

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Filipe Cadmo Mariano de Freitas

**Sistema automotivo de autenticação para
liberação de partida do motor de combustão e
rastreamento em tempo real da localização
geográfica**

João Pessoa

2020

Filipe Cadmo Mariano de Freitas

**Sistema automotivo de autenticação para liberação de
partida do motor de combustão e rastreamento em
tempo real da localização geográfica**

Trabalho de Conclusão de Curso apresentado
à Universidade Federal da Paraíba como exi-
gência para a obtenção do título de Bacharel
em Engenharia Elétrica.

Universidade Federal da Paraíba

Centro de Energias Alternativas e Renováveis

Curso de Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Euler Cássio Tavares de Macedo

João Pessoa

2020

© Filipe Cadmo Mariano de Freitas

Catálogo na publicação
Seção de Catalogação e Classificação

F866s Freitas, Filipe Cadmo Mariano de.

Sistema automotivo de autenticação para liberação de partida do motor de combustão e rastreamento em tempo real da localização geográfica / Filipe Cadmo Mariano de Freitas. - João Pessoa, 2020.

62 f. : il.

Orientação: Euler Cássio Tavares de Macedo.
TCC (Especialização) - UFPB/CEAR.

1. Internet das coisas. 2. Microcontrolador. 3. Servidor Web. I. Tavares de Macedo, Euler Cássio. II. Título.

UFPB/BC

Filipe Cadmo Mariano de Freitas

Sistema automotivo de autenticação para liberação de partida do motor de combustão e rastreamento em tempo real da localização geográfica

Trabalho de Conclusão de Curso apresentado à Universidade Federal da Paraíba como exigência para a obtenção do título de Bacharel em Engenharia Elétrica.

Trabalho aprovado. João Pessoa, 01 de abril de 2020:

**Prof. Dr. Euler Cássio Tavares de
Macedo**
Orientador

**Prof. Dr. Jose Mauricio Ramos de
Souza Neto**
Convidado 1

Prof. Dr. Lucas Vinícius Hartmann
Convidado 2

João Pessoa
2020

Agradecimentos

Agradeço a minha família pelo apoio prestado durante toda a graduação. Vocês foram a minha fonte de energia para continuar caminhando.

Agradeço imensamente a todos os professores da UFPB que possibilitaram, passo a passo, a minha formação e fomentaram o meu interesse pela engenharia, capacitando-me para os desafios futuros. Agradeço em especial aos professores Euler e Juan pela confiança depositada durante os projetos desenvolvidos em conjunto.

Agradeço aos colegas que conheci durante a graduação pelas ideias compartilhadas. A jornada foi muito mais agradável com vocês por perto.

Resumo

Com a modernização da sociedade e aglomeração da população em grandes centros urbanos, houve um aumento do número de veículos circulantes no Brasil. Aliado a isso, as empresas brasileiras expandiram sua frotas ao longo dos anos e ao permitir que os funcionários utilizem seus veículos, a empresa é responsável por administrar a responsabilidade de infrações de trânsito.

Baseado na ideia de internet das coisas, propõe-se neste trabalho o desenvolvimento de um dispositivo eletrônico que permita o registro de utilização veicular. Isso será feito com a identificação do motorista por meio da aproximação de um cartão de identificação, tal como um crachá. Os dados de utilização serão salvos em um servidor remoto e podem ser acessados por um usuário administrador.

Um sistema de prova de conceito foi desenvolvido e atendeu as especificações por meio da utilização de um microcontrolador ESP8266. Um servidor web remoto foi desenvolvido com a utilização da linguagem de programação Python. O microcontrolador, que possui comunicação com a internet, realiza requisições do tipo HTTP ao servidor para confirmar a identidade do motorista e enviar dados capturados.

Palavras-chave: Internet das coisas. Microcontrolador. Servidor Web.

Abstract

As society develops and the population migrates towards urban areas, there has been a rise in the number of utilized automobiles in Brasil. On top of that, brazilian companies expanded their vehicle fleet troughout the years and by allowing their employees to use these vechiles, the company is responsible to manage the resposability of traffic violations.

Based on the idea of internet of things, this paper proposes the development of an electronic device that allows the logging of vehicle utilization. This shall be done through the identification of drivers using a card identification by approximation technology. The usage data shall be saved in a remote server and may be accessed by a administrator user.

A proof of concept system was developed and met the specifications using a microcontroller of type ESP8266. A remote web server was developed with the utilization of Python progamming language. The microcontroller, which can communicate with the internet, make requests of type HTTP to the server to confirm the identity of the driver and send captured data.

