Ele é capaz de detectar presença de objetos a uma distância de até 4 metros. Este sensor possui 4 pinos: 2 deles são de alimentação (5Volts e Ground) e outros pinos são o receptor de sinais, chamado ECHO e o pino gerador de ondas ultrassônicas chamado TRIGGER.

Figura 23 - Sensor de distância

Fonte: Filipeflop.

Para iniciar a medição da distância, deve-se acionar o pino trigger do sensor durante 10 microssegundos, desta maneira este dispositivo vai gerar ondas sonoras que ao encontrar um obstáculo, retornará ao módulo. Durante esse tempo de ida e volta das ondas sonoras, o pino de recepção (ECHO) permanecerá no nível lógico alto (1) e será contabilizado o tempo em que ele permaneceu nesse estado.

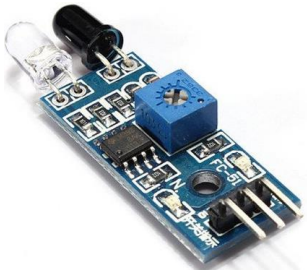
E levando-se em consideração que a velocidade do som é 340m/s, é possível realizar um cálculo matemático para se chegar à distância percorrida pela onda sonora e assim descobrir a distância do sensor para o obstáculo.

Este sensor foi descartado do projeto pois ele apresentou grandes desvantagens. Uma delas é que este sensor utiliza 2 pinos do microcontrolador: um para gerar o trigger e outro pra receber o sinal. Outra desvantagem é que é esse sensor demanda tempo de processamento do

microcontrolador para a leitura da distância. Esses dois fatos serão ainda mais agravados quando for feito a solicitação de leitura das vagas de todo o estacionamento.

Outra sugestão de sensor de distância para a simulação do sistema foi o infravermelho. E no mercado, há sensores de distância infravermelho bem baratos e que podem ser utilizados na maquete produzida para simulação. E o módulo escolhido foi o sensor de obstáculos reflexivo infravermelho e é apresentado na Figura 24.

Figura 24 - Sensor de Obstáculo Reflexivo



Fonte: Filipeflop.

O módulo em questão possui um led emissor de infravermelho, um fotodiodo responsável por detectar o sinal infravermelho, um trimpot e três pinos de conexão que são o 5Volts, Ground e o Sinal.

Quando o sensor está energizado e um obstáculo passa pelo seu ângulo de reflexão dentro da faixa de distância definida, ele muda seu sinal lógico de alto para baixo. E o ajuste da faixa de distância para detecção de obstáculos é feita através do trimpot do dispositivo e essa faixa pode variar de 2 a 30 centímetros.

As características citadas acima, são favoráveis a implantação deste dispositivo no projeto pois o tipo de saída é digital (alto ou baixo), pode ser utilizado em maquete devido a faixa de distância está dentro das dimensões da estrutura e por utilizar apenas um pino para sinal.

Em geral, estacionamentos comportam diversos veículos em seu interior. E cada vaga disponibilizada, para ser monitorada, deve possuir um sensor no teto. E para auxiliar microcontroladores na leitura do status das vagas utiliza-se o PCF8574.

4.2.2. PCF8574

De acordo com o datasheet disponibilizado pelo fabricante, o PCF8574 é um circuito integrado expansor de portas digitais que se comunica com dispositivos por meio do barramento I²C.

Conforme a NXP, este dispositivo consiste de 8 portas quasi-bidirecionais (pois podem ser definidos como entrada ou saída), três pinos para endereço no barramento e seus pinos operam com tensão entre 2.5 a 6 volts. No barramento I²C, estes dispositivos comportam-se como escravos. O dispositivo mestre no barramento, pode ler ou escrever nos pinos do PCF8574 por meio de registros enviados para o barramento. A Figura 25 mostra a pinagem deste CI.

Figura 25 - Pinagem do PCF8574.

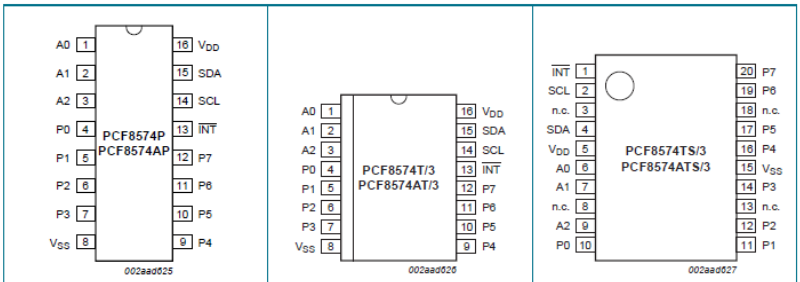


Fig 3. Pin configuration for DIP16 Fig 4. Pin configuration for SO16 Fig 5. Pin configuration for SSOP20

Fonte: Datasheet NXP

A Figura 26 mostra a descrição dos pinos. Este CI disponibiliza o pino de interrupção que é ativado quando o PCF8574 está configurado para o modo de leitura e gera pulso quando percebe uma alteração na leitura dos seus pinos, uma informação muito importante pra quem desenvolve hardware e firmware.

Figura 26 - Descrição dos pinos.

Table 3. Pin description

Symbol	Pin		Description
	DIP16, SO16	SSOP20	
A0	1	6	address input 0
A1	2	7	address input 1
A2	3	9	address input 2
P0	4	10	quasi-bidirectional I/O 0
P1	5	11	quasi-bidirectional I/O 1
P2	6	12	quasi-bidirectional I/O 2
P3	7	14	quasi-bidirectional I/O 3
V_{SS}	8	15	supply ground
P4	9	16	quasi-bidirectional I/O 4
P5	10	17	quasi-bidirectional I/O 5
P6	11	19	quasi-bidirectional I/O 6
P7	12	20	quasi-bidirectional I/O 7
$\overline{INT}$	13	1	interrupt output (active LOW)
SCL	14	2	serial clock line
SDA	15	4	serial data line
V_{DD}	16	5	supply voltage
n.c.	-	3, 8, 13, 18	not connected

Fonte: Datasheet NXP.

Outra informação importante é de que a fabricante disponibiliza no mercado o PCF8574 e o PCF8574A. Ambos possuem a mesma função, porém o range de endereços o diferenciam, o que permite conectar um número maior de expansor de portas no circuito e, portanto, um

número maior de entradas e saídas podendo chegar a 128. O formato do endereçamento dos PCF8574's é apresentado na Figura 27 abaixo.

Figura 27 - Formato do endereço do PCF8574 E PCF8574A.

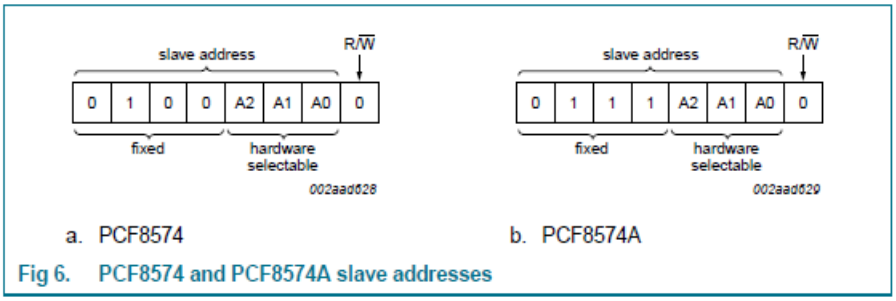


Fig 6. PCF8574 and PCF8574A slave addresses

Fonte: Datasheet NXP.

O mapeamento do endereçamento dos PCF's é mostrado nas Figura 28 e Figura 29.

Figura 29 - Mapeamento do Endereço do PCF8574.

Table 4. PCF8574 address map

Pin connectivity			Address of PCF8574								Address byte value		7-bit hexadecimal address without R/W
A2	A1	A0	A6	A5	A4	A3	A2	A1	A0	R/W	Write	Read	
V_{SS}	V_{SS}	V_{SS}	0	1	0	0	0	0	0	-	40h	41h	20h
V_{SS}	V_{SS}	V_{DD}	0	1	0	0	0	0	1	-	42h	43h	21h
V_{SS}	V_{DD}	V_{SS}	0	1	0	0	0	1	0	-	44h	45h	22h
V_{SS}	V_{DD}	V_{DD}	0	1	0	0	0	1	1	-	46h	47h	23h
V_{DD}	V_{SS}	V_{SS}	0	1	0	0	1	0	0	-	48h	49h	24h
V_{DD}	V_{SS}	V_{DD}	0	1	0	0	1	0	1	-	4Ah	4Bh	25h
V_{DD}	V_{DD}	V_{SS}	0	1	0	0	1	1	0	-	4Ch	4Dh	26h
V_{DD}	V_{DD}	V_{DD}	0	1	0	0	1	1	1	-	4Eh	4Fh	27h

Fonte: Datasheet NXP.

Figura 28 - Mapeamento do Endereço do PCF8574A.

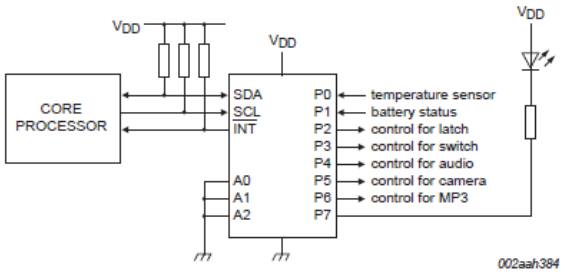
Table 5. PCF8574A address map

Pin connectivity			Address of PCF8574A								Address byte value		7-bit hexadecimal address without R/W
A2	A1	A0	A6	A5	A4	A3	A2	A1	A0	R/W	Write	Read	
V_{SS}	V_{SS}	V_{SS}	0	1	1	1	0	0	0	-	70h	71h	38h
V_{SS}	V_{SS}	V_{DD}	0	1	1	1	0	0	1	-	72h	73h	39h
V_{SS}	V_{DD}	V_{SS}	0	1	1	1	0	1	0	-	74h	75h	3Ah
V_{SS}	V_{DD}	V_{DD}	0	1	1	1	0	1	1	-	76h	77h	3Bh
V_{DD}	V_{SS}	V_{SS}	0	1	1	1	1	0	0	-	78h	79h	3Ch
V_{DD}	V_{SS}	V_{DD}	0	1	1	1	1	0	1	-	7Ah	7Bh	3Dh
V_{DD}	V_{DD}	V_{SS}	0	1	1	1	1	1	0	-	7Ch	7Dh	3Eh
V_{DD}	V_{DD}	V_{DD}	0	1	1	1	1	1	1	-	7Eh	7Fh	3Fh

Fonte: Datasheet NXP.

O datasheet do fabricante sugere, também, como deve ser usado o PCF no circuito:

Figura 30 - Aplicação do PCF8574.

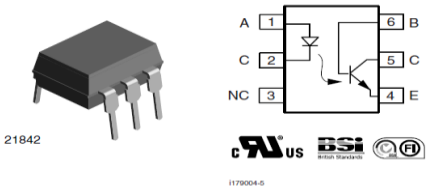


Fonte: Datasheet NXP.

4.2.3. OPTOACOPLADOR 4N35

A conexão de vários dispositivos com funcionalidades específicas é essencial para o sucesso do circuito. Porém pode haver alguns problemas em alguns dos dispositivos, como: sobrecarga, curtos e interferências. Pensando nisso, optou-se por colocar um circuito integrado que isola circuitos protegendo os demais componentes do circuito. Uma solução de proteção é o uso de optoacopladores. Esses chip's possuem em seu interior um diodo led infravermelho e um transistor photoacoplador, componentes que permitem o isolamento do circuito pois a transmissão do bit é feita por luz. A Figura 31 mostra a estrutura interna deste componente.

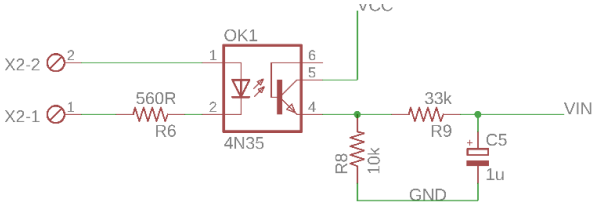
Figura 31 - Estrutura 4N35.



Fonte: Datasheet da Vishay Semiconductors.

A função dele, neste projeto, é isolar o circuito de alimentação e sinal do sensor para a o PCF8574. Do lado esquerdo e no pino 1 deste CI, será conectado o sinal do sensor que quando mudar seu estado lógico, acionará ou desligará o led infravermelho que realiza o chaveamento do fototransistor. A Figura 32 mostra um exemplo de aplicação desse chip.

Figura 32 - Exemplo de aplicação do 4N35.



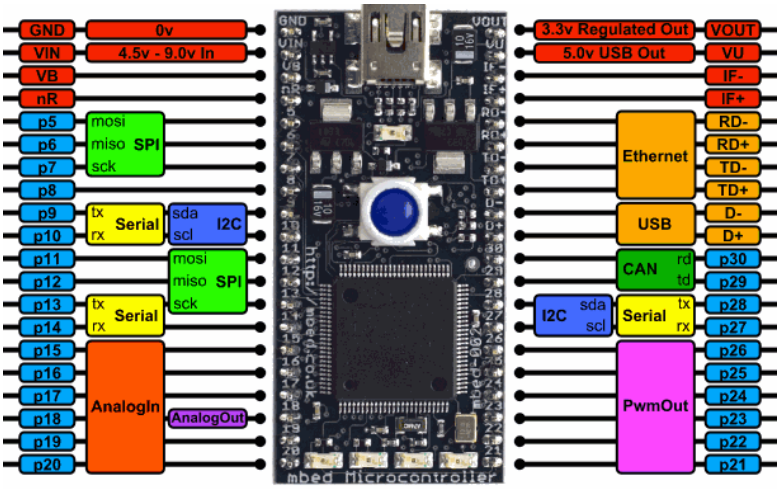
Fonte: CONTROL-LAB DOCS, 2018.

4.2.4. LPC1768

O microcontrolador é um dos componentes essenciais quando se fala de sistema embarcado e automação. Ele é responsável por fornecer inteligência, por tomar decisões com base nas informações lidas pelos sensores e por controlar atuadores para interagir com o ambiente ao qual está inserido.

E para o desenvolvimento deste projeto, será utilizado o kit de desenvolvimento LPC1768, fabricado pela NXP, que fornece periféricos prontos para serem usados, como: Ethernet, USB, interfaces de comunicação UART, SPI, I²C, ADC, PWM entre outros. Essas características são mostradas na Figura 33.

Figura 33 - Pinagem do LPC1768.



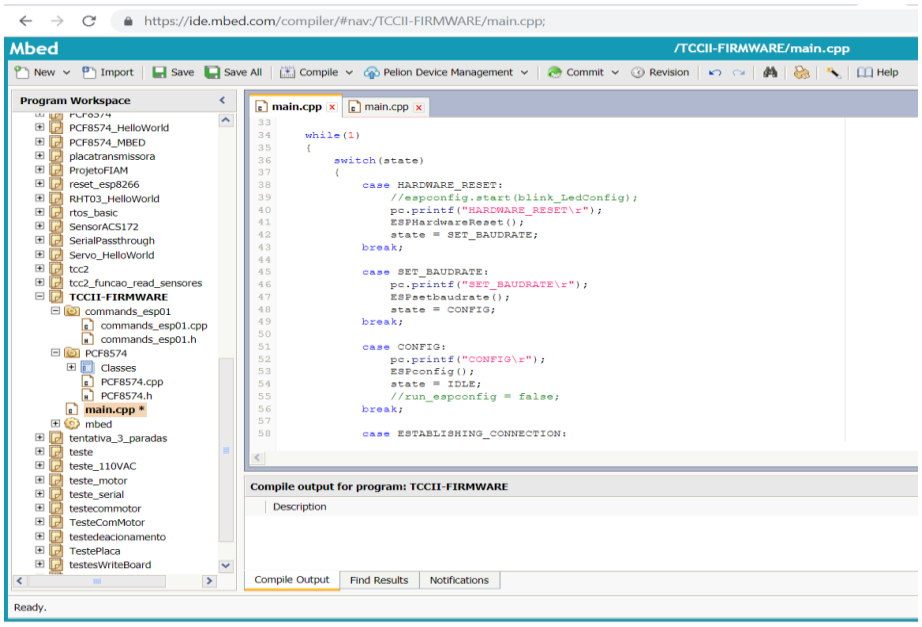
Fonte: MBED LPC1768, 2018.

Uma grande vantagem de se utilizar este microcontrolador nesta aplicação, é o fato de ele possuir dois barramentos I²C. Aumentando ainda mais a quantidade de expansores de portas que se possam conectar.

Além disso, a fabricante fornece também uma IDE online gratuita, chamada de MBED COMPILER, que é acessada por navegador web. Este software é baseado nas linguagens C e C++ e disponibiliza diversas bibliotecas básicas para controle dos seus periféricos. Esta IDE permite programar, compilar o código desenvolvido e gerar o binário para embarcar no microcontrolador. Pelo fato de ser um software da web é compatível com diversos navegadores e permite ser trabalhado de qualquer sistema operacional.

O compilador utilizado para gerar o binário é o ARMCC profissional. Possui, também, uma aba, chamada de workstation, que mostra os trabalhos desenvolvidos pelo usuário. O software permite a publicação dos códigos e também de exportar arquivos desenvolvidos e disponibilizados por outros usuários. A interface gráfica da IDE é mostrada na Figura 34.

Figura 34 - Interface Gráfica da IDE.



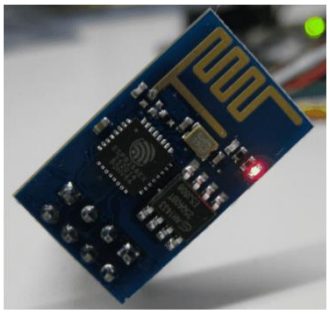
Fonte: Autor.

4.2.5. ESP8266-01

Como observado acima, o kit de desenvolvimento possui interfaces de comunicação como a Ethernet, UART e entre outros, e podem ser soluções para comunicar com um computador e assim enviar os dados para o servidor. Porém, para a aplicação real, torna-se inviável utilizar cabeamentos, pois certamente haverá muitas vagas no interior do estacionamento, fazendo-se necessário o uso de vários microcontroladores e o PC não teria suporte para todos eles. Então a solução encontrada foi utilizar um módulo wifi que seja capaz de conectar o microcontrolador a rede.

Segundo Kolban (2015), o ESP8266 é um microcontrolador, projetado pela Espressif System, que se pode definir como soluções de redes Wifi autossuficientes, podendo fornecer meio de integrar um microcontrolador a uma rede local e até a Internet. Uma das vantagens de

Figura 35 - ESP8266-01



Fonte: Filipeflop.

se utilizar esse tipo de dispositivo é o seu baixo custo. Esses fatores contribuem para que este módulo seja bastante utilizado em projetos de automação (Manhães, 2018). A tabela 1 apresenta características importantes do ESP8266-01.

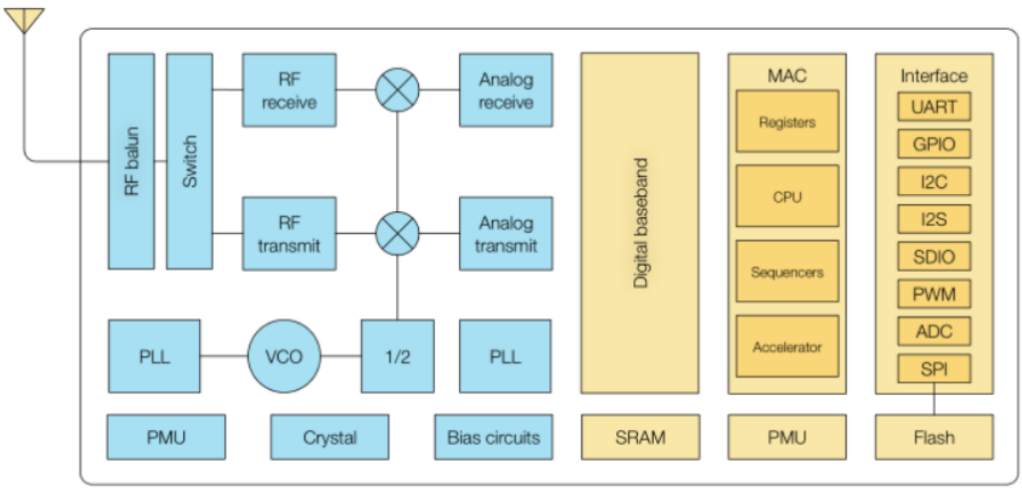
Tabela 1 - Especificações ESP8266-01

ESPECIFICAÇÕES	
Tensão de Operação	3V3
Consumo de energia	10mA – 70mA
Velocidade do Processador	80 – 160 MHz
Suporte aos protocolos 802.11	/b/g/n/d/e/i/k/r
Quantidade máxima de conexões TCP/IP	5

Fonte: Elaborado pelo Autor.

Conforme os documentos disponibilizados pela fabricante, a respeito do módulo wifi, a Figura 36 apresenta a arquitetura interna do chip.

Figura 36 - Arquitetura Interna do ESP8266.



Fonte: Ai-Thinker.

Conforme a Figura 36, observa-se um chip que disponibiliza diversas interfaces de comunicação. E uma das mais importante para a implementação, é a UART.

Para comunicar com o ESP8266 através da serial UART utiliza-se os pinos RX e TX e como forma de comunicação utiliza-se comandos AT. A seguir serão apresentados alguns comandos importantes para que o ESP8266 se conecte como um cliente TCP na rede local. A sequência de comandos apresentadas a seguir seguem o modelo fornecido pela fabricante. E são eles:

AT+CWMODE: usado para configurar o módulo como um ponto de acesso e cliente em uma rede local. Sendo o último, de suma importância para o projeto.

Retorno: OK.

AT+CWJAP=”SSID”,”PASSWORD”: comando para conectar o módulo wifi na rede local.

Retorno: OK.

AT+CIFSR: comando para requisitar o IP do módulo wifi na rede local.

Figura 37 – Resposta ao Comando

Response:

```
+CIFSR:APIP,"192.168.4.1"
+CIFSR:APMAC,"1a:fe:34:a5:8d:c6"
+CIFSR:STAIP,"192.168.3.133"
+CIFSR:STAMAC,"18:fe:34:a5:8d:c6"
OK
```

Fonte: Espressif System.

AT+CIPSTART=”TCP”,”ip_do_servidor”,porta: comando para conectar o módulo wifi no servidor e o IP informado é a do máquina que o servidor do sistema está instalado e o número depois da vírgula é a porta.

Retorno: OK.

AT+CIPSEND: Comando utilizado para enviar dados para o servidor. Exemplo de como utilizar o comando é mostrado na Figura 38.

Figura 38 – Enviar dados para o servidor

```
AT+CIPSEND=4    // set date length which will be sent,  such as 4 bytes
>test           // enter the data,  no CR
```

Response:

```
Recv 4 bytes
SEND OK
```

Fonte: Espressif System.

Logo após a confirmação, a resposta é recebida conforme a Figura 39.

Figura 39 – Receber dados do servidor.

7. When ESP8266 received data from server, it will prompt message below:

```
+IPD,n:xxxxxxxxxx    // received n bytes,  data=xxxxxxxxxx
```

Fonte: Espressif System.

AT+CIPCLOSE: comando para encerrar comunicação com o servidor.

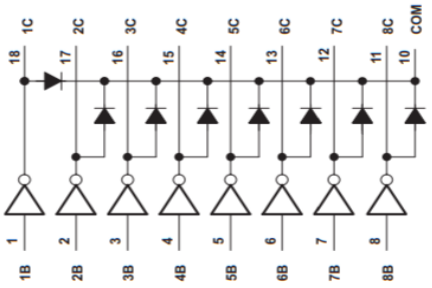
Retorno: CLOSED

OK

4.2.6. ULN2308A

Conforme a Texas Instruments (2017), este chip é um arranjo de transistores Darlington de alta voltagem. Ele permite que dispositivos de potência, como relés e motores de passo, sejam acionados com baixa tensão em seus pinos de entrada. Uma das grandes vantagens é que ele suporta cargas de até 50V e uma desvantagem é que o limite de corrente é de 500mA (Macedo, 2010).

Figura 40 – Arquitetura interna do ULN2803A.

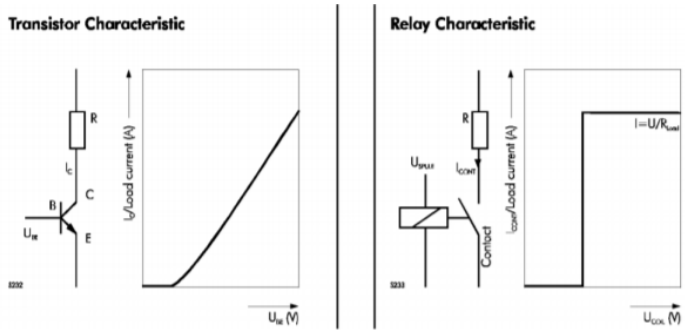


Fonte: Texas Instruments.

4.2.7. RELÉS

Conforme Pedro Abelha (2014), os relés são dispositivos comutadores eletromecânicos que abre ou fechar circuitos de cargas elevadas. O elemento comutador interno ao relé, muda a posição, de aberto para fechado, com a aplicação de energia elétrica em sua bobina. Ao contrário do transistor, o relé não é um amplificador analógico, mas sim um amplificador digital. A comparação do comportamento dos dois dispositivos é observada na Figura 41.

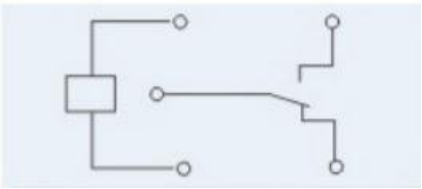
Figura 41 - Comparação entre Transistor e Relé.