Keywords: Internet of things. Microcontroller. Web server.

Lista de ilustrações

| | |
|--|----|
| Figura 1 – Frota circulante no Brasil. | 12 |
| Figura 2 – Participação de vendas diretas para automóveis e veículos comerciais leves ao decorrer dos anos. | 13 |
| Figura 3 – Exemplo de interação entre o identificador e leitor RFID. | 16 |
| Figura 4 – Diagrama de triangulação do Sistema GPS. | 18 |
| Figura 5 – Interação entre dispositivos eletrônicos e roteador Wi-Fi. | 20 |
| Figura 6 – Interação entre um navegador web e um servidor. | 21 |
| Figura 7 – Diagrama de casos de uso do sistema de autenticação veicular. | 25 |
| Figura 8 – Diagrama de componentes do sistema leitor de autenticação. | 27 |
| Figura 9 – Ambiente de desenvolvimento <i>Arduino IDE</i> | 28 |
| Figura 10 – Diagrama de pinos da placa de desenvolvimento baseada em Node-MCU. | 28 |
| Figura 11 – Algoritmo utilizado para desenvolver o código do microcontrolador. | 31 |
| Figura 12 – Diagrama de interação entre os componentes físicos, servidor web e usuário administrador. | 33 |
| Figura 13 – Diagrama de sequência das operações feitas pelo usuário administrador. | 34 |
| Figura 14 – Modelo entidade-relacionamento do banco de dados. | 35 |
| Figura 15 – Página inicial do site. | 38 |
| Figura 16 – Mapa estático com o caminho percorrido pelo veículo durante uma sessão. | 38 |
| Figura 17 – Formulário de registro de usuário. | 39 |
| Figura 18 – Formulário de associação de tags. | 40 |
| Figura 19 – Formulário de encerramento de associação de tags. | 40 |
| Figura 20 – Formulário de registro de um novo veículo. | 41 |
| Figura 21 – Formulário de associação de um leitor a um veículo. | 41 |
| Figura 22 – Formulário de encerramento de associação de leitor. | 42 |
| Figura 23 – Protótipo desenvolvido. | 43 |
| Figura 24 – Esquemático de circuito elétrico do sistema leitor. | 43 |
| Figura 25 – Fluxograma da função de atualizar posição geográfica. | 44 |
| Figura 26 – Captura do monitor serial do sistema físico em funcionamento. | 45 |

Lista de tabelas

| | |
|--|----|
| Tabela 1 – Especificações elétricas de operação do Leitor RFID MF. | 29 |
|--|----|

Lista de abreviaturas e siglas

FENABRAVE Federação Nacional da Distribuição de Veículos Automotores

IBGE Instituto Brasileiro de Geografia e Estatística

AIAFA Associação Internacional de Administradores de Frota e de Mobilidade

GPS Global Positioning System

RFID *Radio Frequency Identification*

IoT *Internet of Things*

Sumário

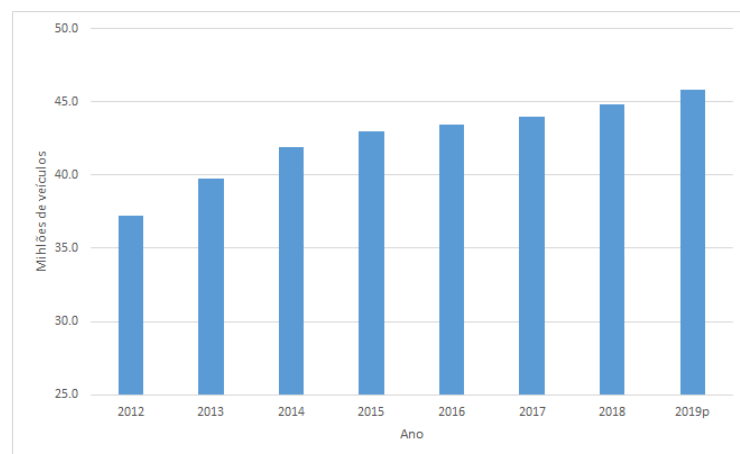
| | | |
|------------|--|-----------|
| 1 | INTRODUÇÃO | 12 |
| 1.1 | Motivação | 13 |
| 1.2 | Objetivos Gerais | 14 |
| 1.3 | Objetivos Específicos | 14 |
| 2 | FUNDAMENTAÇÃO TEÓRICA | 16 |
| 2.1 | Identificação por radiofrequência | 16 |
| 2.1.1 | Tag RFID Ativa | 16 |
| 2.1.2 | Tag RFID Passiva | 17 |
| 2.2 | GPS | 17 |
| 2.3 | Rede de área local sem fio e Wi-Fi | 19 |
| 2.4 | Internet das coisas - IoT | 20 |
| 2.4.1 | Servidor Web | 21 |
| 2.4.2 | Python e Django | 21 |
| 2.4.3 | Banco de dados relacional | 22 |
| 2.5 | Sistemas Comerciais Disponíveis | 23 |
| 3 | MÉTODOS E FERRAMENTAS | 24 |
| 3.1 | Sistema de autenticação veicular | 24 |
| 3.2 | Sistema leitor de autenticação veicular | 26 |
| 3.2.1 | ESP8266 e NodeMCU | 26 |
| 3.2.2 | Tag e Leitor RFID MFRC522 | 29 |
| 3.2.3 | Leitor de cartão SD e relógio RTC | 29 |
| 3.2.4 | Algoritmo do microcontrolador | 30 |
| 3.3 | Sistema virtual: servidor | 32 |
| 4 | RESULTADOS | 37 |
| 4.1 | Servidor Web | 37 |
| 4.2 | Sistema Autenticador | 42 |
| 5 | CONCLUSÃO | 46 |