Fonte: Pedro Abelha - Capítulo 2

O relé definido para o projeto é o JQC-3F(T73). Ele é um dispositivo que possui cinco pinos, sendo que dois são para o controle do acionamento que são os terminais que energizam a bobina, enquanto os outros três são para serem inseridos em um circuito secundário que em nosso caso é para chavear o ground entre os pinos da placa do led. Na Figura 42 observa-se a estrutura interna deste relé:

Figura 42 – Estrutura Interna do Relé.



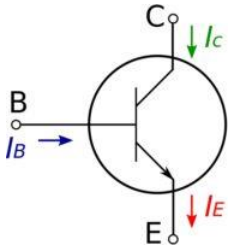
Fonte: Datasheet do Fabricante.

4.2.8. TRANSISTOR BC547

O transistor é um componente semicondutor que pode exercer diversas funções: amplificador de sinal, comutador de circuitos e regulador de corrente. (MATIAS, 2013)

Uma das formas mais simples de utilizá-lo é como chave, como o transistor operando no modo saturação e corte. O símbolo que representa o transistor é:

Figura 43 - Símbolo do Transistor.



Fonte: http://www.josematias.pt/eletr/o-que-sao-transistores/

Modos de operação do transistor:

Saturação: o transistor atua como interruptor. Neste modo a corrente flui do coletor ao emissor, tornando-se um curto circuito.

Corte: também atua como interruptor, porém não flui corrente do coletor pro emissor, ou seja, é um circuito aberto.

Em transistores, a corrente de base define o modo de operação do dispositivo. Se Ib for 0, a corrente do coletor é próxima de 0 e o transistor está no modo de corte. E na saturação, a corrente de base, quando próxima da corrente de saturação, a corrente no coletor é máxima e o transistor entra no modo de saturação.

Para projetar circuitos com transistores deve-se conhecer o seu ganho que é disponibilizado em seus datasheets. A Figura 44 apresenta dados a respeito do BC547.

Figura 44 - Características Elétricas do BC547

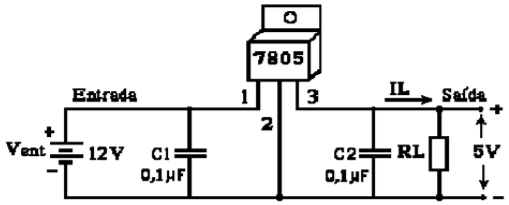
Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
I_{CBO}	Collector Cut-off Current	V_{CB}=30V, I_E=0			15	nA
h_{FE}	DC Current Gain	V_{CE}=5V, I_C=2mA	110		800	
V_{CE} (sat)	Collector-Emitter Saturation Voltage	I_C=10mA, I_B=0.5mA I_C=100mA, I_B=5mA		90 200	250 600	mV mV
V_{BE} (sat)	Base-Emitter Saturation Voltage	I_C=10mA, I_B=0.5mA I_C=100mA, I_B=5mA		700 900		mV mV
V_{BE} (on)	Base-Emitter On Voltage	V_{CE}=5V, I_C=2mA V_{CE}=5V, I_C=10mA	580	660	700 720	mV mV
f_T	Current Gain Bandwidth Product	V_{CE}=5V, I_C=10mA, f=100MHz		300		MHz
C_{ob}	Output Capacitance	V_{CB}=10V, I_E=0, f=1MHz		3.5	6	pF
C_{ib}	Input Capacitance	V_{EB}=0.5V, I_C=0, f=1MHz		9		pF
NF	Noise Figure : BC546/547/548 : BC549/550 : BC549 : BC550	V_{CE}=5V, I_C=200μA f=1KHz, R_G=2KΩ V_{CE}=5V, I_C=200μA R_G=2KΩ, f=30~15000MHz		2 1.2 1.4 1.4	10 4 4 3	dB dB dB dB

Fonte: Fairchild Semiconductor

4.2.9. REGULADORES DE TENSÃO

De acordo com Wendling (2009), os reguladores de tensão têm a função de manter a tensão de saída constante mesmo que ocorra variação de tensão de entrada. A seguir é apresentado uma aplicação para onde deseja-se regular a tensão fornecida de 12 volts para 5 Volts.

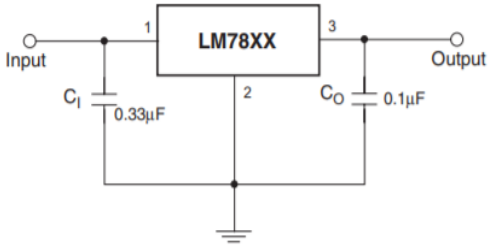
Figura 45 - Exemplo de Aplicação de um Regulador.



Fonte: CI Reguladores de Tensão, 2009.

Há diversos reguladores de tensão no mercado. A seguir, serão apresentados a aplicação de dois reguladores: o 7805 que regula a tensão para 5 volts e o LD33V que regula para 3,3 volts. Características elétricas dos reguladores:

Figura 46 - Aplicação do Regulador de Tensão



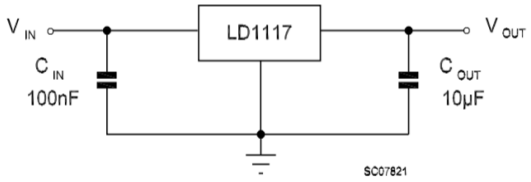
Fonte: Farchild Semiconductor,

Figura 47 – Características Elétricas do 7805CV.

Symbol	Parameter		Value	Unit
V_I	Input Voltage	V_O = 5V to 18V	35	V
		V_O = 24V	40	V
$R_{\theta JC}$	Thermal Resistance Junction-Cases (TO-220)		5	°C/W
$R_{\theta JA}$	Thermal Resistance Junction-Air (TO-220)		65	°C/W
T_{OPR}	Operating Temperature Range	LM78xx	-40 to +125	°C
		LM78xxA	0 to +125	
T_{STG}	Storage Temperature Range		-65 to +150	°C

Fonte: Farchild Semiconductor, 2012.

Figura 49 - Aplicação do Regulador de Tensão LD33V.



Fonte: Datasheet, ST.

Figura 48 – Características Elétricas LD33V.

Symbol	Parameter		Value	Unit
V_{IN} (1)	DC input voltage		15	V
P_{TOT}	Power dissipation		12	W
T_{STG}	Storage temperature range		-40 to +150	°C
T_{OP}	Operating junction temperature range	for C version	-40 to +125	°C
		for standard version	0 to +125	°C

Fonte: Datasheet, ST.

4.2.10. PROTEUS

De acordo com Kleiton (2014), o simulador Proteus é uma ferramenta útil para desenvolvimento de placas eletrônicas. É um software que permite o desenho de circuitos no qual é possível colocar os símbolos representativos de componentes e realizar simulações evitando risco de ocasionar danos aos circuitos. É um software que disponibiliza vários componentes, incluindo microcontroladores, com a possibilidade de observar o funcionamento de forma virtual (SANTOS, 2009).

Dentro do software Proteus, encontra-se uma ferramenta que permite o desenvolvimento de esquema elétrico chamado Schematic Capture e o PCB Layout - ferramenta para o desenvolvimento de layouts de placas de circuito eletrônico (PCB), na qual importa os componentes e a características dos circuitos produzidos no Schematic Capture e permite o roteamento de trilhas, tamanhos dos pads, etc.

Este software foi essencial para o desenvolvimento e confecção das placas eletrônicas do projeto. A versão utilizada para o projeto foi o Proteus 8 Professional.

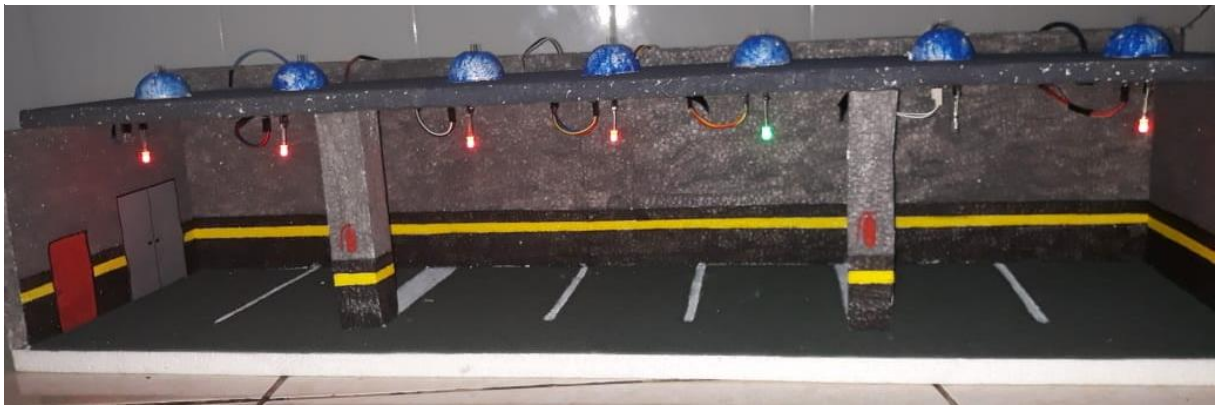
4.2.11. AMBIENTE DE SIMULAÇÃO

Para testes e validação do sistema, foi realizado a montagem de uma estrutura em maquete com características de um estacionamento com uma escala menor. A estrutura possui lugar para 7 carros em miniatura e é coberta (para auxiliar no posicionamento dos sensores). A estrutura montada foi baseada na Figura 51 e o resultado da maquete montada na Figura 50.

Figura 51 - Maquete de Estacionamento.

Fonte: https://www.egobox.com.br/produto/estacionamento-escala-1-43-modulos-diorama-maquete/687648

Figura 50 - Maquete Pronta.



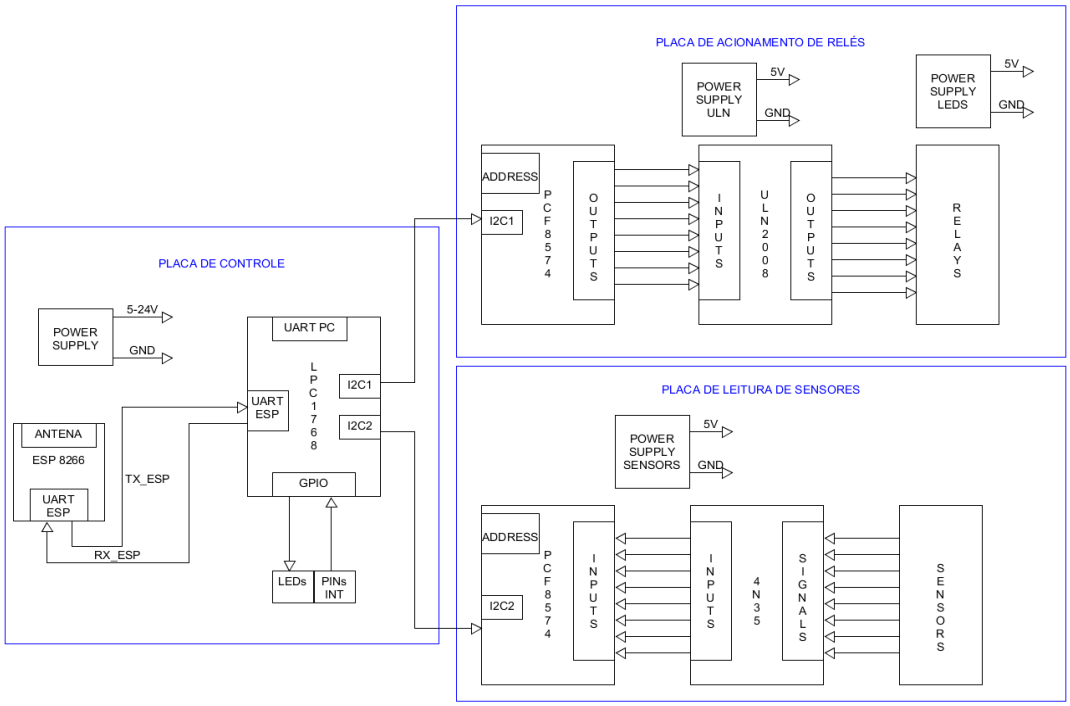
Fonte: Autor.

4.3. ARQUITETURA DE HARDWARE

Após a definição dos componentes a serem utilizados no projeto, deve-se integrá-los de forma que seja alcançado os resultados esperados que é realização da leitura do estado das vagas por meio de sensores de distância (Sensor de Obstáculo Reflexivo Infravermelho) posicionados no teto das vagas. Serão utilizados expansores de portas (PCF8574) para que sejam lidos vários sensores simultaneamente utilizando apenas o barramento de comunicação I^2C. Os sinais captados serão enviados para o microcontrolador principal que é responsável pela tomada de decisões, o LPC1768. Após o processamento das informações, este microcontrolador se comunicará com o módulo wifi através da interface de comunicação serial. Além disso o sistema oferecerá sinalização por leds para indicar visualmente o estado das vagas. Para acionar os leds e mudar a sua cor de acordo a situação apresentada pelas vagas, serão utilizados relés que é acionado pelo ULN2803 que, por sua vez, será controlado pelo PCF8574. Para atender a todas essas condições, propôs-se a arquitetura de hardware apresentada na Figura 52.

A arquitetura de hardware proposta está dividida em três módulos: placa de controle, placa de acionamento de relés e placa de leitura de sensores.

Figura 52 - Arquitetura de Hardware Proposto.



Fonte: Autor.

4.3.1. Placa de Controle

Com base na arquitetura de hardware proposta e pesquisas realizadas em datasheets a respeito dos componentes eletrônicos definidos para o hardware da placa de controle do sistema, utilizou-se o software Proteus.

No Proteus, a etapa de construção da placa de circuito impresso acontece em duas etapas: a primeira é definir o esquema elétrico no Schematic Capture e em seguida a montagem do layout da placa no Layout PCB.

4.3.1.1. ESQUEMA ELÉTRICO

Para montar o esquema elétrico, deve-se selecionar os componentes que serão utilizados no Schematic Capture. A placa de controle será composta por:

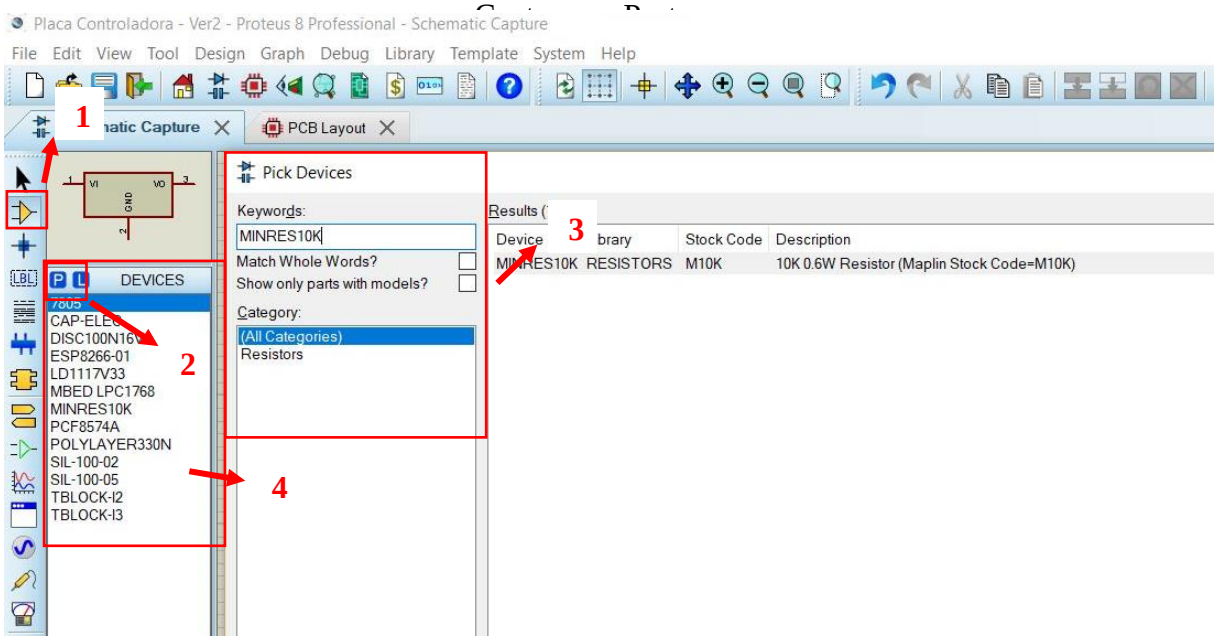
- 6 resistores de 10KΩ, serão os resistores pullup do barramento I²C (3 para cada barramento);
- 1 regulador de tensão do tipo 7805, será usado para regular a tensão de entrada para 5 volts para energizar o kit de desenvolvimento LPC1768;

- 1 regulador de tensão do tipo LD33V que será utilizado para regular a tensão de entrada para 3,3 volts para o ESP8266;
- Capacitores conforme solicitado pelo datasheet dos reguladores de tensão;
- Pinos de entrada da alimentação por fonte externa e pinos para a saída dos barramentos I²C;
- LPC1768, microcontrolador principal;
- ESP8266, módulo wifi.

Seguindo a arquitetura de hardware definida para a placa de controle, deve-se importar os componentes para a área de trabalho onde será desenvolvido o esquema elétrico.

Para a inserção de componentes no projeto, deve-se usar o Schematic Capture. A Figura 53 apresenta a interface gráfica deste software.

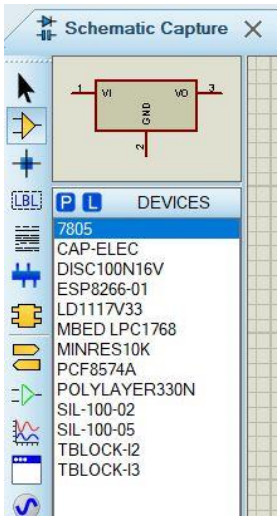
Figura 53 - Interface Gráfica do Schematic



Fonte: Autor.

O item destacado como número 1 deve ser clicado e a ferramenta numerada com o número 4 aparece. Esta ferramenta apresenta a letra P e L. A letra P deve ser clicada para que a janela Pick Devic e seja apresentada disponibilizando todos os componentes do software. E na barra Keywords deve ser digitado os componentes que deveram ser adicionados. Para o projeto da placa de controle, os componentes do projeto já foram adicionados, como mostra a Figura 55.

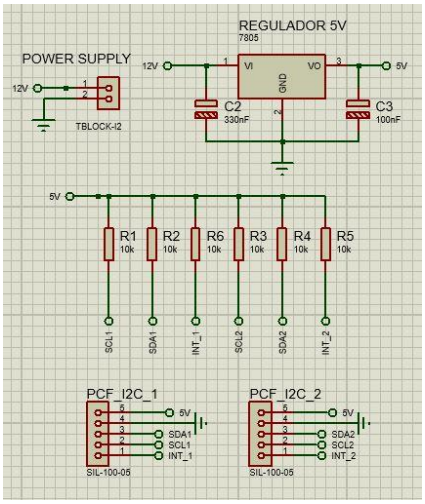
Figura 55 – Componentes para a Placa de Controle.



Fonte: Autor.

Após a inserção dos componentes, deve-se conectá-los conforme especificado na Figura 52 que apresenta a arquitetura de hardware. como observado, serão utilizados os pinos dos dois barramentos I²C disponibilizados pelo kit de desenvolvimento que são os pinos: P9 e P10 para o I²C_1 e os pinos P27 e P28 para o I²C_2. O pino de saída do regulador de 5 volts, 7805CV, deve fornecer tensão de alimentação para os dispositivos conectados aos barramentos. Os resistores de 10 KΩ devem ser conectados entre o VCC (5 volts) e os pinos dos barramentos I²C, que são: o Interrupt, o SDA e o SCL.

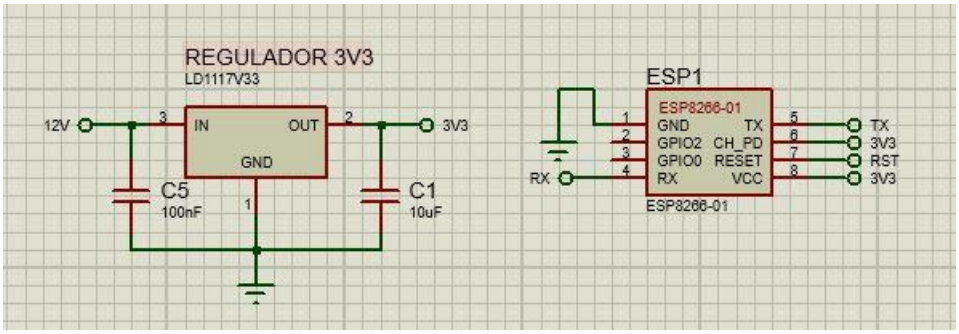
Figura 54 - Conexões do barramento I²C.



Fonte: Autor.

Deve-se também, conectar o ESP8266-01. Primeiramente tem que disponibilizar tensão de alimentação de 3,3 V para o módulo e para isso, foi disponibilizado o regulador de tensão LD117V33. Além da alimentação, o microcontrolador deve disponibilizar três pinos para comunicar via UART com o ESP8266-01, que são: o P12 - para conectar no pino de RESET do módulo, e os pinos P13 (conectar no pino Rx do módulo) e P14 (conectar no Tx do módulo) para comunicação serial entre eles.

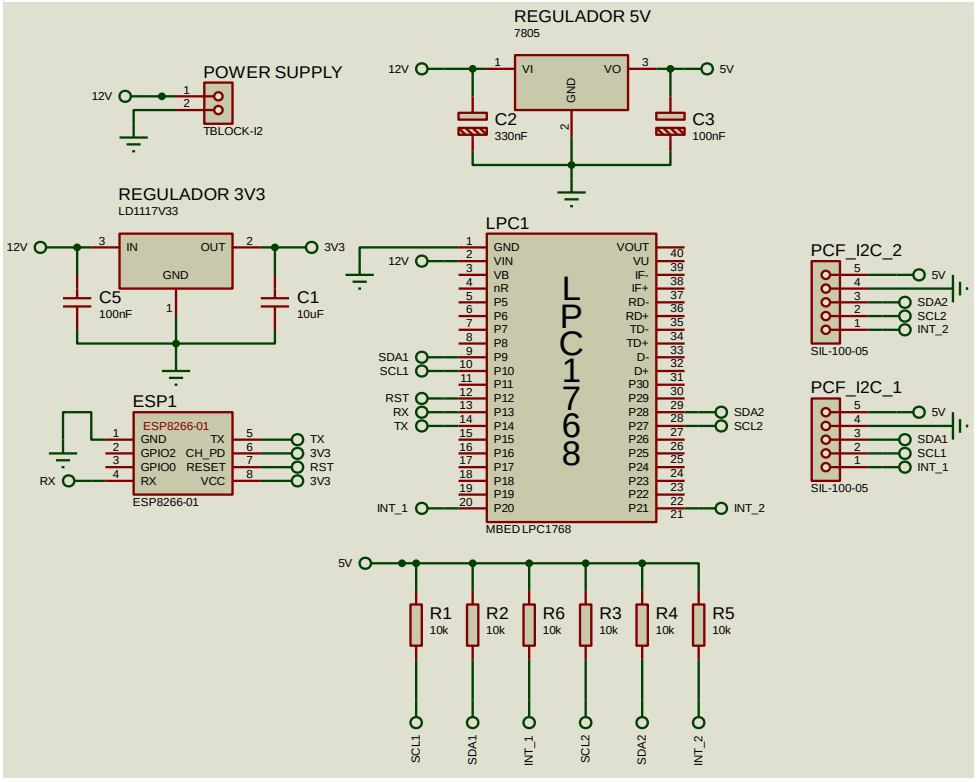
Figura 56 - Conexões do ESP8266-01.



Fonte: Autor.

O LPC1768 é alimentado diretamente pela fonte externa, sendo que o valor da tensão deve estar entre 5 a 12 volts. Nessas condições, o resultado do esquema elétrico ficou conforme a Figura 57.

Figura 57 - Esquema Elétrico da Placa de Controle.



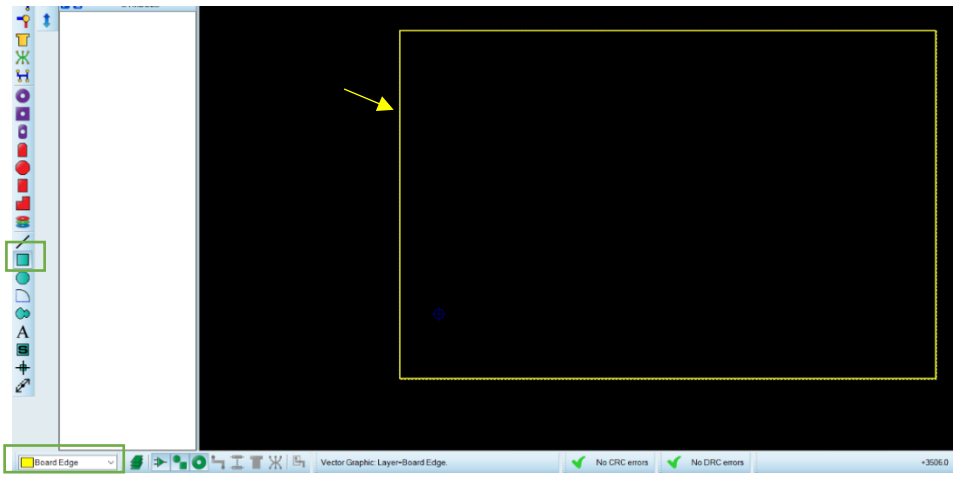
Fonte: Autor.

4.3.1.2. LAYOUT PCB

Após a conclusão do esquema elétrico, é necessário montar o layout da placa. Para isso, é usado a ferramenta contida no Proteus que é o PCB Layout. Esse programa realiza a organização dos componentes, configura e realiza roteamento das trilhas.

Antes de posicionar os elementos, deve-se definir o tamanho da placa. Este procedimento é realizado selecionando o objeto retangular chamado 2D Graphics Box Mode na aba a esquerda, em seguida é selecionado o modo Board Edge e depois é feito a delimitação da área, conforme a Figura 58.

Figura 58 - Definindo Limites

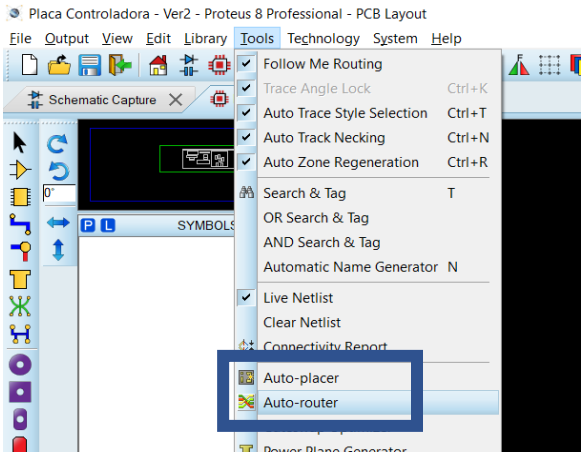


Fonte: Autor.

No Layout PCB, existe a ferramenta *Auto-placer* – que é responsável por fazer o posicionamento de todos os componentes automaticamente. Porém esta função não foi utilizada neste projeto e os posicionamentos foram realizados manualmente. E há, também, a ferramenta Auto-router que realiza o roteamento de trilhas automaticamente, o que é uma grande vantagem. Pois se fosse realizado manualmente, demandaria bastante tempo. Esta ferramenta é facilmente configurada e pode ser encontrada clicando em Tools, conforme mostrada na Figura 59.

Porém, antes de rotear trilhas, deve-se definir as características e as regras das trilhas, informar a quantidade de Layers que a placa deve possuir. Estes parâmetros são configurados através da função Design Rule Manager encontrada na aba Technology.

Figura 59 - Configurações básicas.

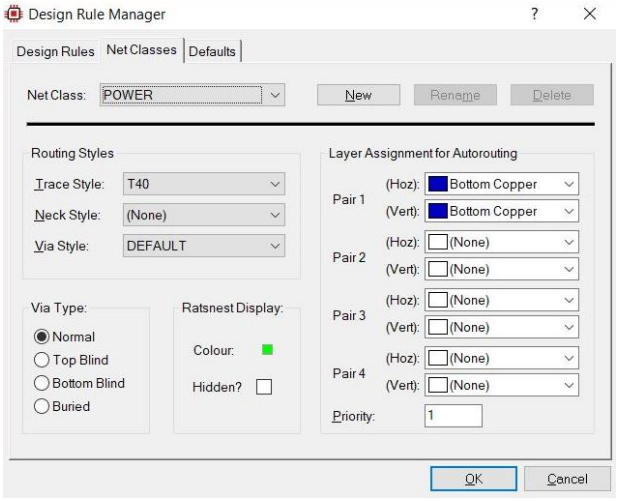


Fonte: Autor.

É necessário definir a trilha para sinais e para as alimentações. Os parâmetros de configuração das trilhas utilizadas são: Trace Style e Layer Assignment for Autorouting. O trace style define a largura da trilha enquanto o Layer Assignment for Autorouting define onde deverão ser posicionados as trilhas.

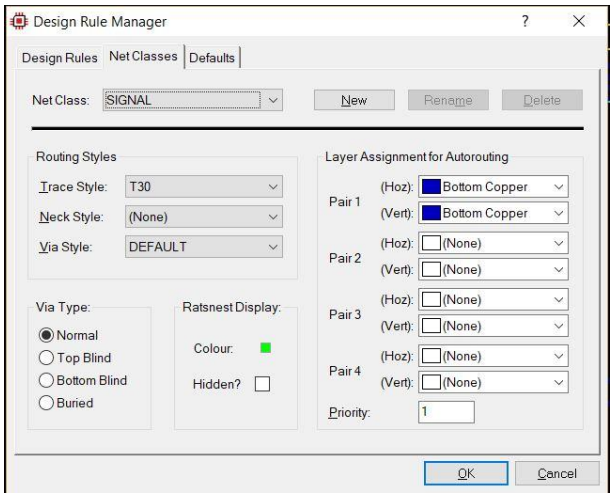
Para todas as placas desenvolvidas neste projeto foi definido que para as trilhas de alimentação o Trace Style será T40 e o Layer Assignment for Autorouting serão definidos como Bottom Copper, ou seja, as trilhas serão posicionadas na parte inferior da placa, conforme a Figura 60. As placas de fenolite, utilizadas para confecção de PCB, encontradas em Manaus, só possuíam placa de cobre em uma face. E as configurações das trilhas dos sinais foi definido conforme a Figura 61.

Figura 60 - Regra das trilhas de alimentação.



Fonte: Autor.

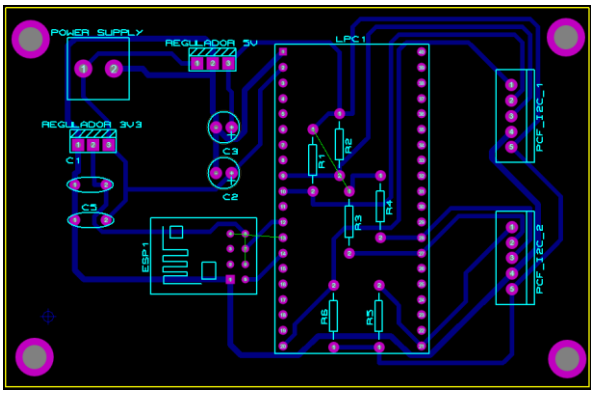
Figura 61 - Regra das trilhas de sinais.



Fonte: Autor.

Após posicionar todos os componentes e configurar as trilhas, deve-se realizar o roteamento automático com o Auto-router. Essa ferramenta utiliza métodos matemáticos para rotear as trilhas da melhor maneira possível. Caso não seja possível realizar o roteamento de algumas trilhas, será mostrado linhas verdes sinalizando que a conexão especificada não foi realizada. O resultado do roteamento foi:

Figura 62 – Layout da Placa de Controle.



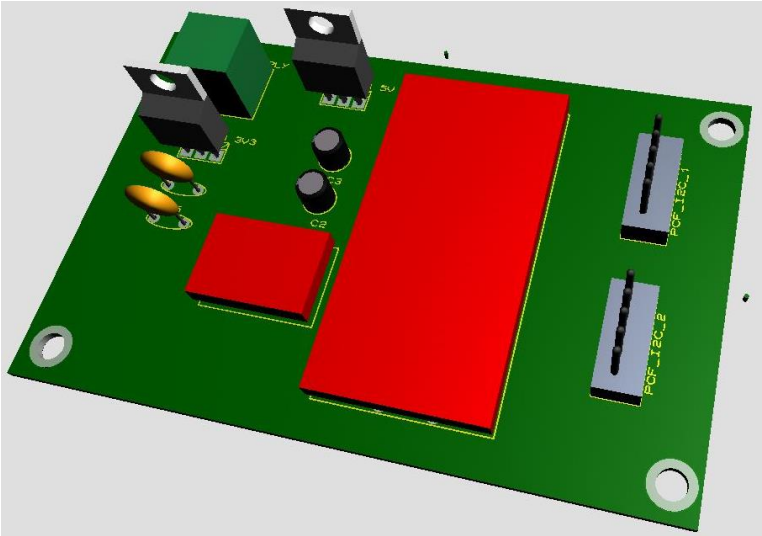
Fonte: Autor.

Observa-se que o roteamento automático não conseguiu realizar três conexões. Faltou conectar: a trilha dos 5V na trilha que conecta os resistores R3 e R4, o pino RX do módulo wifi com o P13 do microcontrolador e o pino CH_PD do ESP8266 com a trilha de 3V3.

4.3.1.3. VISUALIZAÇÃO EM 3D DA PLACA

Outra ferramenta disponibilizada pelo Proteus é a visualização em 3D da placa. Usando esta ferramenta para visualizar a placa de controle, teremos:

Figura 63 - 3D da placa de controle.



Fonte: Autor.

A Figura 63 apresenta dois retângulos vermelhos no qual o componente vermelho maior é o kit de desenvolvimento LPC1768 e o menor é o ESP8266-01.

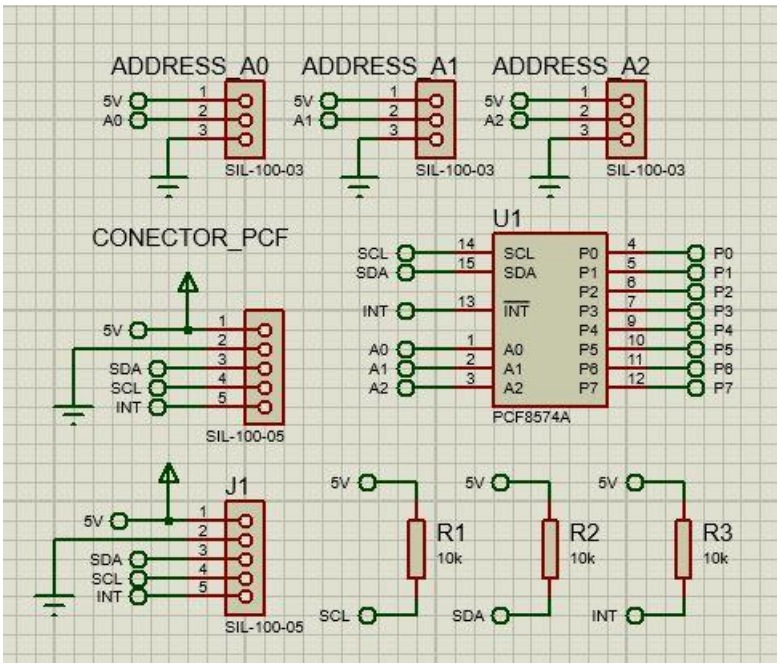
4.3.2. Placa de Leitura

Para o desenvolvimento da placa de leitura, criou-se um novo projeto no Proteus. E antes de começar o desenvolvimento da placa deve-se levantar os requisitos da placa e eles são citados a seguir:

- Disponibilizar duas conexões de entrada para o barramento I²C, um para conectar a placa de controle e outro para uma possível conexão com outra placa de leitura;
- Conexão para alimentação externa;
- Conectores para definir o endereçamento do PCF;
- Conectores para conexão dos sensores que devem possuir três pinos, o VCC, GND e Ground;
- Três resistores pull-up para os pinos do barramento I²C da placa de leitura;
- Oito optoacopladores 4N35;
- Oito transistores.

A placa é composta por um PCF, logo deve-se preparar o barramento I²C da placa. A placa deve disponibilizar dois terminais de cinco pinos, uma para conectar com a placa controladora e outra para conectar com outra placa de leitura. O barramento possui os pinos de Interrupt, SDA e SCL, e cada um deve ser conectado a um resistor pull-up de 10Kohm. Além disso, deve-se disponibilizar terminais para que seja definido o endereço do PCF no barramento. A Figura 64 apresenta o circuito do barramento I²C juntamente com o PCF8574.

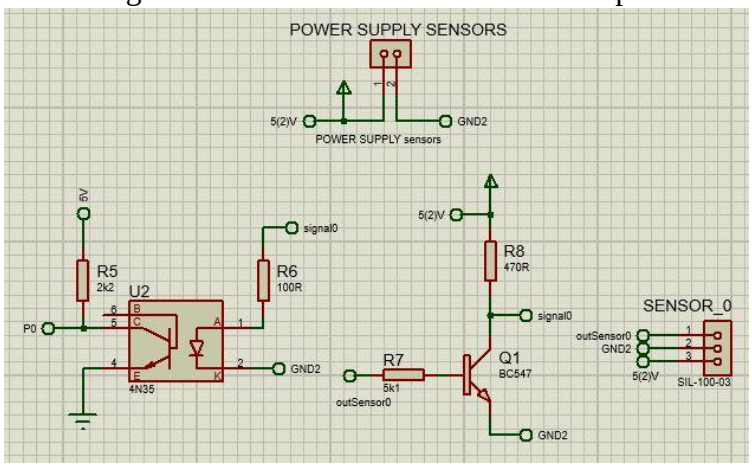
Figura 64 - Esquema elétrico do Barramento I²C.



Fonte: Autor.

A placa de leitura possui terminais de alimentação externa para alimentar os sensores. E para proteger o PCF8574 de possíveis sobrecargas causadas por esta alimentação externa, coloca-se o optoacoplador entre o PCF8574 e os sensores. Os sinais dos sensores devem ser colocados em série com o fototransistor, pois quando energizado emitirá luz que fecha o circuito secundário, então os sensores conectam-se nos pinos 1 e 2 do CI 4N35. E os pinos 5 de cada optoacoplador, devem ser conectados aos pinos P0 ao P7 do CI PCF8574. Com essas informações, o circuito apresentará as seguintes características:

Figura 65 - Conexões com os sensores ópticos.

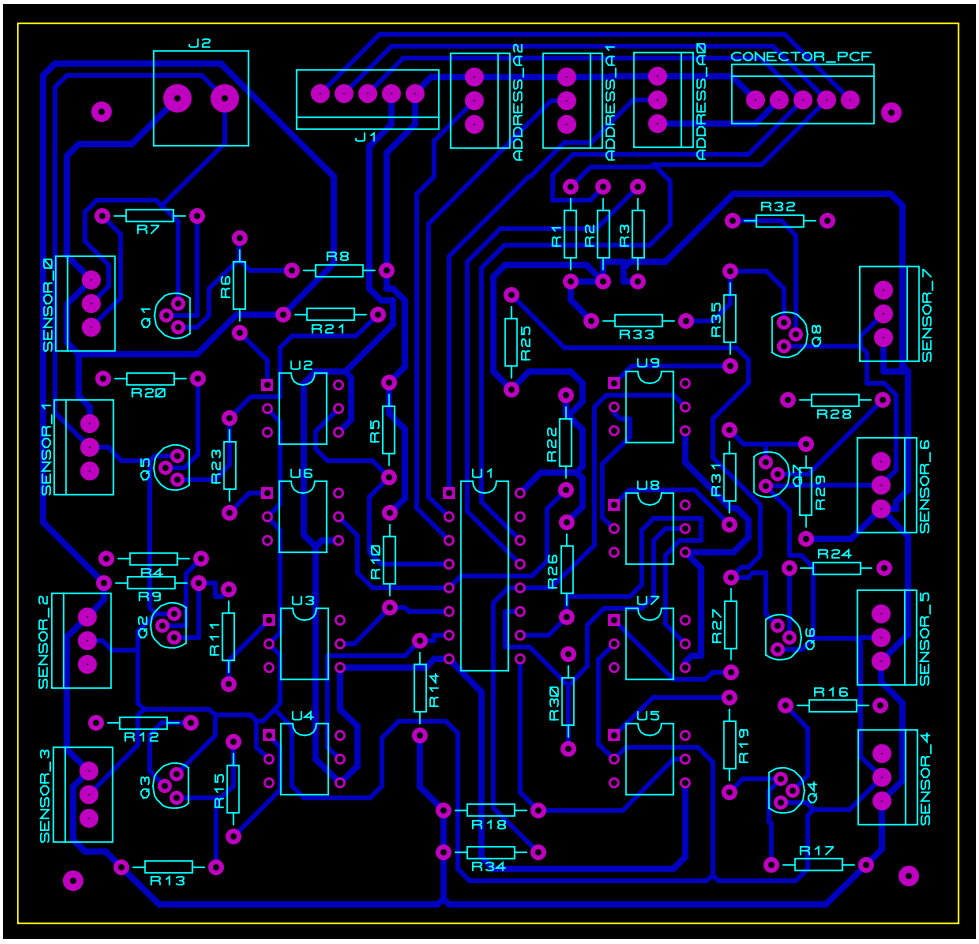


Fonte: Autor.

4.3.2.1. LAYOUT PCB

A pós a finalização do esquema elétrico, o layout da PCI pode ser desenvolvido. E o resultado do layout para esta placa foi o seguinte:

Figura 66 - Layout PCI da Placa de Leitura.



Fonte: Autor.

4.3.3. Placa de Escrita

Cria-se um novo projeto no *Proteus* e levanta-se os requisitos que devem ser atendidos para o desenvolvimento da placa de escrita:

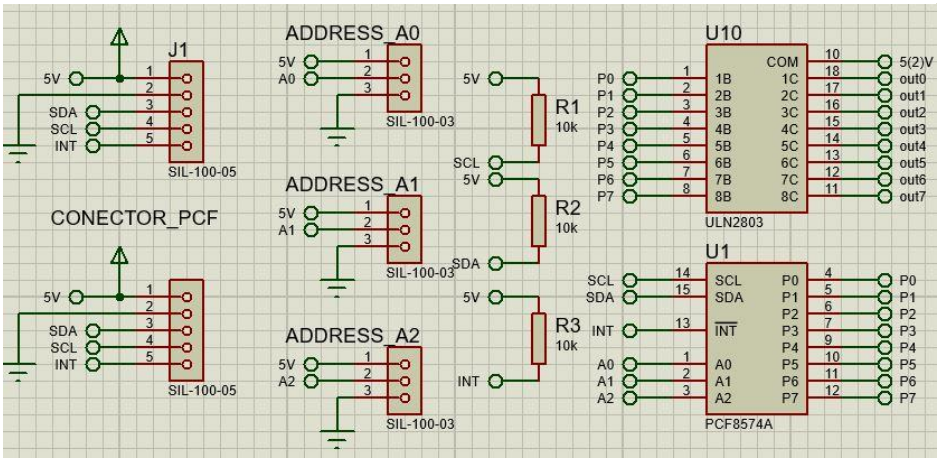
Duas conexões para fonte externa, uma para os sinais do relé e um para alimentar a placa dos leds;

- Acionamento de até oito relés;
- Utilizar o PCF8574 para acionamento dos oito pinos do ULN2803A;
- Disponibilizar três terminais de três pinos para definir o endereço do PCF8574 da placa;
- Três resistores *pull-up* de 10KΩ para conectar entre o Vcc e os sinais *Interrupt*, SDA e SCL do barramento I²C;
- Utilizar um ULN2803A para acionamento dos relés;
- Disponibilizar oito terminais de três pinos para conectar na placa do Led;
- Disponibilizar dois terminais de cinco pinos para o barramento I²C, um para a placa de controle e outra para conexão com outra placa de escrita.

4.3.3.1. ESQUEMA ELÉTRICO

Conforme definido pela arquitetura de hardware, uma das conexões para a placa de escrita é a conexão dos pinos P0 ao P7 do PCF8574 aos pinos de entrada do ULN2803A que são o 1B até o 8B. E por se tratar de um PCF, deve ser fornecidos terminais que permitam configurar o endereço e utilizar em seu barramento I²C resistores pull-ups nos sinais SDA, SCL e Interrupt. Então define-se as conexões conforme a Figura 67:

Figura 67 - Conexões do PCF8574 com ULN2803A.



Fonte: Autor.

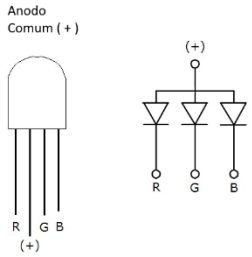
Conforme citado no capítulo 2, o acionamento dos relés é realizado energizando-se seus pinos de controle. O relé utilizado neste projeto é mostrado na Figura 68.

Figura 68 - Relé Utilizado no Projeto.

Fonte: Datasheet do Fabricante.

Um dos pinos de controle do relé deve ser conectado a trilha de alimentação de 5 volts, enquanto o outro deve ser conectado a uma das saídas do ULN2803A que será o responsável por enviar sinal para o relé comutar. As conexões dos outros três pinos de potência do relé que são o NO (normalmente aberto), NC (normalmente fechado) e COM (comum) dependem do dispositivo que será acionado. O dispositivo a ser conectado aos relés serão os leds RGB, os quais possuem as características mostrada na Figura 69.

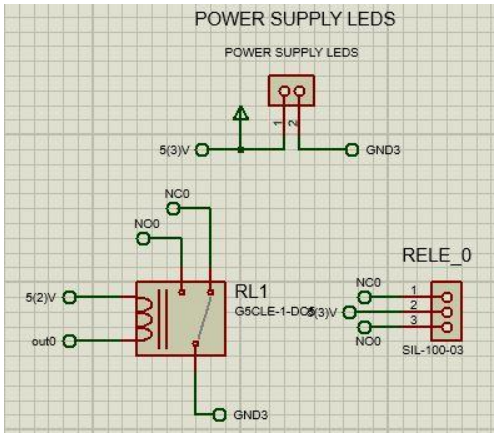
Figura 69 - LED RGB.



Fonte: Filipeflop.

Analisando a Figura 69, conclui-se que o relé deve chavear o ground para os pinos R (red) e G (green) do led para acender um ou outro. Portanto, os terminais de saída da placa da escrita para a placa de led deve conter um pino fornecendo 5volts - que será fornecida por um terminal de dois pinos que deve-se conectar uma fonte externa, e os outros dois terminais vão chavear o ground da fonte externa para o R (red) ou G (green) do Led. E com isso, as conexões realizadas estão representadas conforme a Figura 70.

Figura 70 - Conexões do Relé.

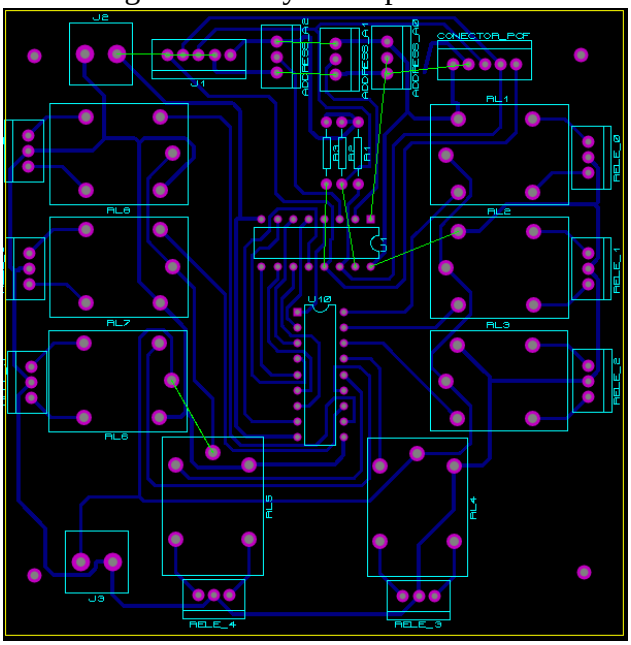


Fonte: Autor.

4.3.3.2. LAYOUT PCB

Conforme realizado em outros projetos de placas anteriores, deve-se desenvolver o layout da placa. O resultado é mostrado na Figura 71:

Figura 71 - Layout da placa de escrita.

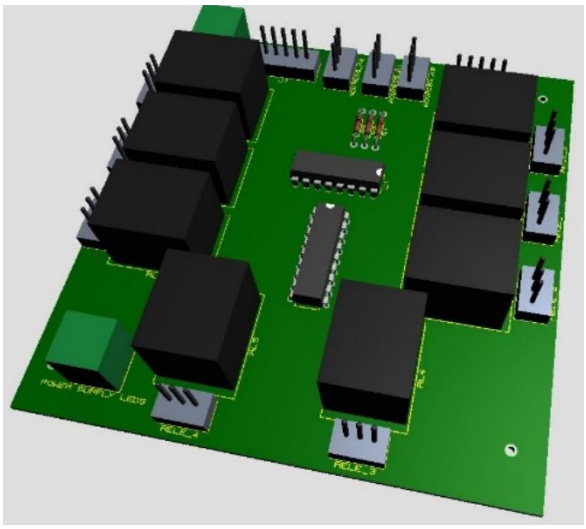


Fonte: Autor.

Observa-se que houve nove trilhas que não foram conectadas. A conexão dessas trilhas será corrigida após a confecção das placas, utilizando jumpers para conectá-los.

4.3.3.3. VISUALIZAÇÃO EM 3D DA PLACA

Figura 72 - Visualização da Placa de Escrita em 3D.



Fonte: Autor.

4.3.4. Placa do LED

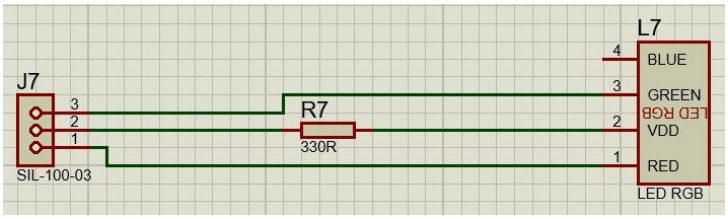
Esta placa tem apenas um objetivo, acender o Led RGB para sinalizar visualmente se a vaga está livre ou ocupada e, portanto, torna-se o projeto de placa mais simples do sistema. Os requisitos para esta placa são:

- Fornecer terminais para energização do led;
- Um resistor de 330 ohms para limitar a corrente do led.

4.3.4.1. ESQUEMA ELÉTRICO

A única conexão do circuito é a do resistor entre o Vcc da placa e o anodo do led RGB. O resultado da conexão é:

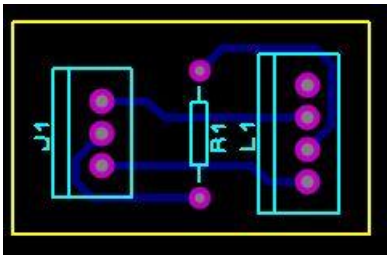
Figura 73 - Esquema Elétrico da Placa do Led.



Fonte: Autor.

4.3.4.2. LAYOUT PCB

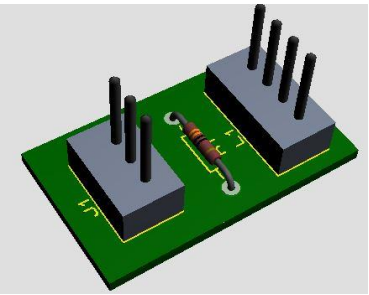
Figura 74 - Layout da Placa do Led.



Fonte: Autor.