| | |
|---|----|
| REFERÊNCIAS | 48 |
| APÊNDICES | 50 |
| APÊNDICE A – CÓDIGO DO MICROCONTROLADOR | 51 |

1 Introdução

Em todo o mundo, centenas de milhões de pessoas utilizam diariamente os meios de transporte terrestres para se locomover para o trabalho, escola, bancos e outras localidades. O cidadão brasileiro, por exemplo, gasta diariamente uma hora e vinte minutos (99/IPSOS, 2018) para realizar sua atividade principal do dia. Além disso, um estudo realizado pela Sindipeças revela que a projeção para 2019 do número da frota circulante brasileira é de 45,8 milhões de veículos, enquanto em 2012 a frota era de 37,2 milhões, como pode ser observado na Figura 1. Observa-se, então, que a utilização de automóveis passou a fazer parte de uma parcela crescente do cotidiano moderno da população brasileira.

Figura 1 – Frota circulante no Brasil.

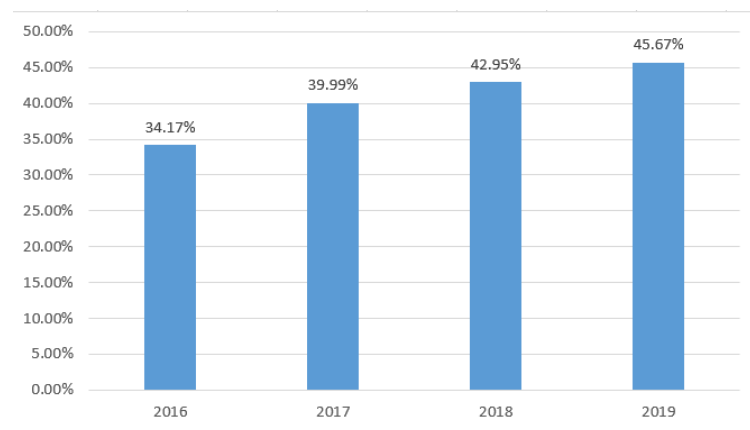


Fonte: Sindipeças.

Segundo o IBGE, o número de pessoas que exercem atividades remuneradas no trânsito, como taxistas, motoristas e trocadores de ônibus, atingiu alta histórica de 3,6 milhões de pessoas em 2018 (CRELIER, 2019). Além disso, em uma pesquisa encomendada pelo Observatório de Veículos de Empresas, 32% das empresas brasileiras diz que tem intenção de aumentar a frota de veículos empresarial, enquanto apenas 12,8% disse que tinha intenção de diminuir (AIAFA, 2018). Somado a isso, as vendas diretas, que são aquelas feitas para corporações e frotistas, ou feitas na concessionária para taxistas, produtores rurais ou pessoas com deficiência, cresceram ao longo dos anos, assim como detalhado no gráfico da Figura 2. A análise desses dados, em conjunto, sugere que existe uma tendência de expansão da utilização de veículos para fins comerciais no Brasil.

Uma vez que uma empresa adquire um veículo, é necessário preocupar-se com as possíveis infrações de trânsito cometidas pelos seus funcionários, uma vez que ela é responsável por indicar o infrator, sob penalidade de sofrer multa por não identificação, conforme previsto no artigo 257, parágrafo oitavo do Código de trânsito Brasileiro. Portanto, cabe a cada empresa fazer a gestão de sua frota veicular, que muitas vezes é um processo manual lento e burocrático.

Figura 2 – Participação de vendas diretas para automóveis e veículos comerciais leves ao decorrer dos anos.



Fonte: FENABRAVE.

1.1 Motivação

Em um ambiente empresarial onde há compartilhamento de automóveis, é necessário que haja histórico de quem utilizou um determinado veículo e quando, pois em um caso de multa de trânsito, a empresa é legalmente responsável por indicar o infrator. Visto que o registro manual dessas informações é burocrática e trabalhosa, além de ser suscetível a erros humanos, este trabalho sugere a implementação de um sistema digital que substitua o registro manual.

Uma grande vantagem desse sistema é a agilidade criada no compartilhamento de carros, pois elimina a burocracia de registro de utilização. Além disso, uma vez que usuário motorista tem ciência que pode ser penalizado caso dirija imprudentemente, há um impacto positivo no comportamento no trânsito de trabalhadores que dirijam carros de frota, possivelmente reduzindo os índices de acidentes e infrações.

1.2 Objetivos Gerais

Dado o contexto de veículos automotivos utilizado em frotas corporativas, este trabalho tem como objetivo geral desenvolver um sistema de rastreabilidade de veículos automotores baseado em tecnologia IoT, que permita além de identificar o motorista, realizar o rastreamento georreferenciado por meio de GPS. Uma autenticação deve ser feita por meio de um cartão de identificação pessoal e o sistema deve consultar uma base de dados *online* para verificar que o usuário está cadastrado e pode utilizar o automóvel. Além disso, o sistema eletrônico deve monitorar a posição geográfica do automóvel e salvá-los remotamente. Todos os dados devem ser salvos em um servidor web para que consultas posteriores sejam possíveis.

1.3 Objetivos Específicos

Para que o objetivo geral seja cumprido, deve ser desenvolvido um sistema eletrônico físico e um servidor web que guarde as informações de configuração e os dados recebidos. O primeiro deve ser capaz de:

- Ler cartões de identificação por radiofrequência;
- Capturar localização geográfica através de um sistema GPS;
- Interceptar a conexão elétrica entre a bateria do carro e os demais sistemas veiculares, fazendo com que essa conexão seja reestabelecida a partir de um relé comandado eletricamente caso o usuário motorista se autentique com sucesso;
- Comunicar-se com um servidor *web* remoto;
- Salvar dados localmente em caso de perda de comunicação por meio de um cartão de memória;

Já o servidor web deve cumprir as seguintes tarefas:

- Permitir que um usuário administrador cadastre usuários veiculares, carros e cartões através de uma interface gráfica para fácil identificação posterior;

- Receber requisições de autenticação e retornar se um usuário está liberado, enquanto guarda o histórico de requisições recebidas em um banco de dados;
- Receber informações de localização dos sistemas físicos e salvar essas aquisições em um banco de dados;
- Exibir o histórico de autenticação e localização para o usuário administrador por meio de uma interface gráfica;

2 Fundamentação Teórica

Neste capítulo serão apresentados os conceitos associados aos equipamentos e técnicas utilizadas no desenvolvimento deste projeto. O escopo deste projeto não tem como objetivo desenvolver ou aprimorar as tecnologias mencionadas, e sim utilizá-las a fim de resolver um problema de engenharia.

2.1 Identificação por radiofrequência

Esta Seção apresentará um resumo técnico da tecnologia de identificação por radiofrequência, também comumente chamada de RFID, do inglês *Radio Frequency Identification*. RFID é uma tecnologia de identificação automática por aproximação que utiliza ondas eletromagnéticas para realizar a troca de dados. Os componentes fundamentais para o funcionamento é o identificador RFID, também chamado de *RFID Tag*, e o dispositivo leitor. A interação é simples e intuitiva, conforme ilustrado na Figura 3

Figura 3 – Exemplo de interação entre o identificador e leitor RFID.



Fonte: Autor.