4.3.4.3. VISUALIZAÇÃO EM 3D DA PLACA

Figura 75 - Visualização em 3D

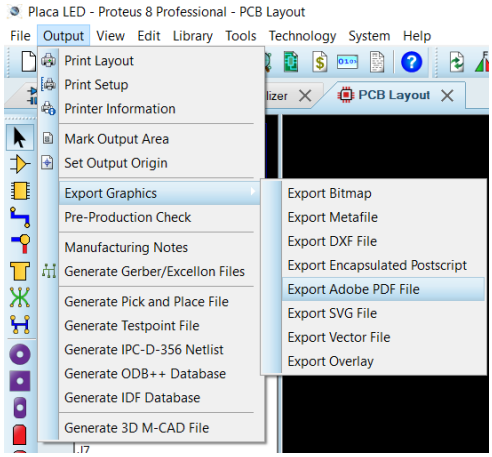


Fonte: Autor.

4.4. CONFECÇÃO DAS PLACAS

Primeiramente deve ser gerado documentos no formato pdf das trilhas das placas para que elas sejam impressas em papel de fotografia. Podem-se gerar o arquivo no próprio software Proteus na ferramenta Layout PCB, como é mostrado na Figura 76.

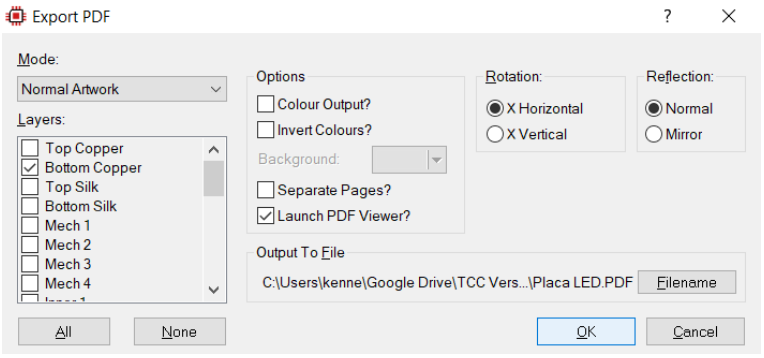
Figura 76 - Gerando Layout das Placas em Arquivo PDF.



Fonte: Autor.

Após selecionar a opção de exporta como PDF, a janela a seguir vai aparecer na tela. E deve-se marcar somente as opções de Layers como Bottom Copper e Board Edge e clique em OK.

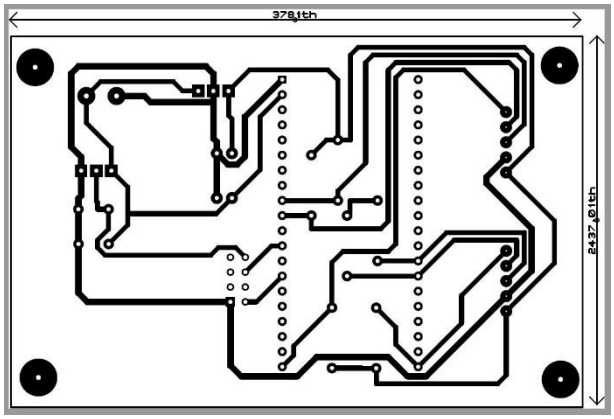
Figura 77 - Tela de configuração do documento.



Fonte: Autor.

Ao abrir o arquivo pdf gerado, o conteúdo armazenado é mostrado na Figura 78.

Figura 78 - Layout Impresso.



Fonte: Autor.

As placas de circuito impresso são feitas a partir de uma base não condutiva, como fenolite, cobertas por uma camada de cobre, que é um bom condutor de corrente elétrica. O fenolite utilizado no processo é mostrado na Figura 79.

Figura 79 - Placa de Fenolite Cobreada de Face Simples.

Fonte: Autor.

Durante a confecção da placa, todo o cobre deve ser subtraído, excetuando as trilhas impressas que farão a conexão dos componentes. Para que as trilhas permaneçam na placa de fenolite após o processamento químico de corrosão, as trilhas geradas no documento em pdf deverão estar “impressas” na camada cobreada da placa.

A trilha gerada no arquivo pdf deve ser impressa, de preferência, em papel de fotografia, a qual será usada para transferir a tinta impressa no papel para a camada cobreada. Antes de transferir a tinta do papel para a placa de fenolite, a placa deve estar limpa. A limpeza é feita esfregando esponja de lã de aço na camada cobreada, até que o cobre esteja brilhando. O próximo passo é limpar a camada de cobre com álcool isopropílico. Durante o processo deve-se evitar tocar na placa para não adicionar sujeiras.

Figura 80 - Produtos

Fonte: Autor.

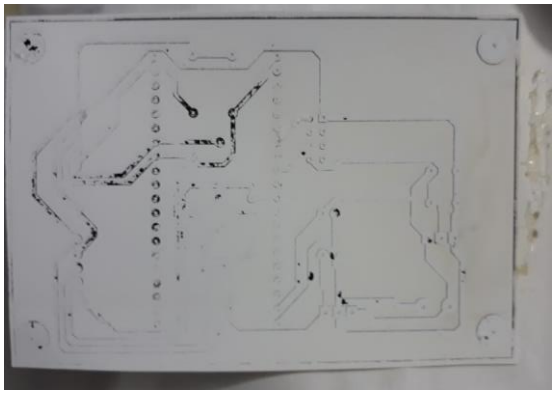
Com uma fita adesiva, deve-se fixar a trilha impressa no papel na parte cobreada da placa de fenolite.

Figura 81 - Fixação da Trilha impressa na parte cobreada.

Fonte: Autor.

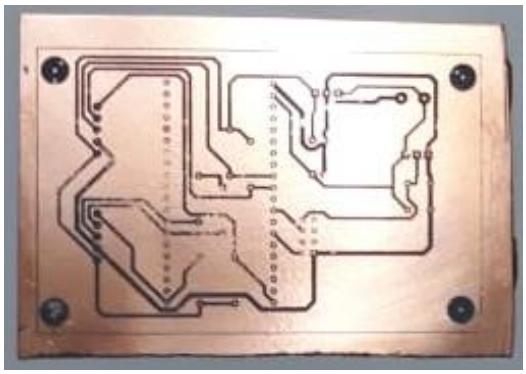
Após a fixação da placa e do papel, com um ferro de passar roupa, esfregou-se o conjunto da Figura 81 por 7 minutos. O resultado foi a transferência da tinta contida no papel para a placa de cobre, conforme mostrado a seguir.

Figura 83 – Resultado no Papel.



Fonte: Autor.

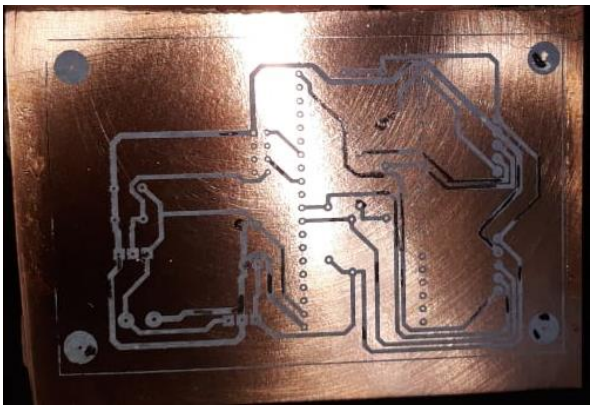
Figura 82 – Resultado no Fenolite.



Fonte: Autor.

Conforme observado, nas figuras após o aquecimento do conjunto, nem toda a tinta foi transferida para a face cobreado e por isso algumas trilhas apresentaram falhas. Para contornar esse problema, utilizou-se caneta de tinta permanente para completar ou consertar as trilhas na placa.

Figura 84 - Trilhas falhadas corrigidas.



Fonte: Autor.

Figura 85 - Caneta Permanente.

Fonte: Autor.

Com as trilhas transferidas para a placa, agora deve-se subtrair o cobre da placa. Isso é feito utilizando solução de percloreto de ferro. Foi adquirido este elemento em pó, conforme a Figura 86, e o diluiu em 0,5 litros de água em um vasilhame de vidro. Após misturar a solução, mergulhou-se a placa até que restasse somente as trilhas na placa.

Figura 86 - Percloreto de Ferro Usado.

Fonte: Mercado Livre.

Figura 87 - Processo de Corrosão da Placa.

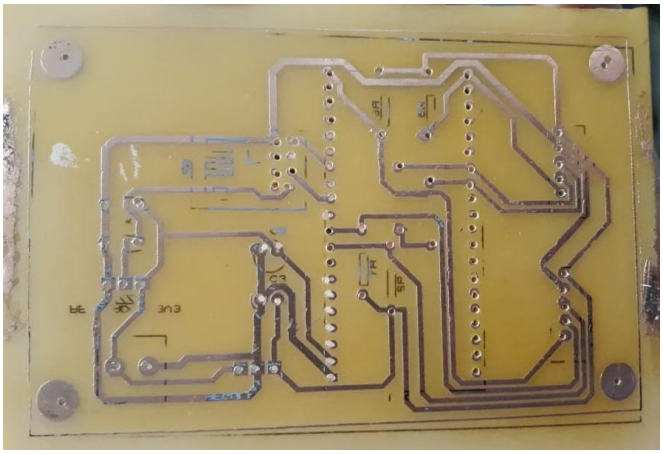
Fonte: Autor.

Quando restar somente as trilhas, deve-se lavar a placa com água e retirar a tinta de impressão, para isso, usa-se a esponja de lã de aço novamente. Após a lavagem, a placa deve ser furada para que os componentes sejam posicionados e soldados. O furador utilizado é apresentado na Figura 88 e a placa após as furações é mostrada posteriormente na Figura 89.

Figura 88 - Furador de Placas.

Fonte: Autor.

Figura 89 - Placa Furada.



Fonte: Autor.

Após as furações, as placas estão preparadas para receber os componentes. Para a soldagem utilizou-se ferro de solda, solda, malha dessoldadora, fluxo de solda e os componentes eletrônico das placas. Os materiais são apresentados a seguir.

Figura 90 - Materiais usados para a soldagem.

Fonte: Autor.

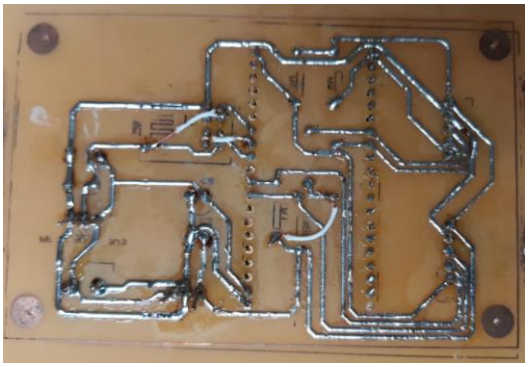
Com a soldagem dos componentes concluídas, deve-se ressaltar que durante o roteamento nem todas as trilhas foram conectadas. E para realizar essas conexões, utiliza-se jumpers, que são fios comuns, para consertar as conexões. Os procedimentos citados, foram repetidos para a confecção de todas as placas do projeto.

A seguir serão apresentadas as placas montadas, juntamente com as correções das conexões identificadas no desenvolvimento do layout.

Figura 93 - Placa de Controle Soldada.

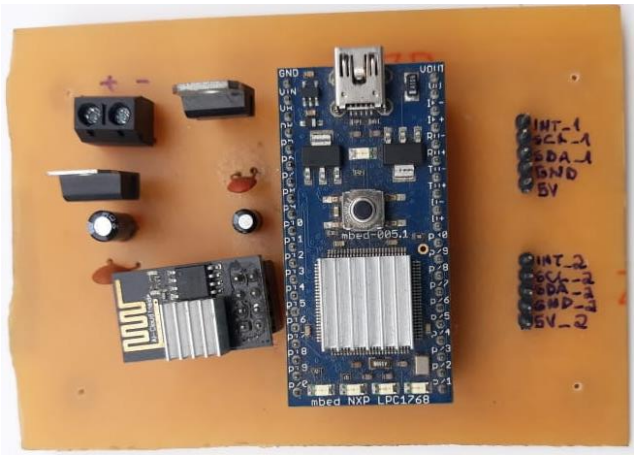
Fonte: Autor.

Figura 92 - Verso da Placa de Controle.



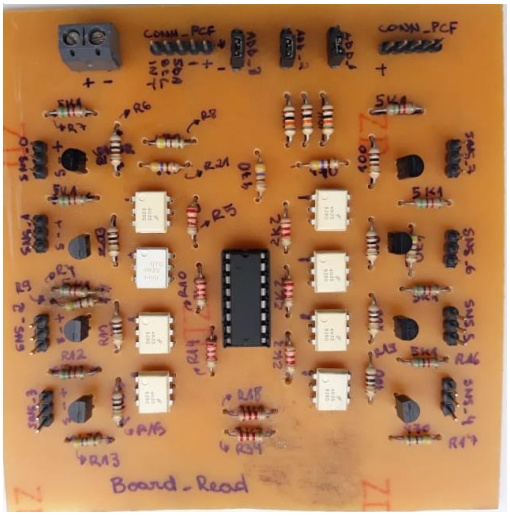
Fonte: Autor.

Figura 91 - Placa de Controle Completa.



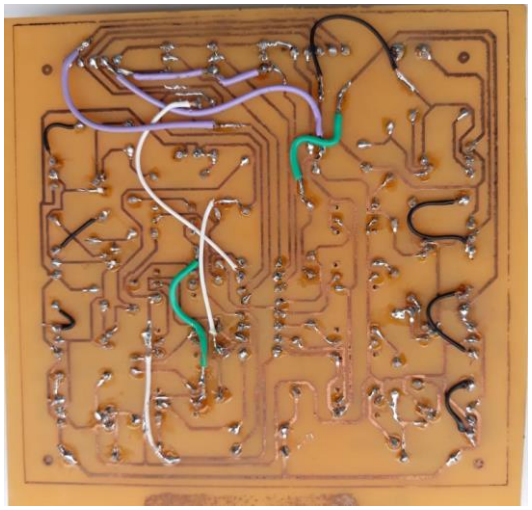
Fonte: Autor.

Figura 97 - Placa de Leitura Soldada.



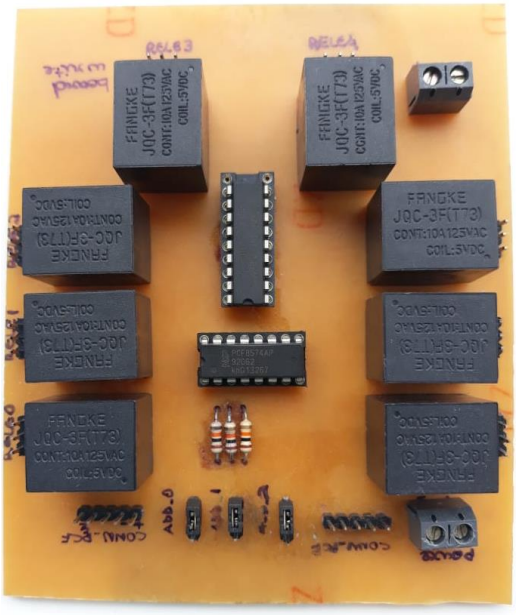
Fonte: Autor.

Figura 98 - Verso da Placa de Leitura.



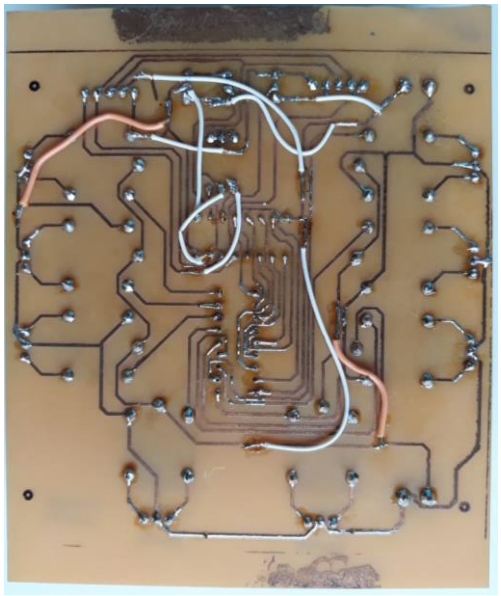
Fonte: Autor.

Figura 94 - Placa de Escrita Soldada.



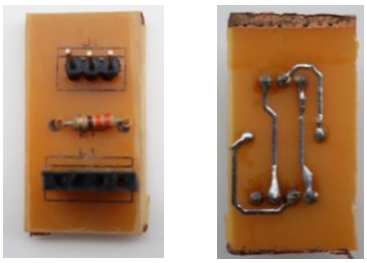
Fonte: Autor.

Figura 96 - Verso da Placa de Escrita.



Fonte: Autor.

Figura 95 - Placa do Led.



Fonte: Autor.

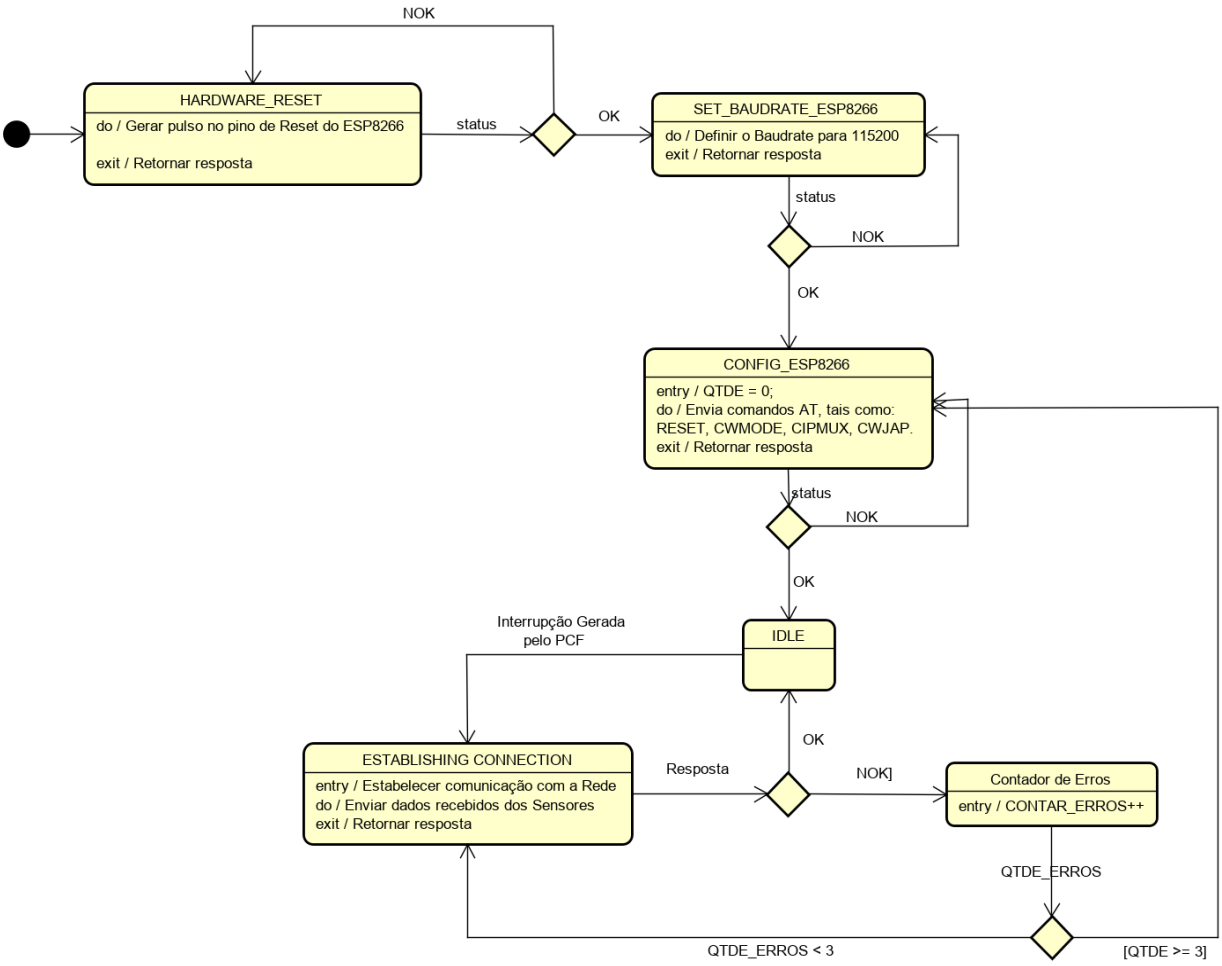
4.5. FIRMWARE

O firmware é um conjunto de instruções pré-programadas e compiladas pelo desenvolvedor, utilizando uma IDE, o qual é gerado um binário e este arquivo é embarcado no chip do microcontrolador. Este elemento é responsável por tomar as decisões e controlar as atividades do sistema. Antes que se desenvolva o firmware, utiliza-se a linguagem UML para desenvolver, visualizar e identificar os estados do sistema, e assim é gerado a máquina de estados do software.

4.5.1. Modelagem do Firmware em UML

O diagrama de máquina de estados da Figura 99 é a proposta para o desenvolvimento do firmware do projeto. O diagrama é composto por seis estados: o HARDWARE_RESET, SET_BAUDRATE_ESP8266, CONFIG_ESP8266, IDLE, ESTABLISHING_CONNECTION e CONTADOR DE ERROS.

Figura 99 - Modelagem em UML do firmware.



Fonte: Autor.

4.5.2. Descrição dos Estados

HADWARE_RESET: este estado é responsável por gerar um pulso no pino reset do ESP8266-01 que está conectado ao pino p12 do microcontrolador. Após gerado o pulso, o módulo wifi reiniciará suas configurações internas.

SET_BAUDRATE_ESP8266: neste estado é configurado a velocidade da comunicação serial entre o ESP8266-01 e o microcontrolador para 115200 bits por segundo.

CONFIG_ESP8266: nessa fase, o microcontrolador zera a variável que conta quantidade de erros e envia uma série de comandos para realizar a configuração do módulo wifi. Os comandos enviados são: o "AT", para verificar se o módulo wifi está pronto, o "AT+RST" para reiniciar o módulo, "AT+CWMODE" para definir o módulo como access point e station, "AT+CIPMUX" para definir o modo de comunicação e o comando "AT+CWJAP" que o conecta na rede wifi local.

IDLE: estado normal, o qual aguarda a interrupção gerada pelo PCF8574.

ESTABLISHING_CONNECTION: estado que conecta o ESP8266 ao servidor e envia e recebe dados do mesmo.

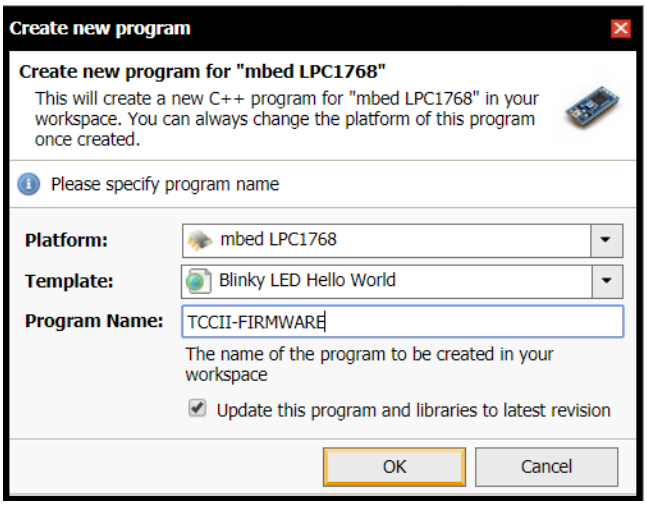
CONTADOR_DE_ERROS: adiciona 1 (um) a variável que conta a quantidade de erros e verifica se já contabilizou 3 (três) erros. Caso sim, o firmware retorna ao estado CONFIG_ESP8266 para que ele possa ser reiniciado por meio do comando AT, se não, ele tenta conectar o ESP8266 ao servidor mais uma vez.

4.5.3. Desenvolvendo o Firmware

Após a criação da máquina de estados, inicia-se o desenvolvimento do firmware utilizando a IDE online fornecida no site do fabricante do microcontrolador, chamada *Mbed Compiler*. Este ambiente de desenvolvimento utiliza a linguagem C/C++ para o desenvolvimento do código. Além de fornecer compilador para gerar binário para o microcontrolador, ela permite também a exportação de bibliotecas que foram fornecidas por outros desenvolvedores. A Figura 34 apresenta os principais componentes da interface gráfica do ambiente de desenvolvimento e a criação do projeto no IDE, que é o primeiro passo para o desenvolvimento do firmware, é apresentado na figura 99.

Para a criação de um projeto, deve-se clicar no item "New" que fica na parte superior direita e uma janela aparecerá no centro da tela, conforme a figura 100:

Figura 100 - Janela para Criação do Projeto.



Fonte: Autor.

Nomeie o projeto na barra ao lado de *Program Name* e clique em *OK* e o projeto será criado na seção *Progam Workspace*. Após criar o projeto, deve-se importar bibliotecas necessárias para o funcionamento do *firmware*. As bibliotecas são códigos que fornecem funções prontas a serem utilizadas a fim de cumprir seu propósito específico. Para o desenvolvimento do firmware utilizou-se a biblioteca desenvolvida e fornecida por Simon Ford para o gerenciamento do PCF8574, a página da web que fornece os códigos é mostrada na Figura 101.

Figura 101 - Biblioteca para PCF8574.

Fonte: Autor.

A biblioteca em questão está com o código comentado e fornece informações de como utilizar as funções. Na biblioteca tem função para inicializar, escrever e ler um byte do PCF8574 conectado ao barramento. As funções da biblioteca são apresentadas na Figura 102.