2.1.1 Tag RFID Ativa

As tags de RFID ativas possuem uma fonte de energia interna, ou seja, uma bateria que alimenta o circuito comunicador. Geralmente são mais caras do que as tags passivas devido a presença da bateria, o que também permite a utilização de frequências maiores, comumente 455 MHz, 2,45 GHz ou 5,8 GHz. A vantagem de utilização desse tipo de

identificador é a possibilidade de comunicação através de uma maior distância, de 20 a 100 metros. A desvantagem é a vida útil limitada (GARFINKEL; HOLTZMAN, 2005).

2.1.2 Tag RFID Passiva

As tags passivas de RFID possuem preço consideravelmente menor em relação as tags ativas, uma vez que não utilizam bateria interna. Devido ao custo inferior, são utilizadas com maior frequência. Quando uma tag entra em contato com a faixa de alcance das ondas eletromagnéticas emitidas pelo leitor, a antena presente na tag carrega um capacitor presente na tag. Quando o capacitor possuir carga suficiente para alimentar o circuito integrado interno, um sinal é modulado com informações presentes em uma memória interna ao identificador. Esse sinal, então, é enviado ao leitor, que processa e envia a informação a um sistema digital, tal como um microcontrolador. Essa comunicação pode ocorrer em diversas faixas de frequência, comumente 128 KHz, 13,6 MHz, 915 MHz e 2,45 GHz. Podem ter alcance de alguns centímetros até cerca de 9 metros (WEINSTEIN, 2005).

As tags passivas podem ser envolvidas por diversos tipos de materiais, sendo plástico o tipo mais comum. Cartões de crédito, crachás e cartões de tarifação de transporte público são alguns exemplos de onde tags passivas são utilizadas. Elas também podem ser fixadas em etiquetas para controle de inventário, por exemplo.

2.2 GPS

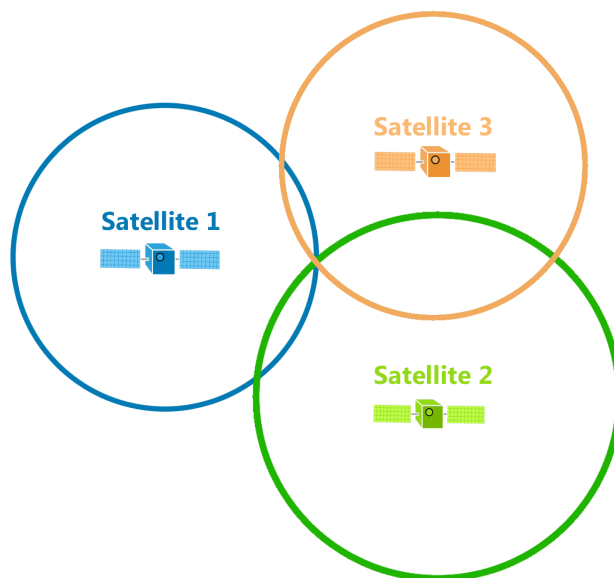
O sistema de posicionamento global GPS (*Global Positioning System*, em inglês) é um sistema que tem como objetivo determinar as coordenadas geográficas de um dispositivo eletrônico. Foi criado pelo Departamento de Defesa dos Estados Unidos e possui, inicialmente, motivação militar. Seu desenvolvimento foi iniciado em 1970, enquanto que foi disponibilizado para uso civil em 1983. Apenas no ano de 1994 foi declarado completamente funcional (WANG, 2012).

O sistema é operado por pelo menos 24 satélites que orbitam a terra em uma altura de 20 mil e 200 quilômetros a uma inclinação de 55 graus da linha do Equador com período orbital de 11 horas e 58 minutos. O processamento dos sinais enviados por esses

satélites permite que as coordenadas geográficas sejam determinadas com uma incerteza de 10 metros (LOMBARDI et al., 2001).

O princípio de funcionamento desse sistema é simples, conforme ilustrado na Figura 4, porém a execução é complexa devido a grande exatidão de medição temporal necessária. Se a distância entre três satélites for conhecida, é possível traçar esferas que determinam a posição geográfica exata por meio das suas interseções. Na realidade, entretanto, o processo é mais complexo, pois não é possível determinar a distância entre os satélites diretamente. Em vez disso, os satélites emitem a informação da data e hora atual com precisão de nanosegundos, uma vez que possuem relógios atômicos.

Figura 4 – Diagrama de triangulação do Sistema GPS.



Fonte: gisgeography.com.