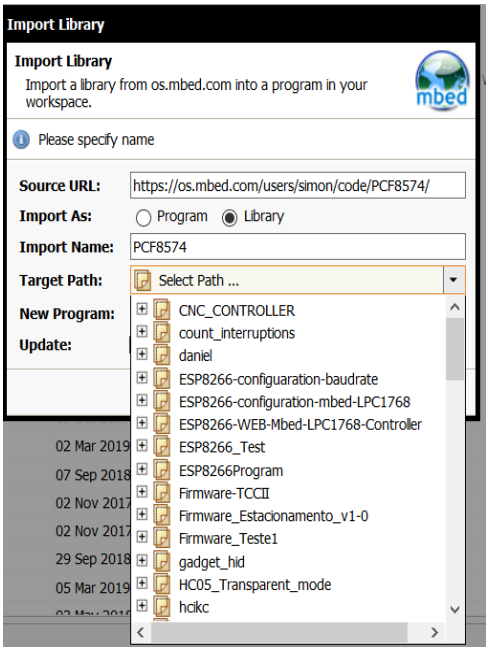
Figura 102 - Funções da biblioteca do PCF8574.

```
 */
PCF8574(PinName sda, PinName scl, int address);

/** Read the IO pin level
 *
 * @return The byte read
 */
int read();

/** Write to the IO pins
 *
 * @param data The 8 bits to write to the IO port
 */
void write(int data);
```

Fonte: Autor.

A importação da biblioteca é uma tarefa bastante simples neste ambiente de desenvolvimento. Ao visitar o site que fornece a biblioteca, deve-se clicar em *Import into Compiler* no canto esquerdo da página, como observado na Figura 101. Ao clicar no item de importação, a IDE será aberto e a janela mostrada na Figura 103 aparece. E deve-se escolher importar como biblioteca, em *Import As*, e em *Target Path* escolha para qual projeto deseja-se importar.

Figura 103 - Importando biblioteca.

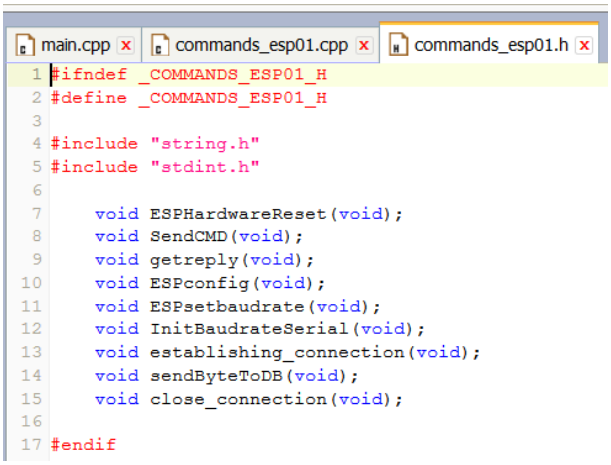


Fonte: Autor.

Além de gerenciar o controle do PCF8574, faz-se necessário o controle do ESP8266-01 por meio de comandos AT. E para isso, encontrou-se no site do fabricante aplicações de exemplos que auxiliam na comunicação do kit LPC1768 com o módulo wifi, figura 104.

E com base nessa documentação, criou-se uma biblioteca que fornece funções para a aplicação principal a fim de organizar e facilitar a manipulação dos códigos. As funções da biblioteca criada podem ser observadas na figura 105.

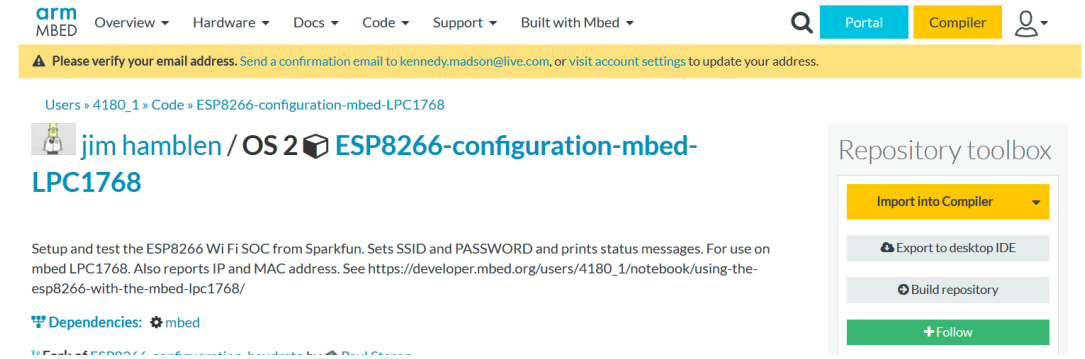
Figura 105 - Funções de Configuração do ESP8266.



Fonte: Autor.

Após a inclusão das bibliotecas no projeto, deve-se desenvolver a aplicação principal

Figura 104 - Aplicação de Configuração do ESP8266.



Fonte:https://os.mbed.com/users/4180_1/code/ESP8266-configuration-mbed-LPC1768/file/9f46b8cdd469/main.cpp/

no arquivo main.cpp do projeto. Com base nas informações geradas no diagrama de máquina de estados do firmware, foi desenvolvido o código principal. E o trecho mais importante do código, segue conforme é mostrado na Figura 107. Onde a função é composta por um switch case que contém os estados da máquina de estados definido no diagrama proposto na Figura 99 e foi declarado, antes da função *main*, uma variável que manipula o estado da máquina realizando, assim, o controle do sistema.

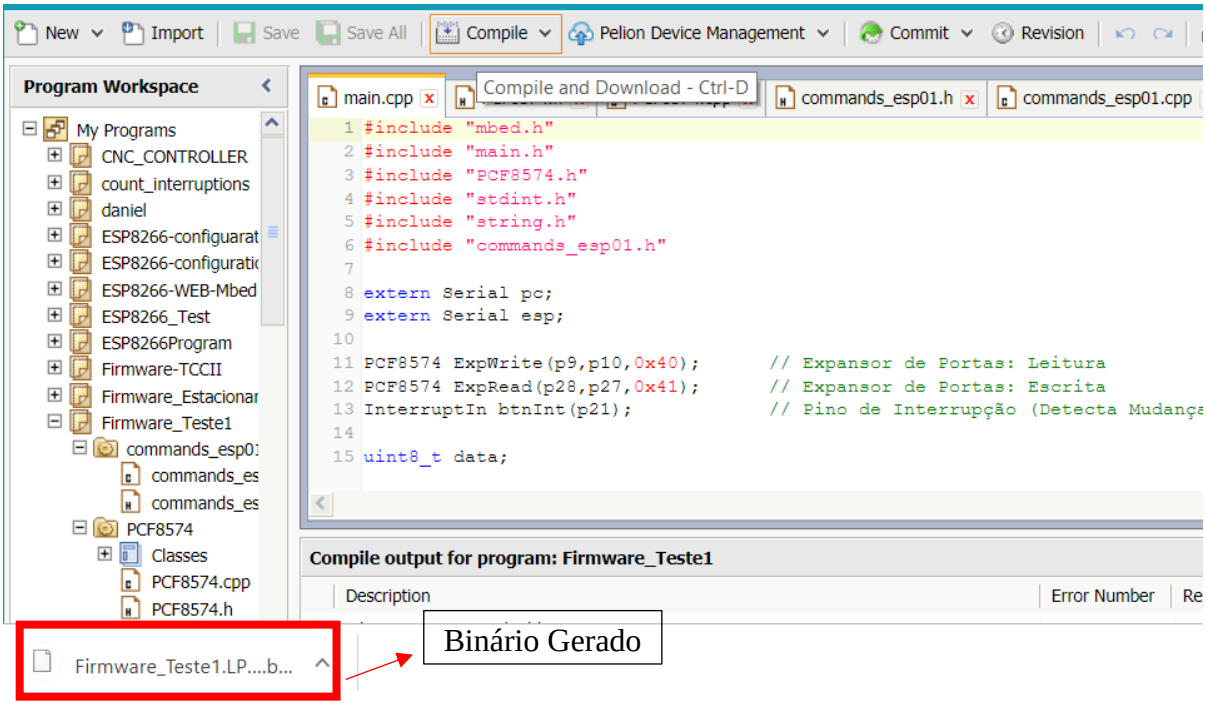
Depois de desenvolver o código, deve-se gerar o arquivo para embarcar no microcontrolador, chamado de arquivo binário. O arquivo binário é gerado pelo compilador e é iniciado a geração desse arquivo ao clicar no item *Compile* na barra de ferramentas da IDE, conforme é mostrado na figura 106.

Figura 107 - Trecho da Função Principal do Firmware.

```
while(1)
{
    switch(state)
    {
        case HARDWARE_RESET:
            pc.printf("HARDWARE_RESET\r");
            ESPHardwareReset();
            state = SET_BAUDRATE;
        break;
        case SET_BAUDRATE:
            pc.printf("SET_BAUDRATE\r");
            ESPsetbaudrate();
            state = CONFIG;
        break;
        case CONFIG:
            pc.printf("CONFIG\r");
            ESPconfig();
            state = ESTABLISHING_CONNECTION;
        break;
        case ESTABLISHING_CONNECTION:
            establishing_connection();
            data = ExpRead.read();
            ExpWrite.write(data);
            sendByteToDB();
            state = IDLE;
        break;
        case CLOSING_CONNECTION:
            pc.printf("CLOSING_CONNECTION\r");
            close_connection();
            state = IDLE;
        break;
        case IDLE:
        default:
            data = ExpRead.read();
            ExpWrite.write(data);
        break;
```

Fonte: Autor.

Figura 106 - Gerando o binário.



Fonte: Autor.

4.6. SERVIDOR WEB

Para o desenvolvimento de um servidor web, utilizou-se o pacote de software chamado WAMP que é um acrônimo para Windows, Apache, MySQL e PHP. Esse pacote de programas permite que o computador se torne um servidor.

É fornecido todas as ferramentas necessárias para o funcionamento do servidor, como:

Apache: permite o desenvolvimento de aplicações Web. É responsável por receber requisições e enviar as respostas ao solicitante que é o cliente. É a interface entre o usuário e o banco de dados.

MySQL: é um Sistema de Gerenciamento de Banco de Dados relacional que utiliza a linguagem padrão SQL (Structured Query Language) (FREITAS, 2016).

PHP: é a linguagem que interpreta a página php solicitada pelo usuário. Uma característica principal dessa ferramenta é a sua portabilidade, ou seja, funciona em qualquer sistema operacional, seja Windows, Linux ou até mesmo Unix. (FREITAS, 2016).

PHPMyadmin: é uma página web que disponibiliza interface gráfica para manipular facilmente o banco de dados (CENTENARO, 2014).

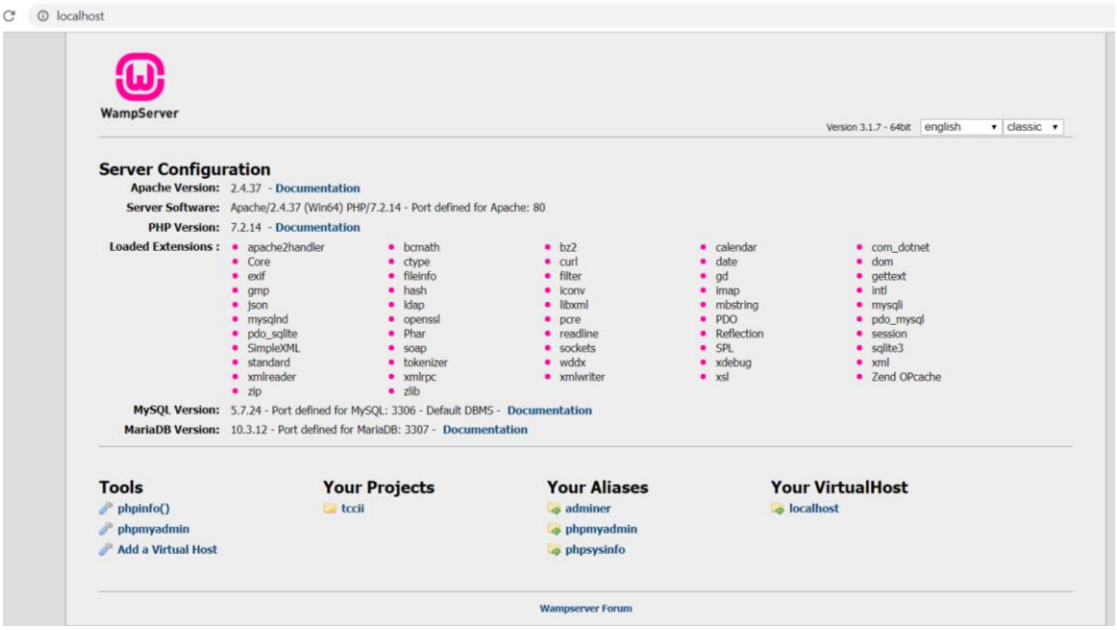
O WAMP é uma ferramenta gratuita disponibilizada para download no site oficial do WAMPSERVER que é http://www.wampserver.com/en/. O site também fornece os passos que devem ser realizados para a instalação do software. E são eles:

Duplo clique no executável que foi baixado do site e deve seguir as instruções para a instalação.

Após a instalação, deve aparecer um ícone na barra de tarefas do Windows. Caso o ícone fique verde, significa que o WampServer está online e já está funcionando. Caso esteja laranja, deve-se clicar com o botão esquerdo do mouse e escolher a opção Put Online.

Depois de colocar o WAMP no modo online o link http://localhost e será apresentado a seguinte tela:

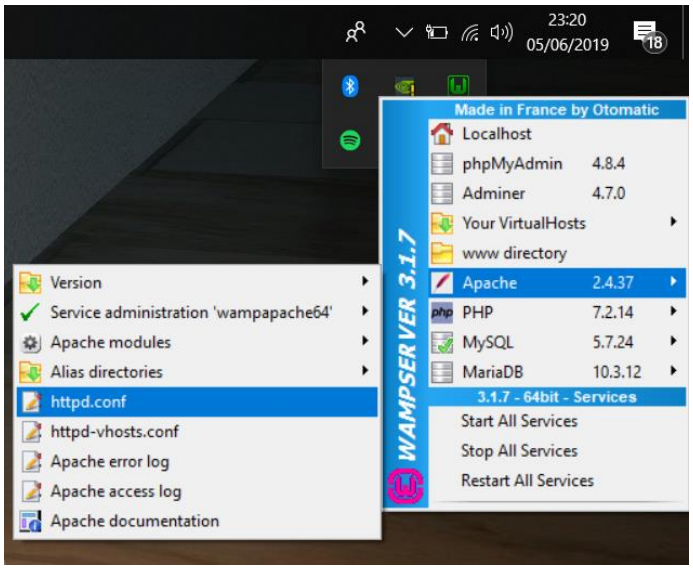
Figura 108 - LOCALHOST.



Fonte: Autor.

Agora, deve-se tornar o WampServer acessível a todos os componentes da rede local e para que isso aconteça, algumas configurações tem que ser realizadas. Com o botão esquerdo do mouse, deve-se clicar no ícone do WampServer na barra de tarefas do Windows e procurar o item httpd.conf no Apache, conforme a figura 109.

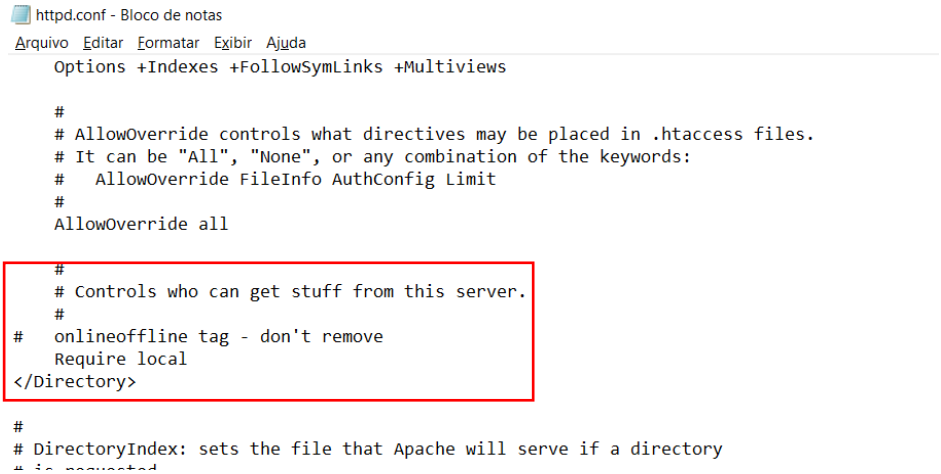
Figura 109 - WAMP na Barra de Tarefas.



Fonte: Autor.

O arquivo selecionado será aberto em um editor de texto de sua preferência. No arquivo de configuração, deve-se procurar o trecho destacado na figura.

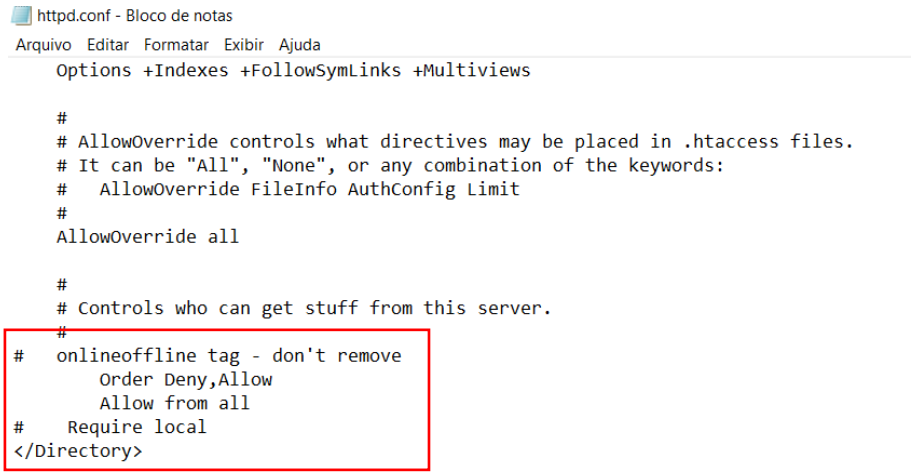
Figura 111 – Trecho do arquivo http.conf a ser editado.



Fonte: Autor.

E editar o trecho destacado conforme a figura 111 apresentada:

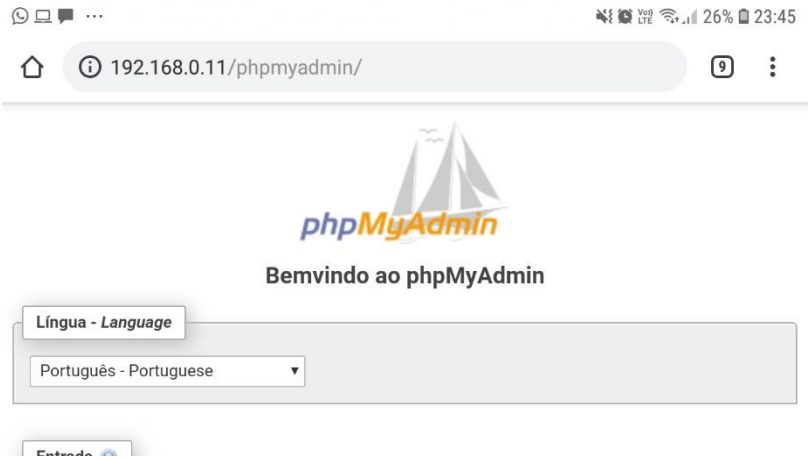
Figura 110 - Editando o arquivo httpd.conf



Fonte: Autor.

Agora, deve-se executar o “Restart All Services” para que as alterações possam surtir efeito. Para verificar se tudo ocorreu conforme o esperado, deve-se descobrir o IP do servidor e por meio da barra de endereços de um navegador de outro elemento conectado a rede local, como um celular, deve-se digitar o link no formato http://ipdoservidor:80/phpmyadmin. Se a página for carregada, está tudo cero. Caso contrário, aconteceu algum erro durante o processo. A resposta esperada é:

Figura 112 - Teste do servidor via *smartphone*.

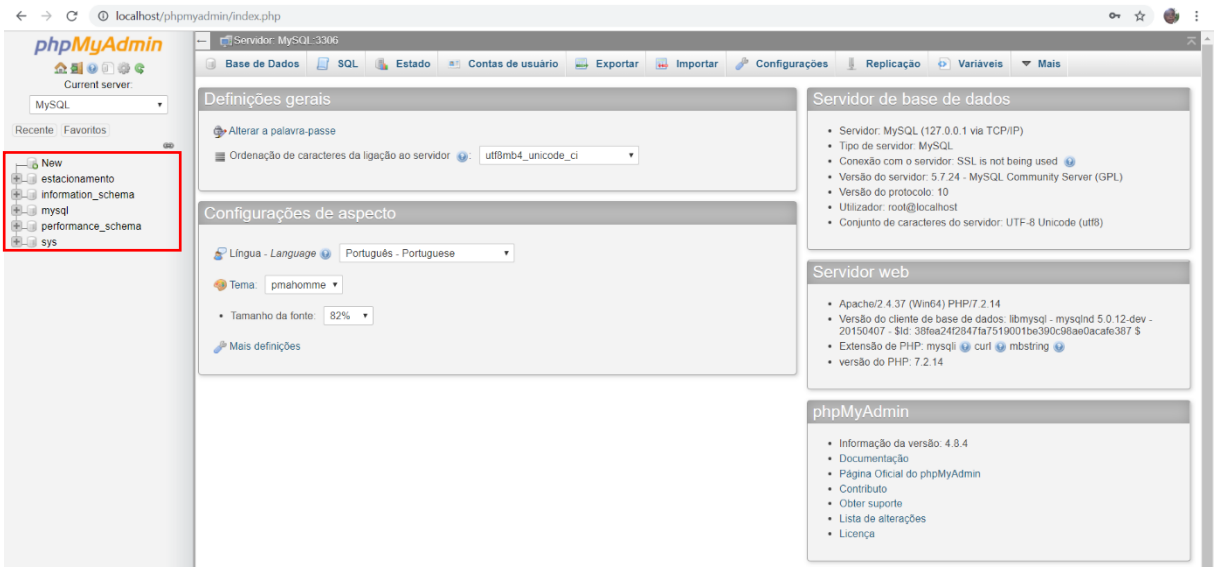


Fonte: Autor.

Após certificar de que o WampServer está aberto para outros dispositivos conectados na mesma rede, deve-se entrar no phpmyadmin com usuário “root” e sem senha para realizar a criação do banco de dados.

Quando é realizado o login com sucesso, é disponibilizado a seguinte página:

Figura 113 - Ambiente PHPMyadmin.



Fonte: Autor.

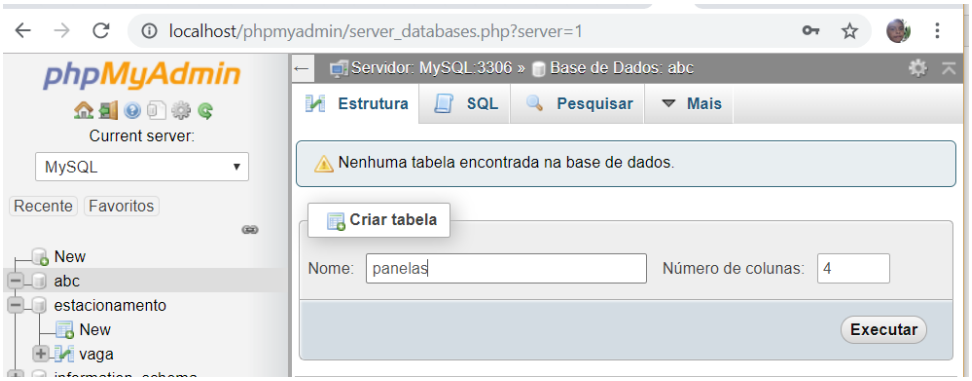
A parte em destaque na figura 113 mostra o item “New” que quando é clicado cria um novo banco de dados e mostra a seguinte tela:

Figura 115 - Criando BD.

Fonte: Autor.

A figura acima é uma janela que auxilia na criação de bancos de dados. Depois de nomear o BD, deve-se apertar o botão de “criar”. O banco de dados criado é posicionado abaixo do item “New” no canto esquerdo da página. E seguinte tela é apresentada:

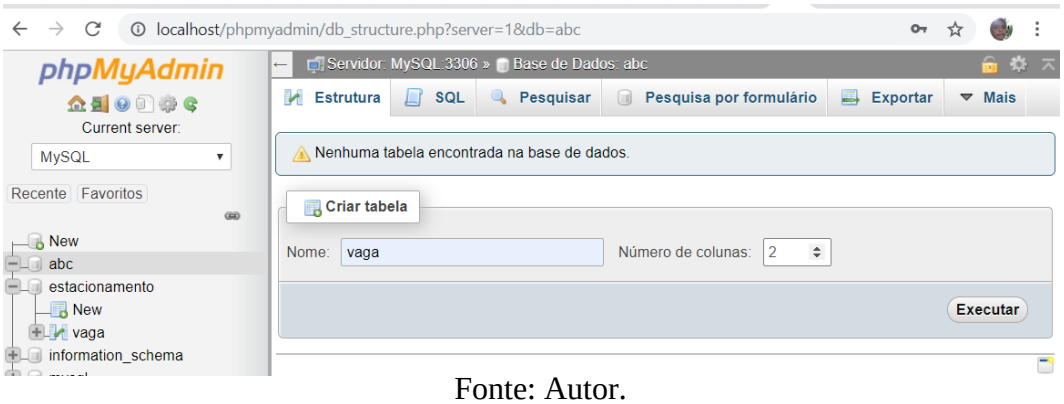
Figura 114 - Janela de Criação de Tabela.



Fonte: Autor.

A Figura 114, solicita que seja fornecido um nome para a tabela que será criada dentro do BD e ao lado deve ser configurado a quantidade de coluna que ela possuirá.

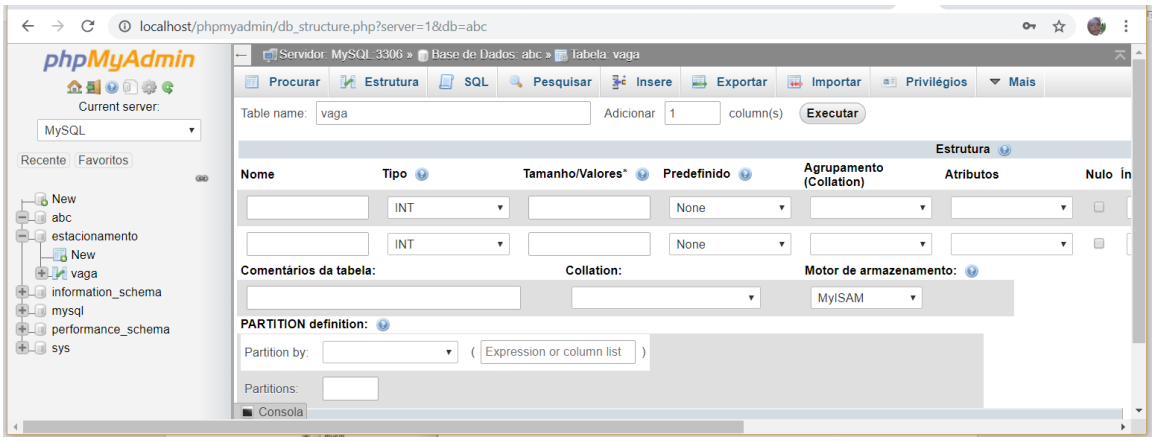
Figura 117 - Nomeando o BD.



Fonte: Autor.

Definido o nome da tabela e a quantidade de colunas, as colunas são nomeadas na próxima página apresentada na Figura 116.

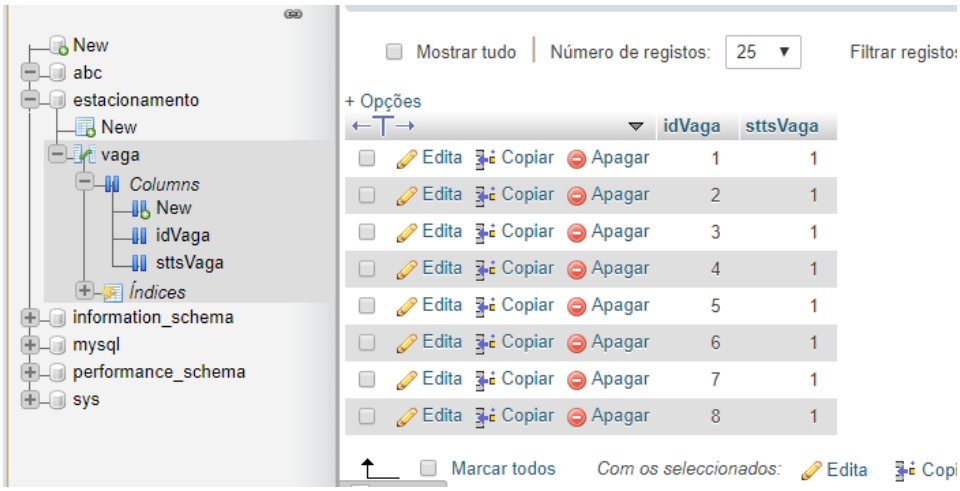
Figura 116 - Definindo Objetos da tabela.



Fonte: Autor.

O sistema proposto necessita apenas de duas colunas, o número identificador da vaga e o status da vaga. O número identificador é AUTO_INCREMENT, ou seja, cada vaga terá um identificador e além disso, outra coluna, deverá armazenar o status de cada vaga cadastrada no ambiente. Foi criado um banco de dados com duas coluna e lugares para até oito elementos. O banco de dados fornecido foi:

Figura 118 - Banco de Dados criado.



Fonte: Autor.

E desta maneira foi criado o banco de dados do sistema. Para manipular este banco o servidor web, Apache, deve receber requisições por meio do protocolo HTTP solicitando páginas em php, buscando-as e interpretando-as.

Para este projeto em questão foi desenvolvidos duas páginas em php: a que atualiza os dados do banco que será solicitado do servidor pelo microcontrolador do kit de desenvolvimento e outra página que fornece informações do banco no formato JSON, que será utilizada pelo aplicativo.

Para criar a página atualizar.php, que será responsável por atualizar os status das vagas, utilizou um editor de texto chamado notepad++ que tem suporte para a linguagem php. As características que esta página deve ter é a de receber dados do microcontrolador por meio do método HTTP Get, conectar-se ao banco de dados, enviar os dados recebidos do microcontrolador e fechar a conexão com o banco. Os relatos a seguir mostrarão os trechos que corresponde a cada uma das etapas citadas.

As configurações necessárias para a conexão com o banco de dados são: nome do servidor (“localhost”), nome do usuário (“root”), senha do wampserver (“kmfa.eng’) e o banco de dados que se quer manipular (“estacionamento”). E todos estes dados são armazenadas nas variáveis a seguir:

Figura 120 - Variáveis que armazenam os dados do BD.

```
5   $servername = "localhost";
6   $username = "root";
7   $password = "kmfa.eng";
8   $dbname = "estacionamento";
```

Fonte: Autor.

Para receber o parâmetro enviado pelo microcontrolador, foi declarado a variável STATUS_PCF que utilizando a função $_GET[nome da variável informada no método HTTP GET] receberá esse número em hexadecimal, conforme repassado pela estrutura HTTP GET. E uma outra variável foi inicializada para receber o valor equivalente do STATUS_PCF em decimal que é a $valDec que utiliza a função hexdec.

Figura 119 - Armazenando os dados recebidos.

```
9
10  $STATUS_PCF = $_GET['PCF'];
11
12  $valDec = hexdec( $STATUS_PCF );
13
```

Fonte: Autor.

O trecho apresentado na figura 119, apresenta a função que realiza a conexão com o Banco de Dados.

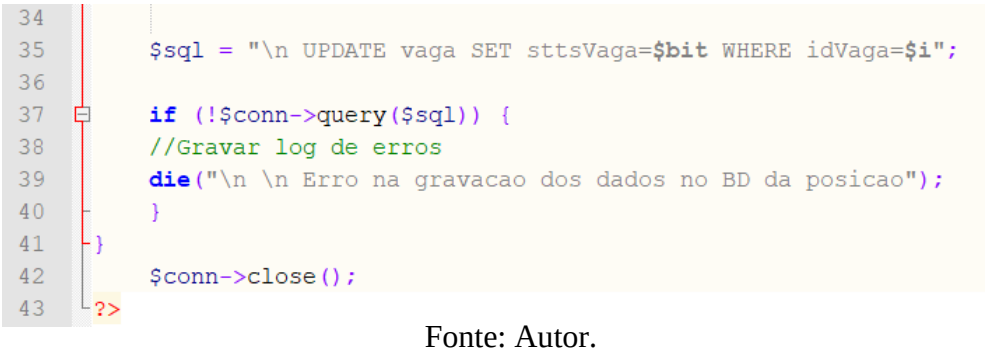
Figura 121 - Função de Conexão com o BD.

```
15
16  $conn = new mysqli($servername,$username,$password,$dbname);
17
```

Fonte: Autor.

A linha 35 do código mostra a estrutura da requisição montada em SQL e a linha 37 envia essa estrutura para o banco de dados que realiza as suas funções de acordo com a requisição enviada.

Figura 123 - Envio da requisição.

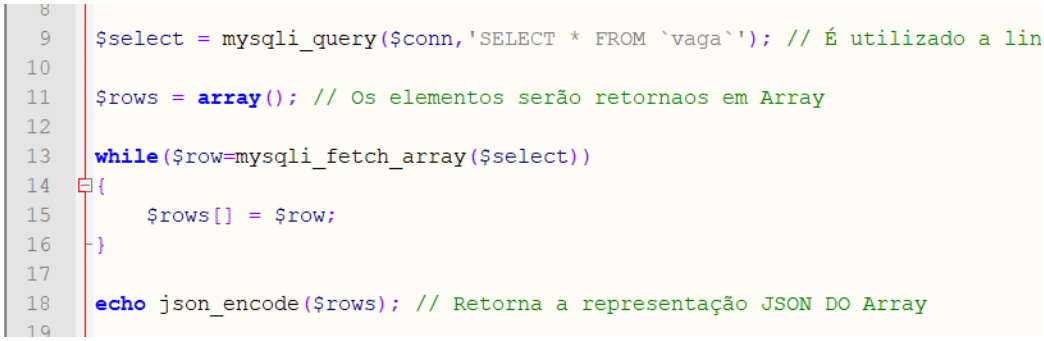


```
34
35      $sql = "\n UPDATE vaga SET sttsVaga=$bit WHERE idVaga=$i";
36
37      if (!$conn->query($sql)) {
38      //Gravar log de erros
39      die("\n \n Erro na gravacao dos dados no BD da posicao");
40      }
41  }
42      $conn->close();
43  ?>
```

Fonte: Autor.

O próximo código define a página que transcreverá os dados do banco de dados para o formato JSON. Da mesma maneira como ocorreu na página **atualizar.php**, a página **format_json.php** deverá conectar-se ao banco e além disso, buscar os elementos utilizando a linguagem SQL, armazená-los em um vetor e convertê-los para serem apresentados no formato JSON. A grande diferença desse código para o anterior é o trecho a seguir:

Figura 122 - Trecho do código



```
9   $select = mysqli_query($conn,'SELECT * FROM `vaga`'); // É utilizado a lin
10
11  $rows = array(); // Os elementos serão retornaos em Array
12
13  while($row=mysqli_fetch_array($select))
14  {
15      $rows[] = $row;
16  }
17
18  echo json_encode($rows); // Retorna a representação JSON DO Array
```

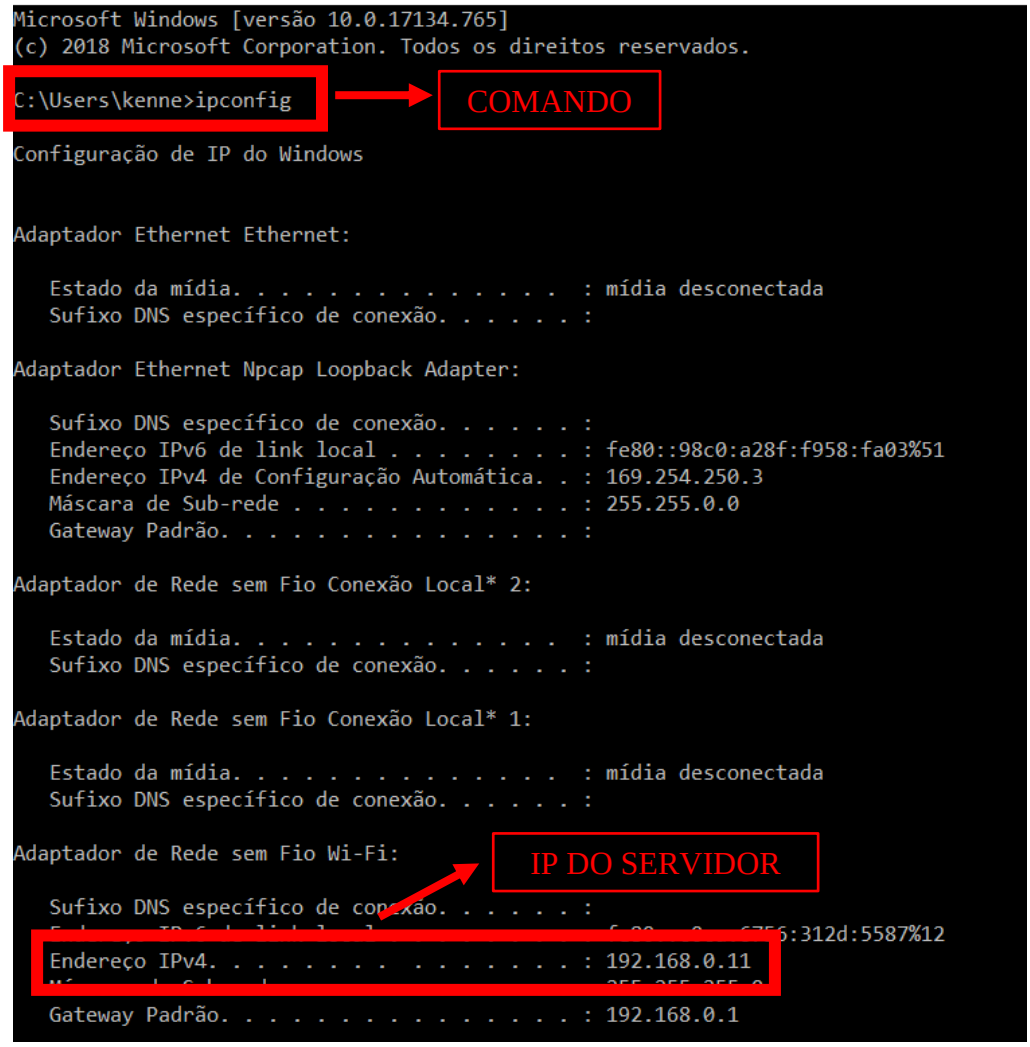
Fonte: Autor.

A linha nove possui a função mysqli_query(). Esta função realiza o envio de requisições SQL ao banco de dados, sendo que o primeiro parâmetro passado é a variável que fez a conexão com o banco de dados e o segundo parâmetro é a requisição. A requisição enviada é a de consultar todos os dados do banco em questão. Os dados serão armazenados em forma de array na variável $rows. A linha dezoito converte os dados recuperados para o formato JSON.

A criação dos arquivos acima, atualizar.php e format_json.php, podem estar funcionais, mas para que sejam integrados no servidor web, eles devem estar armazenados no diretório C:\wamp64\apps\phpmyadmin4.8.4.

Para acessar essas páginas por meio de navegadores, deve-se descobrir o IP da máquina que possui instalado o servidor web em questão. A identificação da máquina na rede pode ser encontrada usando o prompt de comandos do Windows digitando o comando ipconfig, conforme mostrado na figura 124.

Figura 124 - Descobrindo o IP do servidor.



Fonte: Autor.

O IP do servidor é 192.168.0.11 é informação muito importante do projeto. Pois para ter acesso as páginas criadas, a URL principal será o http://192.168.0.11/ e os complementos para acessar as páginas criadas serão /phpmyadmin/atualizar.php e /phpmyadmin/format_json.php.

4.7. APLICATIVO ANDROID

A plataforma Android é um sistema operacional desenvolvido pela empresa Google para smartphones, o qual foi baseado no núcleo do Linux. Atualmente, é o sistema operacional mais

utilizado no mundo e está presente em diversos aparelhos de diversas marcas. (SILVEIRA, 2016)

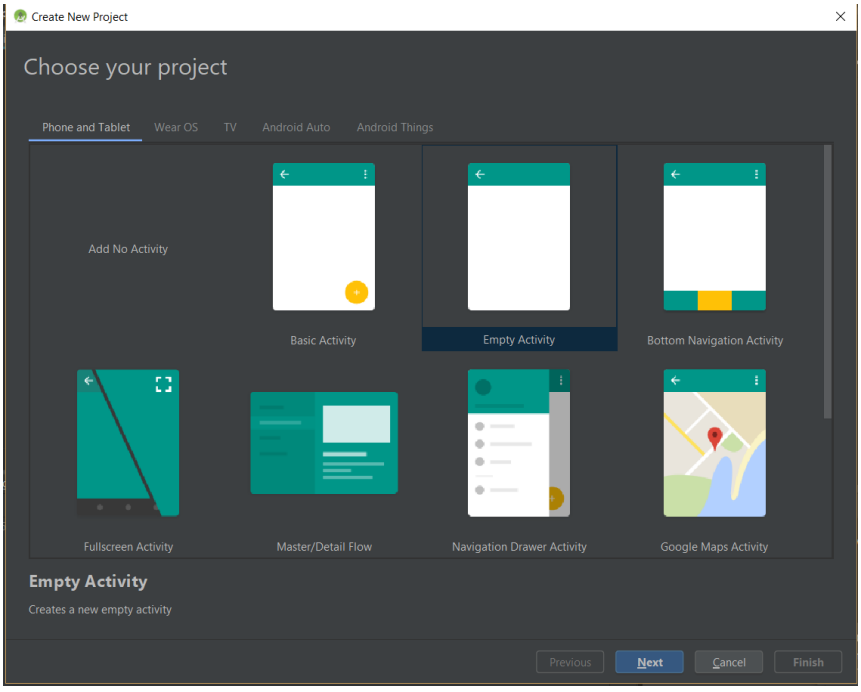
O funcionamento dessa plataforma é similar a outros sistemas operacionais e tem como função o gerenciamento dos processos dos aplicativos e hardware do dispositivo para que funcione de maneira correta e esperada, fornecendo ao usuário uma interface visual capaz de prover interação com o sistema eletrônico (SILVEIRA, 2016).

A aplicação android que será desenvolvida para o projeto tem o objetivo de recuperar a informação das sete vagas de um estacionamento, representado neste projeto por uma maquete, e apresentar a informação para o usuário final.

E uma das IDE's fornecidas para o desenvolvimento de aplicativo android é o Android Studio. Ele possui ferramentas que aumentam a produtividade na criação dos aplicativos, os quais destacam-se: um sistema de compilação flexível baseado no Gradle, um emulador, entre outros. (Android Studio, 2017) E por isso, essa IDE foi escolhida para o desenvolvimento do aplicativo do sistema.

Durante a criação de um novo projeto, foi decidido criar uma activity vazia.

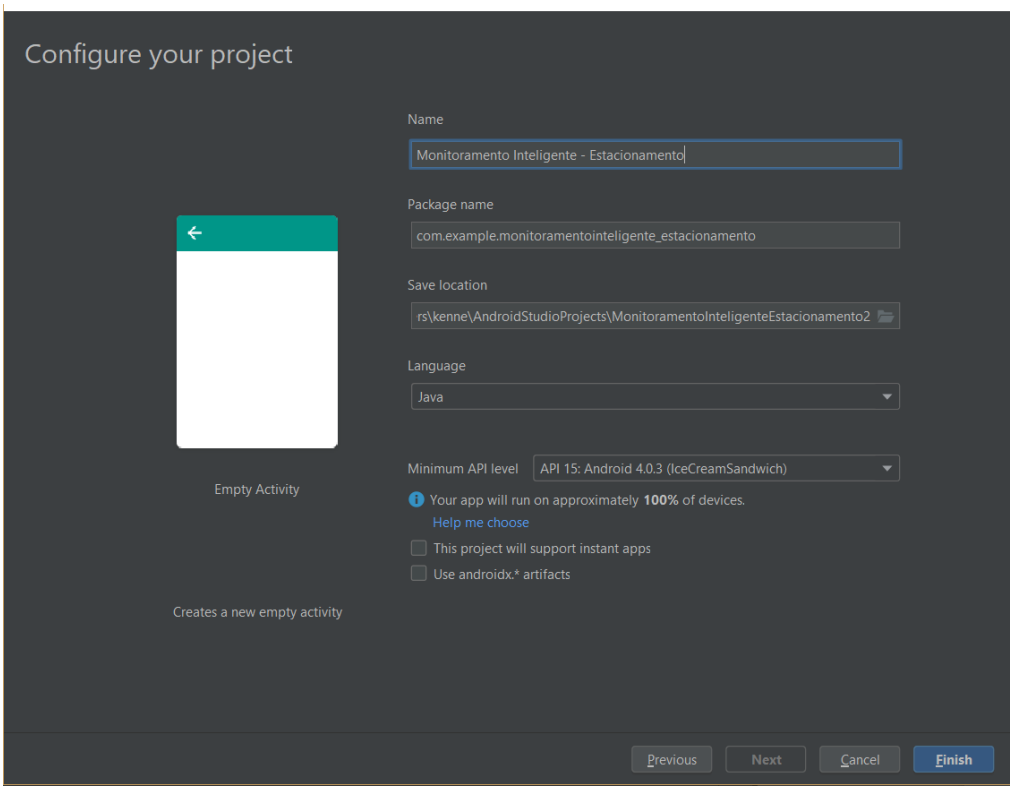
Figura 125 - Selecionando uma acitvity.



Fonte: Autor.

E logo após, foi configurado alguns parâmetros do aplicativo, como: nome, o diretório o qual o projeto será salvo, a linguagem utilizada e API mínima suportada.

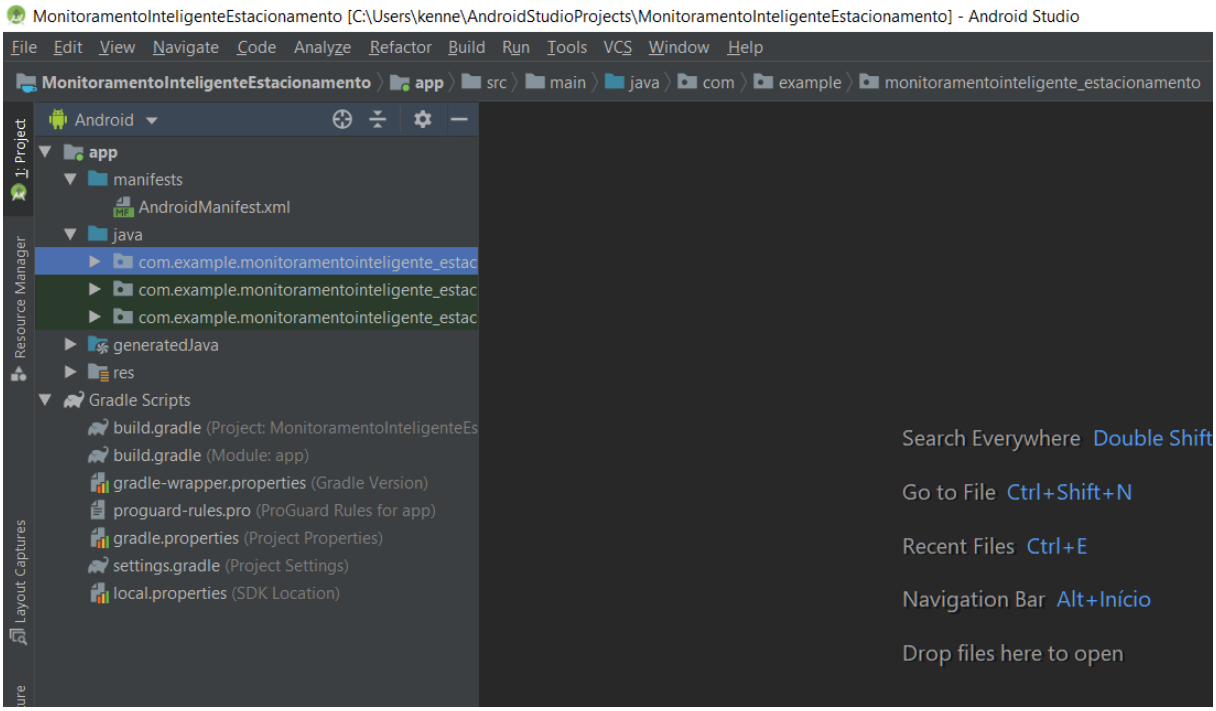
Figura 126 - Definindo parâmetros para o projeto.



Fonte: Autor.

Depois de confirmar os parâmetros de configurações iniciais, serão gerados arquivos padrões para a criação do aplicativo, conforme a Figura 127.

Figura 127 - Interface Gráfica do Android Studio.



Fonte: Autor.

Os principais arquivos gerados na criação do projeto a serem editados são: AndroidManifest.xml, activity_main.xml, build.gradle (Module:app) e a MainActivity. E os arquivos que deverão ser criados são: iRetrofitEstacionamento e o Vagas.java.

Criando classe para receber dados

Deve-se criar uma classe em java para receber os dados convertidos da string JSON. E por isso as variáveis devem possuir o mesmo nome das variáveis que se deseja obter. Neste caso, os dados importantes para a aplicação são: o idVaga que é o número que identifica a vaga e o sttsVaga que informa o estado atual da vaga. A classe criada deve estar no mesmo diretório que a MainActivity.java. E o código que define a classe é:

```
public class Vagas {
        int idVaga;
        int sttsVaga;
}
```

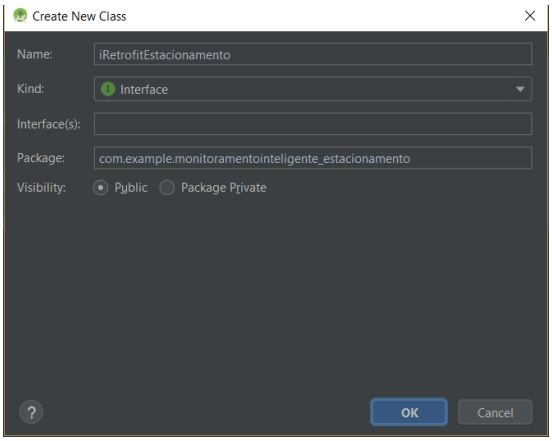
4.7.1. Biblioteca Retrofit

Para utilizar os recursos da biblioteca Retrofit, deve-se primeiramente declarar três dependências no arquivo Build.Gradle e assim deve ser baixado e sincronizado a biblioteca no projeto. As dependêsncias são:

```
implementation 'com.google.code.gson:gson:2.8.2'
implementation 'com.squareup.retrofit2:retrofit:2.0.2'
implementation 'com.squareup.retrofit2:converter-gson:2.0.2'
```

Para iniciar os trabalhos com o Retrofit, deve-se criar uma nova classe java do tipo Interface. O arquivo deve ser criado no mesmo diretório da MainActivity.java. A janela de criação do arquivo, segue conforme a figura 128.

Figura 128 - Janela de Criação de Interface.



Fonte: Autor.

O uso do Retrofit é realizado de maneira eficiente e simples. Ele realiza requisições HTTP e converte em interface java. Essa conversão é realizada através do seguinte código disponibilizado no site oficial da biblioteca:

```
public interface GitHubService {
        @GET("users/{user}/repos")
        Call<List<Repo>> listRepos(@Path("user") String user);
        Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.github.com")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
}
```

Adaptando o código acima para implementar no projeto em questão, foram realizadas mudanças, e o trecho adaptado para esta aplicação resultou em:

```
public interface iRetrofitEstacionamento {
   @GET("/phpmyadmin/{urlVagas}/")
   Call<List<Vagas>> getVagas(@Path("urlVagas") String Vagas);
   public static final Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://192.168.0.11/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    }
```

O código desenvolvido acima realiza requisições ao servidor web solicita os dados do servidor. E para isso, é utilizado o trecho: @GET("/phpmyadmin/{urlVagas}/").

A linha “Call<List<Vagas>> getVagas(@Path("urlVagas") String Vagas);” retorna os dados solicitados da urlVagas especificada e os aloca em uma lista de objetos do tipo Vagas.

O trecho:

“public static final Retrofit retrofit = new Retrofit.Builder()

.baseUrl("http://192.168.0.11/")

.addConverterFactory(GsonConverterFactory.create())

.build();

}

define os atributos do objeto retrofit. Os atributos passados são a URL base e a biblioteca responsável por converter a string JSON em objetos equivalentes em java.

AndroidManifest.xml

Conforme o site oficial do Android, disponibilizada para desenvolvedores, relata que o Android Manifest apresenta informações essenciais sobre o aplicativo ao sistema Android, necessárias para o sistema antes que ele possa executar o código do aplicativo. Além disso, o arquivo em questão serve para:

Nomear o pacote java para o aplicativo que serve como identificador exclusivo para o aplicativo;

Informar sobre as atividades, serviços, receptores de transmissão e provedores de conteúdo que compõem o aplicativo.

Declarar as permissões que o aplicativo deve ter para acessar partes protegidas da API e interagir com outros aplicativos. Ele também declara as permissões que outros devem ter para interagir com os componentes do aplicativo.

Como o projeto em questão utiliza o wifi para se conectar a rede local, deve-se permitir o uso da Internet para isso. Portanto, deve-se inserir o trecho a seguir no arquivo AndroidManifest.xml:

<uses-permission android:name="android.permission.INTERNET" />.

Então, o código implementado nesse arquivo será conforme apresentado a seguir:

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.monitoramentointeligente_estacionamento">

<uses-permission android:name="android.permission.INTERNET" />

<application

```
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity android:name=".MainActivity"
        android:screenOrientation="portrait">
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
</application>

</manifest>
```

4.7.2. activity_main.xml

Este arquivo está alocado no diretório “Layout” do projeto criado. Na activity_main.xml podemos definir como a tela será montada, permitindo a adição de botões, textos de visualização, entre outros componentes, além de permitir a identificação e alteração dos parâmetros de configurações dos componentes. A seguir é apresentado duas maneiras de se definir o layout das telas e os parâmetros de configuração dos componentes: Design e Text.

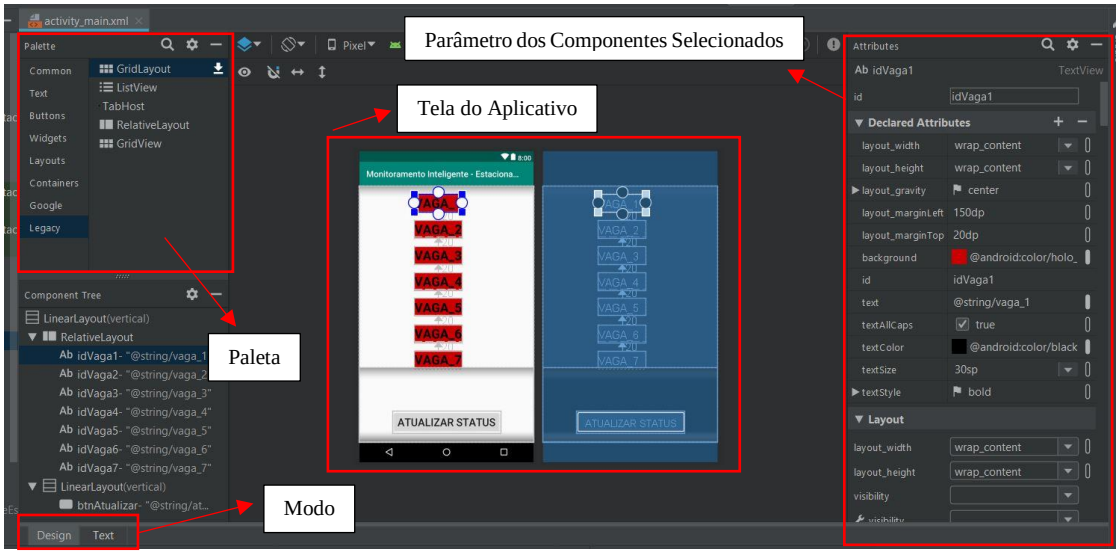
Modo Design

Neste modo, é disponibilizado uma tela de interface para inserção de componentes e alteração do layout para a interação com o usuário. É fornecido, também, uma paleta de componentes composto por: Button, Image View, Recycler View, Scroll View, etc. Para inserir os componentes na tela, basta clicar no componente escolhido e arrastar até o destino final.

Os principais componentes utilizados no aplicativo do projeto são o TextView e o Button. O TextView é um elemento de interação que exibe texto ao usuário, enquanto o Button, além de apresentar texto, permite a geração de eventos quando o usuário clica nele. Ambos são componentes totalmente flexíveis permitindo a mudança no texto apresentado, tamanho do componente, cor da fonte, cor de background e etc.

E com isso, para o desenvolvimento do aplicativo do sistema, foram selecionados oito componentes para a tela, sendo: sete componentes do tipo TextView, que será utilizado para apresentar o status das sete vagas disponibilizadas no estacionamento representado pela maquete, e um elemento do tipo Button que será responsável por atualizar os status das vagas, conforme a figura apresentada. Após a inserção dos componentes, a tela de interface com usuário apresentará o seguinte aspecto:

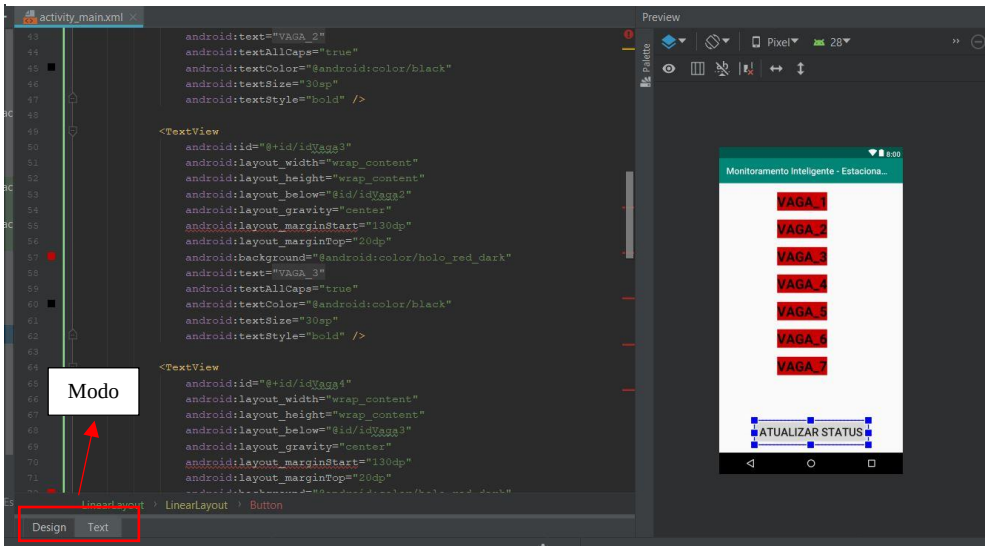
Figura 129 - Ambiente para Desenvolver a Interface do Aplicativo.



Fonte: Autor.

Quando são adicionados componentes na tela, são gerados textos que os representam na activity_main.xml no modo texto, conforme a Figura 130.

Figura 130 - Textos representando a Interface.



Fonte: Autor.

Dentre as principais informações editadas no modo Text da activity_main.xml, vale destacar os identificadores e os textos dos componentes utilizados. A identificação e o texto utilizados nos TextView são, respectivamente:

android:id="@+id/idVaga1" ; android:text="@string/vaga_1"
android:id="@+id/idVaga2" ; android:text="@string/vaga_2"
android:id="@+id/idVaga3" ; android:text="@string/vaga_3"
android:id="@+id/idVaga4" ; android:text="@string/vaga_4"
android:id="@+id/idVaga5" ; android:text="@string/vaga_5"
android:id="@+id/idVaga6" ; android:text="@string/vaga_6"
android:id="@+id/idVaga7" ; android:text="@string/vaga_7"

E no Button:

android:id="@+id/btnAtualizar" ; android:text="@string/atualizar_status"

4.7.3. Build Gradle

O Build Gradle é um kit de ferramentas de compilação avançado utilizado pelo Android Studio para automatizar e gerenciar o processo de compilação permitindo que as configurações de compilação sejam personalizadas e flexíveis. Cada configuração de compilação pode definir seu próprio conjunto de códigos e recursos, reutilizando as partes comuns a todas as versões do aplicativo (ANDROID DEVELOPER, s.p.).

O sistema de compilação gerencia as dependências do projeto através do sistema de arquivos locais ou de repositórios remotos o que evita a necessidade de pesquisar e realizar downloads e copiar manualmente os pacotes binários das suas dependências para o diretório do projeto (ANDROID DEVELOPER, s.p.).

Para que o aplicativo possa buscar informações no servidor web, será necessário a utilização da biblioteca Retrofit. Pois ela fornece funções que realizam a conexão com o servidor e extração das informações da página format_json.php que disponibiliza as informações no formato JSON. Para utilizar os recursos dessa biblioteca, adicionou-se as suas dependências no arquivo Buil.Gradle (Module:app). As dependências são:

implementation 'com.squareup.retrofit2:retrofit:2.0.2'
implementation 'com.squareup.retrofit2:converter-gson:2.0.2'
implementation 'com.android.support:support-annotations:28.0.0'

Desta maneira o trecho que especifica as dependências do projeto fica desta maneira:

dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])

implementation 'com.android.support:appcompat-v7:28.0.0'

implementation 'com.android.support.constraint:constraint-layout:1.1.3'

testImplementation 'junit:junit:4.12'

androidTestImplementation 'com.android.support.test:runner:1.0.2'

androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

implementation 'com.google.code.gson:gson:2.8.2'

implementation 'com.squareup.retrofit2:retrofit:2.0.2'

implementation 'com.squareup.retrofit2:converter-gson:2.0.2'

implementation 'com.android.support:support-annotations:28.0.0'

}

4.7.4. MainActivity.java

É um dos arquivos gerado automaticamente na criação do projeto. A função desse arquivo é a mesma função do arquivo que contém o método “main” em aplicativos desenvolvidos na Linguagem Java (CURY, 2015).

A MainActivity.java é a classe principal do projeto e já foi declarada anteriormente em AndroidManifest.xml como a principal atividade do aplicativo, em:

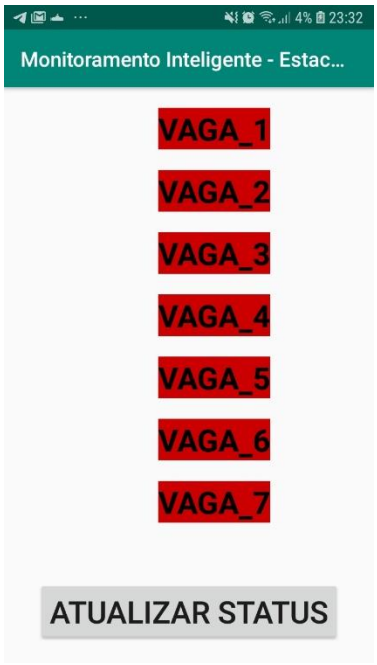
<activity android:name=".MainActivity"

Ela representa a tela inicial da aplicação que foi definida em activity_main.xml. É responsável por controlar o estado e os eventos da tela.

Então, nesta aplicação, será responsável por mostrar mensagens na tela do aplicativo, requisitar informações do banco através da detecção de eventos gerados pelo clique do botão e pelas mudanças que ocorrem nas configurações dos TextView, tais como, cor de background para representar o status das vagas.

Após o desenvolvimento e implementação dos códigos citados acima, gera-se o APK que é o arquivo de instalação do aplicativo e o instala em um smartphone. O resultado foi o seguinte:

Figura 131 – Print da Tela do celular com o aplicativo de Monitoramento de Vagas.



Fonte: Autor.

5. RESULTADOS OBTIDOS

Este capítulo descreve os testes individuais de cada elemento do sistema, tais como: o hardware, firmware, servidor web e aplicativo android. É relatado, também, os resultados obtidos com a integração dos elementos do sistema e trabalhos futuros.

5.1. TESTES INDIVIDUAIS

Inicialmente foram realizados testes específicos em cada placa confeccionada, testes do firmware, teste do servidor web e aplicativo android antes de integrá-los no sistema.

5.1.1. HARDWARE

Os testes foram realizados afim de exercitar todas as funcionalidades que cada placa deve fornecer ao sistema.

As funcionalidades testadas na placa de controle foram:

Alimentação do kit de desenvolvimento e do ESP8266;

Teste de comunicação serial com o modulo wifi;

Testes do barramento I2C.

Para validar os testes acima, foi utilizado multiteste para verificar a tensão nos pinos de alimentação do kit de desenvolvimento e do ESP8266-01, foram desenvolvidos firmwares de teste para enviar comandos AT para o módulo wifi e utilizar terminal serial, chamado PUTTY, para checar as mensagens. E para os dois barramentos, utilizou-se a protoboard com PCF para escrever e ler informações do CI.

Os testes realizados na placa de leitura foram:

- Teste de alimentação da placa, do PCF e do barramento I²C;
- Teste de alimentação dos terminais dos sensores;
- Realização de leitura individuais de cada terminal disponibilizado para cada sensor.

Para a checagem das alimentações desta placa, foram adotados os mesmos procedimentos realizados com a placa de controle. E para os testes individuais de cada terminal da placa disponibilizados para conexão dos sensores, foi utilizado um dos sensores utilizados no projeto para ser conectado em um terminal por vez.

- As funcionalidades testadas na placa de escrita foram:
- Teste das alimentações nas fontes externas, do barramento I2C e dos CI’s da placa;

- Teste dos terminais de endereçamento do PCF;
- Teste de alimentações dos terminais de saída para a placa do led;
- Teste de acionamento dos relés.

O teste das alimentações, ocorreram da mesma maneira que nas placas anteriores. Quanto ao acionamento dos relés, utilizou-se uma placa do led para conectar em cada um dos terminais disponibilizados para eles.

Após a realizações dos testes citados acima, identificou-se que as placas de controle e a placa do led, por serem simples e terem poucos componentes, não apresentaram problemas. No entanto, as placas de escrita e leitura apresentaram problemas em suas funcionalidades devido a erros de soldagem, ocasionada por curtos e mal contato entre a solda derretida e os terminais dos componentes.

Após a identificação dos problemas foram realizadas as correções necessárias para que as placas estivessem totalmente funcionais para a integração no sistema.

5.1.2. FIRMWARE

Após a validação do hardware, pôde-se testar o firmware do sistema conforme o diagrama de estados proposto. Durante os testes, percebeu-se que o firmware que utilizava o código exemplo mostrado na figura demorava cerca de 57 segundos para completas os Pois todas as funcionalidades das placas foram corrigidas e validadas e estão disponíveis para o software.

O teste realizado foi realizado no firmware, que foi desenvolvido baseado no diagrama de estados proposto. Os testes ocorreram da forma esperada e não apresentou problemas na sua implementação.

5.1.3. SERVIDOR WEB

Para testar o servidor web, foi necessário utilizar um smartphone conectados à rede local para fazer requisições ao banco de dados. Utilizando o navegador de internet do dispositivo móvel, digitou-se na barra de endereços, um link no seguinte formato: http://{IP.DA.MÁQUINA}/phpmyadmin/{nome_da_página_criada}.php.

Para o projeto, foram desenvolvidas duas páginas, que são: **atualizar.php** que é responsável por atualizar os dados do BD e a página **format_json.php** que apresenta os dados do BD no formato JSON. E montando o link para a recuperação das informações do BD, conforme as informações do projeto, resultou em: http://192.168.0.1/phpmyadmin/format_json.php. Digitou-se o link na barra de endereços do navegador e obteve-se o seguinte, retorno:

Figura 132 - Informações Recuperadas do Banco de Dados no formato JSON através de um *smartphone*.

Fonte: Autor.

Comparando o retorno do procedimento relatado acima com as informações do BD, conforme apresentado na Figura 133, conclui-se que a recuperação de dados da página criada, **format_json.php**, está funcionando de acordo com o esperado.

Figura 133 - Informações no Banco de Dados.

Fonte: Autor.

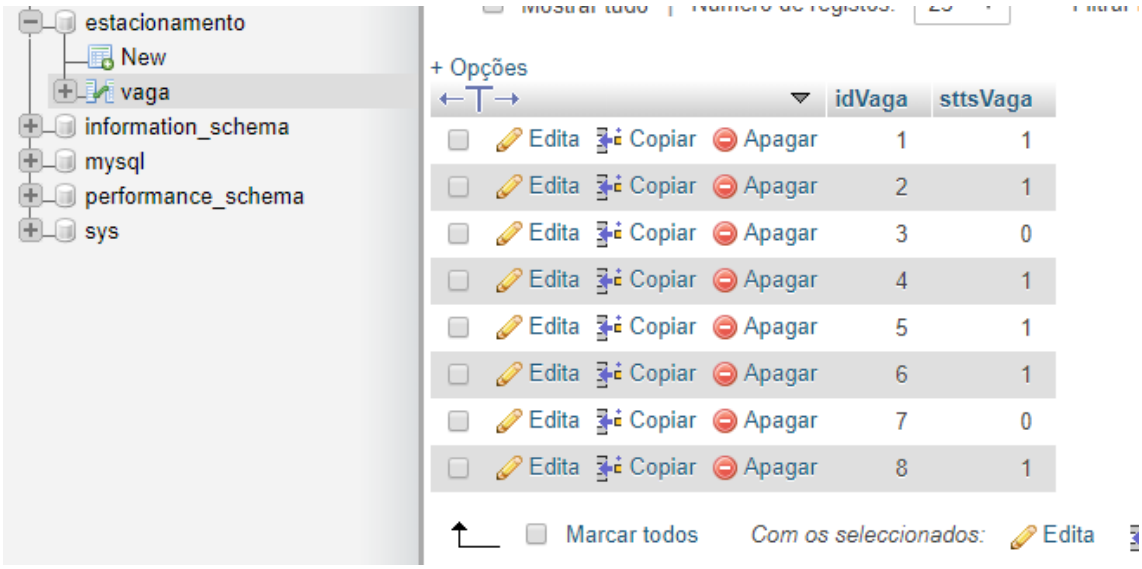
Para testar a segunda página, levou-se em conta os dados armazenados neste momento no banco de dados (Figura 133) para comparação e digitou-se o seguinte link para teste: http://192.168.0.11/phpmyadmin/atualizar.php?PCF=0xBB. Onde o trecho “?PCF=0xBB” é a informação enviada para o BD. Espera-se ter como resposta a atualização das vagas na sequência de bits da informação enviada que é 10111011, onde o bit menos significativo é o da direita.

Figura 134 - Enviando dados ao DB.

Fonte: Autor.

Ao consultar o banco verifica-se que as informações estão conforme o esperado, Figura 135, com isso conclui-se que a atualização de dados está funcionando corretamente.

Figura 135 - BD após o envio de dados.



Fonte: Autor.

Com os testes realizados verifica-se que o banco de dados desenvolvidos está funcional e preparado para integração com o sistema.

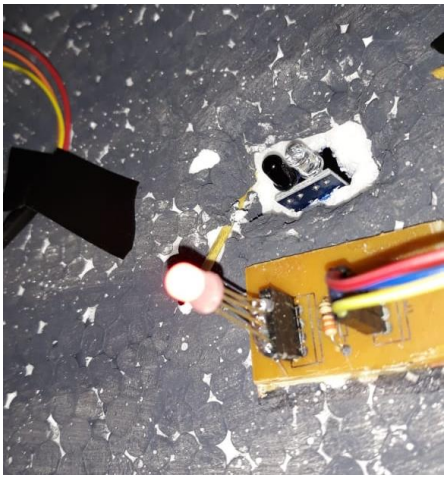
5.1.4. APLICATIVO ANDROID

Para o aplicativo *android*, foi gerado o APK através do software ANDROID STUDIO, importou-se o instalador do aplicativo para o smartphone e o instalou. Antes de iniciar os testes com o aplicativo, verificou-se, primeiramente, se o mesmo estava conectado a rede local. Após a confirmação, abriu-se o aplicativo e clicou-se no botão atualizar status e imediatamente foi atualizado os estados das vagas apresentada no aplicativo. E assim validou-se o aplicativo.

5.2. INTEGRAÇÃO DOS COMPONENTES DO SISTEMA

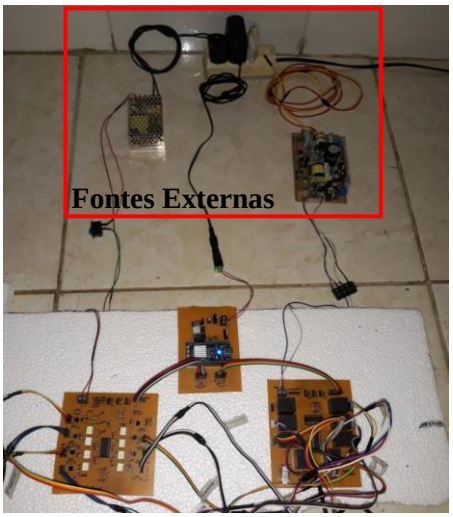
Inicialmente foi preparado o ambiente de teste, posicionando os sensores acima das vagas e instalando as placas dos leds próximos aos sensores, Figura 137. Conectou-se cabos para realizar a ligação dos sensores e dos leds com as placas de leitura e escrita, respectivamente. As placas de leitura e escrita foram conectadas a placa de controle por meio de jumpers fêmea-fêmea. Foram conectadas as alimentações externas em cada uma das placas.

Figura 137 - Leds e sensores sobre as vagas.



Fonte: Autor.

Figura 136 - Placas e suas respectivas fontes.



Fonte: Autor.

O firmware do sistema teve seu código adaptado para realizar o log do sistema, armazenando as informações sobre: quantidade de iterações do sistema (detecção de variação no estado das vagas), número de tentativas de atualização do banco de dados para cada iteração, quantidade de tempo e falhas de atualização. Essas informações geradas em um arquivo de texto de extensão .txt e serão armazenadas dentro do microcontrolador.

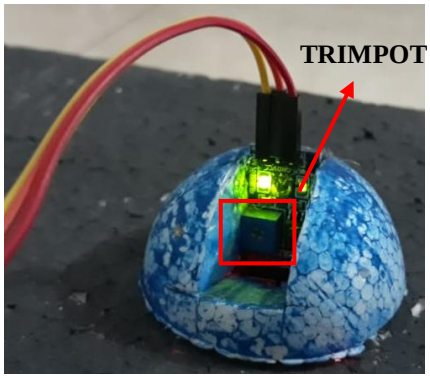
Para os testes usou-se carros em miniatura para alocar as vagas do estacionamento, conforme a Figura 138.

Figura 138 - Carros ocupando vagas.

Fonte: Autor.

Antes de iniciar os testes, resetou-se as configurações do sistema. Para isso reinicializou-se os serviços do WAMP pelo ícone contido na barra de tarefas, Figura 140. Foram calibrados a distância de detecção dos sensores utilizando os seus TRIMPOTS, Figura 139. Retiraram-se todos os obstáculos dos sensores e reinicializou-se o kit de desenvolvimento. O aplicativo foi atualizado e mostrou, inicialmente, todas as vagas livres.

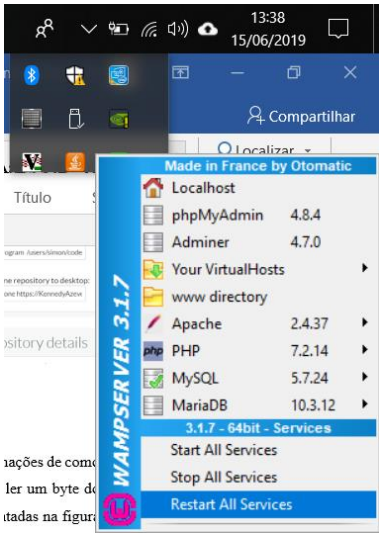
Figura 139 - Trimpot do sensor.



Fonte: Autor.

Após ter resetado o kit de desenvolvimento, aguardou-se cerca de 25 segundos para que o microcontrolador realiza-se todas as configurações necessárias do módulo wifi e para que o sistema iniciasse corretamente e assim iniciar os testes.

Figura 140 - Reiniciando todos os Serviços do WAMP.



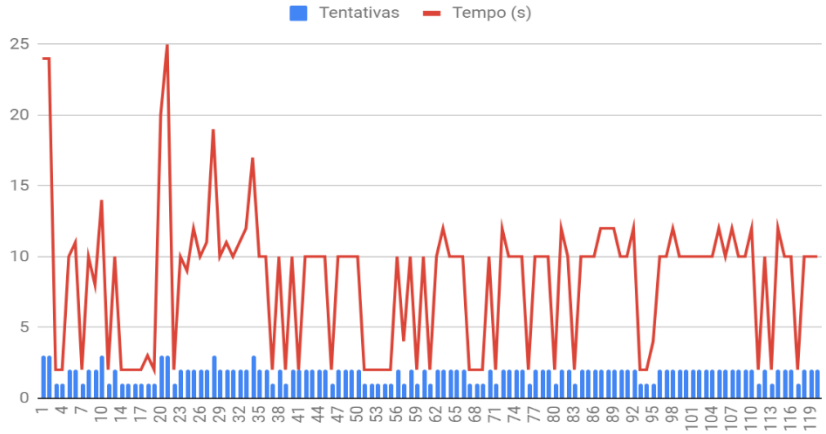
Fonte: Autor.

5.3. RESULTADOS OBTIDOS

A primeira situação testada, foram a movimentação aleatória dos carrinhos. A cada ação de retirar e/ou colocar o carinho em um período de tempo, é chamado de interação. E a cada iteração realizada, foi feito a consulta no aplicativo para verificar o estado das vagas apresentadas por ele. A aplicação android, atualizava a tela quando o botão “ATUALIZAR STATUS” era pressionado e quando havia alguma mudança no banco de dados. Neste teste foram realizadas 119 iterações.

As informações do log do sistema foram resgatadas do microcontrolador do kit de desenvolvimento e representadas em um gráfico o qual é mostrado na Figura 141. Quanto as informações brutas do arquivo que foram gerados pelo microcontrolador serão anexados no Apêndice deste trabalho.

Figura 141 - Gráfico dos Resultados do Sistema.



Fonte: Autor.

Analisando o gráfico, verifica-se que o tempo de atualização do sistema era realizado, em média, de 2 a 10 segundos. Houve casos, conforme as iterações 1, 2 e 21, que o tempo ultrapassou os 20 segundos e foram justamente as iterações que não realizaram a atualização do banco de dados. O gráfico, também, representa a quantidade de tentativas realizadas em cada iteração. A maioria das iterações ocorriam na segunda tentativa. As tabelas a seguir, apresentam estatísticas do gráfico tal.

Tabela 2 - Estatísticas do Sistema.

Tentativas	Ocorrências	Porcentagem
1	35	5,88%
2	77	64,71%
3	7	29,41%

Fonte: Elaborado pelo Autor, 2019.

Tabela 3 - Característica do Sistema.

Tempo Médio de Atualização (s):	10
Quantidade de Falhas:	3

Fonte: Elaborado pelo Autor, 2019.

De acordo com as informações coletadas durante os testes do sistema conclui-se que a hipótese descrita nesse projeto foi alcançada. Pois, com o funcionamento do sistema, é possível disponibilizar, para os usuários, informações sobre as vagas do estacionamento monitorado através de um aplicativo *android* desenvolvido para este fim, possibilitando os usuários visualizar as vagas livres e assim conduzir seu veículo para estacionar sem gastar tempo e gasolina a procura de vagas.

5.4. TABELA DE CUSTOS

Foi realizado um levantamento de custos dos elementos que compõem o sistema de monitoramento inteligente do estacionamento, sem levar em consideração, custos com mão de obra. Nas tabelas constam as quantidades de cada componente que foi utilizado no protótipo. O valor total estimado para a realização desse projeto é de R$ 528,58. As tabelas a seguir, mostram os custos detalhados do sistema.

Tabela 4 - Custos dos Componentes da Placa de Controle.

COMPONENTES	QUANTIDADE	VALOR	TOTAL
ESP8266-01	1	21,00	21,00
Dissipadores de Calor	1 pct 3 pçs	5,90	5,90
7805 Regulador de Tensão	1	2,40	2,40
LD1 733 Regulador de Tensão	1	2,40	2,40
Capacitores Cerâmicos 10nF	2	0,05	0,10
Capacitores Eletrolíticos	2	0,20	0,40
Kit de Desenvolvimento LPCN768	1	248,00	248,00

Fonte: Elaborado pelo Autor, 2019.

Tabela 5 - Custos dos Componentes da Placa de Escrita.

COMPONENTES	QUANTIDADE	VALOR	TOTAL
PCF 8574	1	9,90	9,90
ULN 2803A	1	17,46	17,46
Relés	8	3,00	24,00
Jumpers sem Haste	3	0,30	0,90
Borne de 2 Vias	2	0,80	1,60
Resistores	3	0,25	0,75
Barras de Pino Macho	3	1,50	4,50

Fonte: Elaborado pelo Autor, 2019.

Tabela 6 - Custos dos Componentes da Placa de Leitura.

COMPONENTES	QUANTIDADE	VALOR	TOTAL
Transistores BC547	2 pct 5 pçs	10,90	21,80
Optoacoplador 4N35	8	1,49	11,92
PCF 8574	1	9,90	9,90
Bornes de 2 vias	1	0,80	0,80
Jumpers sem haste	3	0,30	0,90
Resistores	35	0,25	6,00
Barras de Pino Macho	3	1,50	4,50

Fonte: Elaborado pelo Autor, 2019.

Tabela 7 - Custos do Material de Confecção da PCI.

COMPONENTES	QUANTIDADE	VALOR	TOTAL
Placas de fenolite 15x15	3	13,00	39,00
Percloreto de ferro em pó	1	27,50	27,50
Rolo de solda	1	38,95	38,95
Jumpers	2m	28,00	28,00

Fonte: Elaborado pelo Autor, 2019.

5.5. TRABALHOS FUTUROS

Analisando o grande potencial do projeto, pode-se sugerir grandes melhorias para este sistema, como: redução de custos, robustez, simplificação de hardware, melhoria na interface do aplicativo e na disponibilização de informações.

Para o aplicativo sugere-se uma interface mais amigável e que ofereça suporte a vários sensores que o estacionamento possa oferecer. O aplicativo desenvolvido para este sistema suportava apenas sete vagas.

A distribuição das informações era limitada apenas aos dispositivos conectados a rede local. Para o retrabalho do sistema, sugere-se disponibilizar os dados na internet para que as informações não sejam acessadas de qualquer lugar do mundo.

Observando o funcionamento do projeto de hardware, verifica-se uma redundância no uso de microcontroladores. Pois ambos microcontroladores possuem as mesmas

funcionalidades, dentre elas: o barramento I²C, que é um dos recursos fundamentais e necessários para o funcionamento do sistema. Porém, o microcontrolador do ESP8266-01 se sobressai por disponibilizar conexão com a rede sem fio. Então sugere-se a retirada do kit de desenvolvimento, que é um dos componentes mais caros, deixando somente o microcontrolador da Espressif, o ESP8266.

A união das placas de escrita e leitura economizará materiais e componentes. O que facilitará, também, a confecção em larga escala destas placas. Como o projeto trata-se de uma simulação do sistema em um ambiente miniaturizado, utilizou-se leds RGB e sensores de distância infravermelho capaz de detectar obstáculos em até 30 centímetros, no máximo. Porém, para uma aplicação real, esses dispositivos não serão suficientes para serem implementados num ambiente real, necessitando-se de dispositivos mais robustos que podem ser encontrados no mercado.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ALHAK, A., & HIKMAT, S. (2011). *Estacionamento inteligente.* Universidade de Brasília, Brasília.

Araújo, T. (s.d.). Sensor Ultrassônico com Arduino.

BEGHINI, L. B. (2013). *Automação Residencial de baixo custo por meio de dispositivos móveis com sistema operacional Android.* Universidade de São Paulo, São Paulo.

BRAGA, N. C. (2013). Alarmes: Conceito e Aplicação. São Paulo. Fonte: Instituto Newton C. Braga: https://www.newtoncbraga.com.br/arquivos/ncb_alarmes_previa.pdf

Cardoso, V. H., Renato, R. F., de Almeida, M. B., & Cunha, M. J. (12-16 de Outubro de 2015). *Proposta de um Medidor de Consumo Utilizando Tecnologia de Internet das Coisas.* Uberlândia - Minas Gerais Brasil.

CENTENARO, J. (2014). *DESENVOLVIMENTO DE UM SOFTWARE WEB PARA GERENCIAMENTO DE REQUISITOS DE SOFTWARE.* MONOGRAFIA DE ESPECIALIZAÇÃO, Universidade Tecnológica Federal do Paraná , FRANCISCO BELTRÃO .

Eduardo. (2012). *Uma introdução ao JSON.* Fonte: DEVMEDIA: https://www.devmedia.com.br/json-tutorial/25275

EMPRESÔMETRO, I. (20 de 03 de 2018). *REAL FROTA CIRCULANTE NO BRASIL É DE 65,8 MILHÕES DE VEÍCULOS, INDICA ESTUDO*. Acesso em 20 de 03 de 2019, disponível em Instituto Brasileiro de Planejamento e Tributação: https://ibpt.com.br/noticia/2640/REAL-FROTA-CIRCULANTE-NO-BRASIL-E-DE-65-8-MILHOES-DE-VEICULOS-INDICA-ESTUDO

ESTACIONAMENTO ESCALA 1:43 - (MÓDULOS)- DIORAMA MAQUETE. (s.d.). Fonte: EGOBOX: https://www.egobox.com.br/produto/estacionamento-escala-1-43-modulos-diorama-maquete/687648

Fabrício, M. A. (2018). *Monitoramento de Equipamentos Elétricos Industriais Utilizando IoT.* Pontifícia Universidade Católica de Campinas, Campinas.

Fernandes, C. (11 de 07 de 2018). *Sistemas Operacionais Embarcados.* Acesso em 24 de 02 de 2019, disponível em Medium: https://medium.com/@carol.fernandes_/sistemas-operacionais-embarcados-o-sistema-embarcado-também-chamado-de-sistema-embutido-é-um-9e4695923ff3

Ferreira, R. (23 de 10 de 2017). *REST: Princípios e boas práticas.* Acesso em 05 de 03 de 2019, disponível em Caelum: https://blog.caelum.com.br/rest-principios-e-boas-praticas/

Filho, J. d. (2009). *CI Reguladores de Tensão.* Universidade Estadual Paulista, Guaratinguetá.

Freitas, L. F. (2016). *Desenvolvimento de um Sistema de Gestão de Informação para o Website do Laboratório de Processamento de Sinais.* São Carlos.

Matias, J. (24 de 04 de 2013). *O Que São Transístores?* Fonte: josematias: http://www.josematias.pt/eletr/o-que-sao-transistores/

Nunes, K. (2014). *Aplicativo para Aompanhamento de Ocorrências do Paciente Fora do Estabelecimento de Saúde.* Curitiba.

Prado, S. (05 de 01 de 2018). *I3C, o futuro substituto dos barramentos I2C e SPI?* Acesso em 06 de 02 de 2019, disponível em Sérgio Prado: https://sergioprado.org/i3c-o-futuro-substituto-dos-barramentos-i2c-e-spi/

Reichert, R. G., Miranda, R. S., Callai , J. M., Baggio, J. E., Puerari, A. F., Gonçalves, C. d., . . . Linhares, J. C. (3 a 6 de outubro de 2011). Estacionamentos Inteligentes como Solução ao Congestionamento de Grandes Centros Urbanos. *XXXIX Congresso Brasileiro de Educação em Engenharia*.

Rodovalho, L. B. (2012). *Modelagem e Simulação de Processos Biomatemáticos usando Redes de Petri Predicado Transição Diferenciais: Estudo de Caso sobre o Vírus HIV e o Ciclo de Vida do Aedes aegypti.* UNIVERSIDADE FEDERAL DE GOIÁS, Catalão.

Scriptore, D. B., & Júnior, J. d. (2014). *Banco de Dados Distribuído para Consulta de Temperatura e Umidade Utilizando Arduino e Android.* Universidade Paranaense, Paranavaí - PR - Brasil.

Silveira, S. M., & Gonçalves, T. S. (2016). *Automação Residencial utilizando Arduíno e SO Android.* Rio de Janeiro. Fonte: http://bsi.uniriotec.br/tcc/textos/201707ThadeuGoncalvesSandroSilveira.pdf

Sousa, A. F. (2008). *Projeto da Base de Dados de um Sistema Web para Controle de Enasios e Calibração de Laboratórios em Conformidade com a Norma ISO/IEC 17025.* São Carlos.

Tadewald, R. S. (2014). *Desenvolvimento de um aplicativo em Android para o cálculo da perda de carga.* Universidade Federal do Rio Grande do Sul, Departamento de Engenharia Química, Porto Alegre.

Brito, R. C. *Desenvolvimento Android utilizando a IDE Android Studio*. Disponível em:< http://www.devmedia.com.br/desenvolvimento-android-utilizando-a-ide-android-studio/33872>. Data de acesso: 06 de abril de 2016.

Câmara, Rômulo. *Protocolo I2C*. Disponível em: <http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-2C.pdf>. Acesso em: 19 de abril de 2016.

Colunista Portal - Educação. *Protocolos para internet – TCP/IP*. Disponível em:<http://www.portaleducacao.com.br/informatica/artigos/42899/protocolos-para-internet-tcp-ip>. Data de acesso: 06 de abril de 2016.

Longaretti, D.; GIRARDI, A. G.; ENGROFF, A. M. *Implementação de uma interface i2c em fpga*. In: Anais do Salão Internacional de Ensino, Pesquisa e Extensão, v. 7, n. 2, 2016. Anais... Rio Grande do Sul, 2015, Universidade Federal do Pampa, p.1.

Marshall, Brain. *Como funcionam os microcontroladores*. 01 de abril de 2000. Disponível em: <http://informatica.hsw.com.br/microcontroladores.htm>. Acesso em: 16 de maio de 2016.

Mazzaroppi, Marcelo. *Sensores de movimento e presença*. Rio de Janeiro: Universidade do Rio de Janeiro, 2007.

Moreira, Ardilhes. *Frota de veículos cresce 119% em dez anos no Brasil, aponta Denatran*. Disponível em: <http://g1.globo.com/carros/noticia/2011/02/frota-de-veiculos-cresce-119-em-dez-anos-no-brasil-aponta-denatran.html>. Acesso em: 17 de maio de 2016.

Portugal, Arduino. *Conceito de Shields*. Disponível em: <http://www.arduinoportugal.pt/temas/conceito-de-shields>. Acesso em: 05 de maio de 2016.

Quinderé, Patrick R. F., 2009. *Casa Inteligente – Um Protótipo de Sistema de Automação Residencial de Baixo Custo*. Disponível em http://www.ffb.edu.br/sites/default/files/tcc-20082-patrick-romero-frota-quindere.pdf. Acesso em: 17 maio de 2016.

Semiconductors, Philips. *PCF8574 Remote 8-bit I/O Expander for I2C-bus*. Philips Semiconductors, 2002. Disponível em: <http://www.still-creative.de/wp-content/uploads/2013/09/PCF8574_PCF8574A.pdf>. Acesso em: 05 de maio de 2016.

Souza, Fábio. *Arduino – Primeiros Passos*. Disponível em: <http://www.embarcados.com.br/arduino-primeiros-passos/>. Acesso em: 15 de abril de 2016.

Teleco. *Estatística de Celulares no Brasil*. Disponível em: <http://www.teleco.com.br/ncel.asp>. Acesso em: 05 de maio de 2016.

Marcelo, Antônio. *Apache: Configurando o servidor web para Linux*. Rio de Janeiro: Brasport, 2005

Ford, Andrew. Apache 2 pocket reference. Gravenstein Highway North: O`Reilly, 2008.

Hultquist, Steve *"FAQ about Client/Server"* http://non.com/news.answers/client-server-faq.html, 1997

Vargas, Thânia C. S. (2007) *A História de UML e seus Diagramas*. Disponível em: <https://projetos.inf.ufsc.br/arquivos_projetos/projeto_721/artigo.tcc.pdf>. Acessado em: 01/06/2019.

APÊNDICE A – Código Principal

```
#include "mbed.h"
#include "rtos.h"
#include "main.h"
#include "PCF8574.h"
#include "stdint.h"
#include "string.h"
#include "commands_esp01.h"

LocalFileSystem local("local");          // Create the local filesystem under the name "local"

#define T_BLINK 2

extern Serial pc;
extern Serial esp;

PCF8574 ExpWrite(p9,p10,0x40);      // Expansor de Portas: Leitura
PCF8574 ExpRead(p28,p27,0x41);      // Expansor de Portas: Escrita
DigitalIn btnInt(p21);          // Pino de Interrupção (Detecta Mudança na Leitura do PCF8574)

DigitalOut led1(LED1);
DigitalOut led2(LED2);

uint8_t data;
Thread thread;

/**< Definindo as variáveis Globais a serem utilizadas */
```

```
EstadoDaMain state = HARDWARE_RESET;


void readSensors()
{
   while(true)
   {
      data = ExpRead.read();
      ExpWrite.write(data);


      if(state == IDLE && btnInt==0)
      {
         led2 = 1;
         pc.printf("\r\nEnviando.... \r\n");
         state = ESTABLISHING_CONNECTION;
      }
   }
}


int main()
{
   Timer tempo;
   int t=0;
   FILE * fp = fopen("/local/file.txt", "w");  // Open "out.txt" on the local file system for writing


   fclose(fp);
   thread.start(readSensors);
   InitBaudrateSerial(); // Definindo o baudrate das portas seriais
```

```
tempo.start();


while(1)
{
   switch(state)
   {
      case HARDWARE_RESET:
         pc.printf("HARDWARE_RESET\r");
         ESPHardwareReset();
         state = SET_BAUDRATE;
      break;
      case SET_BAUDRATE:
         pc.printf("SET_BAUDRATE\r");
         ESPsetbaudrate();
         state = CONFIG;
      break;
      case CONFIG:
         pc.printf("CONFIG\r");
         ESPconfig();
         state = ESTABLISHING_CONNECTION;
      break;
      case ESTABLISHING_CONNECTION:
         led1 = 0;
         state = IDLE;
         sendByteToDB();
         led2 = 0;
```

```
            t=1;
        break;
        case CLOSING_CONNECTION:
            pc.printf("CLOSING_CONNECTION\r");
            close_connection();
            state = IDLE;
        break;
        case IDLE:
        default:
            if(t==1)
            {
                tempo.stop();
                pc.printf("\r\ntempo = %f\r\n",tempo.read());
                t=0;
            }
        break;
      }
      led1 = !led1;
      Thread::wait(1000);
   }
}
```

APÊNDICE B – Arquivo .cpp da biblioteca criada para o ESP8266-01

```
#include "commands_esp01.h"
#include "mbed.h"
extern FILE * fp;
uint32_t cnvT;
Timer tempo;
uint16_t cntErr=0, iteracao=0;
                    extern uint8_t data;
                    Serial esp(p13, p14); // tx, rx      // Porta Serial de Comunicação entre
ESP01 e ARM
                    Serial pc(USBTX,USBRX);//tx, rx      // Porta serial para debug

                    DigitalOut reset(p12);              // Pino de Controle do Reset do ESP01

                    int  timeout;
                    char buf[1024];
                    char snd[255];

                    char ssid[32] = "Sinal Fraco";      // enter WiFi router ssid inside the quotes
                    char pwd [32] = "kmfa.eng"; // enter WiFi router password inside the
quotes
void InitBaudrateSerial(void)
{
                    pc.baud(9600);
                    esp.baud(115200);
}
void ESPHardwareReset(void) // Hardware reset for 8266
{
```

```
    pc.printf("\f\n\r-------------ESP8266 Hardware Reset-------------\n\r");
    reset=0;
    wait(1);
    reset=1;
    timeout=2;
    getreply();
}
void ESPconfig(void)
{
    wait(1);
    strcpy(snd,"AT\r\n");
    SendCMD();
    wait(1);
    strcpy(snd,"AT\r\n");
    SendCMD();
    wait(1);
    strcpy(snd,"AT\r\n");
    SendCMD();
    timeout=1;
    getreply();
    getreply();
    pc.printf(buf);
    wait(1);

    //pc.printf("\f---------- Starting ESP Config ----------\r\n\n");
    //pc.printf("---------- Reset & get Firmware ----------\r\n");
```

```
strcpy(snd,"AT+RST\r\n");
SendCMD();
timeout=5;
getreply();
pc.printf(buf);
wait(1);

strcpy(snd, "AT+CWMODE=3\r\n");
SendCMD();
timeout=4;
getreply();
pc.printf(buf);
wait(1);

strcpy(snd, "AT+CIPMUX=0\r\n");
SendCMD();
timeout=4;
getreply();
pc.printf(buf);
wait(1);

strcpy(snd, "AT+CWJAP=\"");
strcat(snd, ssid);
strcat(snd, "\",\"");
strcat(snd, pwd);
strcat(snd, "\"\r\n");
SendCMD();
```

```
    timeout=10;
    getreply();
    pc.printf(buf);
    wait(1);
}
void SendCMD(void)
{
    esp.printf("%s", snd);
}
STTS_CMD_ESP getreply(void)
{
                        char respOK[]="OK";
                        char respFAIL[]="FAIL";
                        char respERROR[]="ERROR";

                        STTS_CMD_ESP stts = CMD_FAIL;

                        bool flag = false;

                        int  count,ended;
                        Timer t;

    memset(buf, '\0', sizeof(buf));
    t.start();
    ended=0;
    count=0;
    while(!ended)
```

```
{
   if(esp.readable())
   {
      buf[count] = esp.getc();


      if(count>0)
      {
                  if(memcmp(&buf[count-1], respOK, sizeof(respOK))==0)
                  {
                     pc.printf("\r\n\r\nOK[1]\r\n\r\n");
                     stts = CMD_OK;
                     flag=true;
                  }
      }
      if(count>3)
      {
                  if(memcmp(&buf[count-4], respFAIL, sizeof(respFAIL))==0)
                  {
                     pc.printf("\r\n\r\nFAIL[1]\r\n\r\n");
                     stts = CMD_FAIL;
                     flag=true;
                  }
                  else if(memcmp(&buf[count-1], respERROR, sizeof(respERROR))==0)
                  {
                     pc.printf("\r\n\r\nERROR[1]\r\n\r\n");
                     stts = CMD_ERROR;
                     flag=true;
```

```
                        }
            }
            count++;
        }
        if(t.read() > timeout)
        {
            ended = 1;
            t.stop();
            t.reset();
            pc.printf("\r\n\r\nTIMEOUT[1]\r\n\r\n");
            stts = CMD_TIMEOUT;
            flag = true;
        }
        if(flag == true)
                        break;
    }
    return stts;
}
// Sets new ESP8266 baurate, change the esp.baud(xxxxx) to match your new setting once this has been executed
void ESPsetbaudrate(void)
{
    strcpy(snd, "AT+CIOBAUD=115200\r\n");   // change the numeric value to the required baudrate
    SendCMD();
    getreply();
    pc.printf(buf);
    wait(1);
```

```
};

void sendByteToDB(void)
{
            STTS_CMD_ESP stts = CMD_FAIL;
             int tam, tries=0;
              char aux[100] = "";
               char postRequest[255] = "";

                        FILE * fp = fopen("/local/file.txt", "a");  // Open "out.txt" on the local file system for writing
   tempo.start();
   while(stts!=CMD_OK && tries<3)
   {
                        strcpy(snd, "AT+CIPSTART=\"TCP\",\"192.168.0.11\",80\r\n");
                        SendCMD();
                        timeout=2;
                        stts = getreply();
                        pc.printf(buf);
                        wait(1);

                        sprintf(postRequest, "GET /phpmyadmin/atualizar.php?PCF=0x%02x HTTP/1.1\r\nHost: 192.168.0.11:80\r\n\r\n\r\n",(data ^ 0xFF));
                        tam = strlen(postRequest);

                        pc.printf(postRequest);

                        sprintf(aux,"AT+CIPSEND=%d\r\n",tam);
```

```
                    strcpy(snd, aux);
                    SendCMD();
                    timeout=2;
                    stts=getreply();
                    pc.printf(buf);
                    wait(1);

                    strcpy(snd, postRequest);
                    SendCMD();
                    timeout=2;
                    stts=getreply();
                    pc.printf(buf);
                    tries++;
                    }
   iteracao++; cntErr = tries; cnvT = tempo.read();
                    tempo.stop(); tempo.reset();

                    if(stts != CMD_OK && tries==3)
                       fprintf(fp, "iteracao = %d\tTentativa = %d\tcnvT%d\tNÃO
ATUALIZOU\r\n", iteracao,cntErr,cnvT);
                    else
                       fprintf(fp, "iteracao = %d\tTentativa = %d\tcnvT%d\r\n",
iteracao,cntErr,cnvT);
                    fclose(fp);

   wait(1);
}
```

```
void close_connection(void)
{
   strcpy(snd, "AT+CIPCLOSE\r\n");
   SendCMD();
   timeout=10;
   getreply();
   pc.printf(buf);
   wait(3);
}
```

APÊNDICE C – Arquivo .h da biblioteca criada para o ESP8266-01.

```
#ifndef _COMMANDS_ESP01_H
#define _COMMANDS_ESP01_H

#include "string.h"
#include "stdint.h"

   typedef enum
   {
      CMD_FAIL=0,
      CMD_ERROR,
      CMD_TIMEOUT,
      CMD_OK
   } STTS_CMD_ESP;

   void ESPHardwareReset(void);
   void SendCMD(void);
   STTS_CMD_ESP getreply(void);
   void ESPconfig(void);
   void ESPsetbaudrate(void);
   void InitBaudrateSerial(void);
   void sendByteToDB(void);
   void close_connection(void);

#endif
```

APÊNDICE D – Código em PHP do atualizar.php

```
<?php

header("Content-type: text/html; charset=utf-8");

$servername = "localhost";
$username = "root";
$password = "kmfa.eng";
$dbname = "estacionamento";

$STATUS_PCF = $_GET['PCF'];

$valDec = hexdec( $STATUS_PCF );

echo "Valor Recebido = ".$valDec."<br />";

$conn = new mysqli($servername,$username,$password,$dbname);

for($cnt=0, $i=1; $cnt <=7 ; $cnt++, $i++)
{
        echo "<br />";

        if ($conn->connect_error){
                die("Não foi possível estabelecer conexão com o BD: " . $conn-
>connect_error);
        }

        $bit = ($valDec & ( 1 << $cnt)) >> $cnt;

        echo "Vaga".$i;
```

```
        if($bit == 1)
                echo " Ocupada <br />";
        else
                echo " Livre <br />";

        $sql = "\n UPDATE vaga SET sttsVaga=$bit WHERE idVaga=$i";

        if (!$conn->query($sql)) {
        //Gravar log de erros
        die("\n \n Erro na gravacao dos dados no BD da posicao");
        }
}
        $conn->close();
?>
```

APÊNDICE E – Código em PHP do format_json.php

```
<?php

header('Access-Control-Allow-Origin: *');

header('Content-type:application/json');


$conn = mysqli_connect('localhost','root','kmfa.eng') or die (mysqli_error()); // É realizado a conexão com o banco de dados


$db = mysqli_select_db($conn,'Estacionamento'); // É selecionado o Bando de Dados


$select = mysqli_query($conn,'SELECT * FROM `vaga`'); // É utilizado a linguagem de requisição estruturada para buscar todos os elementos do banco chamado "vagas"


$rows = array(); // Os elementos serão retornaos em Array


while($row=mysqli_fetch_array($select))

{

                $rows[] = $row;

}

echo json_encode($rows); // Converter os dados para o formato json

?>
```