O dispositivo eletrônico receptor também possui um relógio interno, porém do tipo quartz devido ao custo inferior. Uma vez que o sinal de data e hora é recebida pelo receptor, é possível calcular a diferença de tempo entre a hora do sinal transmitido e a do relógio de quartz. Sabendo a diferença de tempo, é possível calcular a distância percorrida pelo sinal, uma vez que esse é transmitido na velocidade da luz. Além disso, como a órbita dos satélites são previsíveis, o módulo receptor possui a informação de localização geográfica de todos os satélites em uma memória, usualmente chamada de almanaque. Finalmente, em posse da posição e distância dos satélites, é possível realizar o cálculo de triangulação e determinar a posição geográfica do dispositivo eletrônico. Um quarto satélite é utilizado para constantemente sincronizar o relógio de quartz dos dispositivos receptores, uma vez

que esse tipo de relógio não possui precisão de nanosegundos (BRAIN; HARRIS, 2006).

2.3 Rede de área local sem fio e Wi-Fi

Baseada no conjunto de padrões IEEE 802.11, a rede de área local sem fio é uma tecnologia que utiliza a técnica de espalhamento espectral para que o seus usuários consigam ocupar o mesmo espaço físico sem causar interferências que comprometam a comunicação. Redes baseadas nesse tipo de padrão conseguem atingir velocidades de até 600 Mbits/s e alcance máximo de 250 metros, variando baseado na versão do padrão.

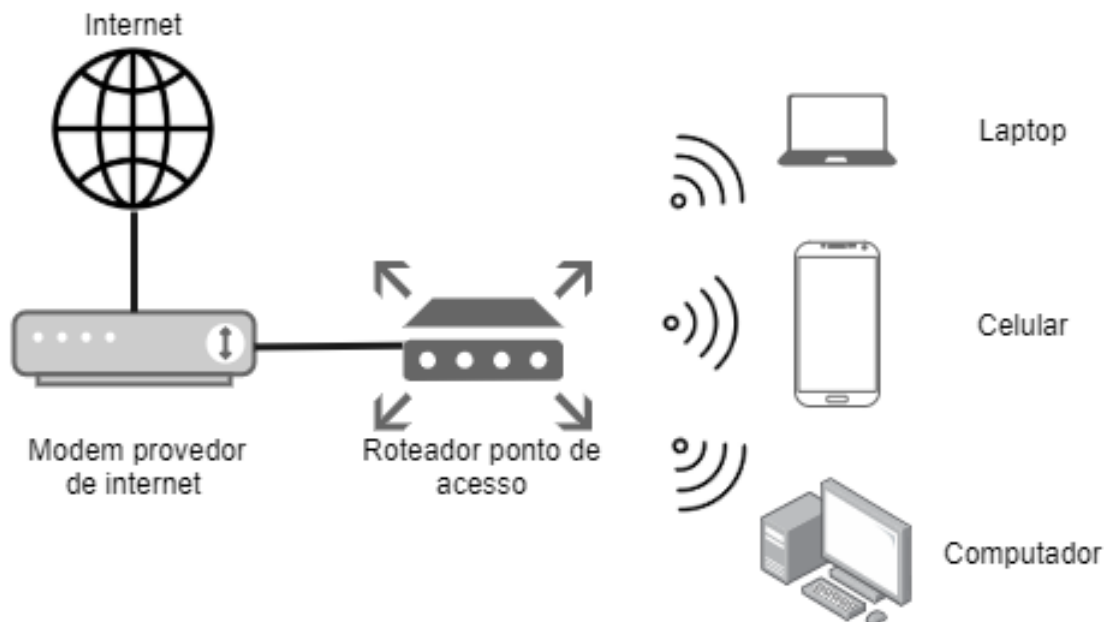
O padrão 802.11b define operação em 2,4 GHz utilizando modulação de sequência direta do espalhamento do espectro e consegue atingir uma taxa de transmissão de até 11 Mbits/s e latência de 3.2 a 17 ms. Semelhante ao tipo b, o padrão 802.11g também opera em frequência 2.4 GHz, porém utilizando modulação de multiplexação por divisão de frequências ortogonais, o que permite atingir velocidades de até 54 MHz. Publicado em 2006, o padrão 802.11n utiliza um esquema de múltiplas entradas e múltiplas saídas, o que permite atingir uma taxa de transferência de 54 Mbits/s até 600 Mbits/s, operando em frequência de 2,4 GHz ou opcionalmente 5 GHz (MAHMOOD; JAVAID; RAZZAQB, 2015).

O nome Wi-Fi é comercialmente mais conhecido para dispositivos que utilizem os protocolos 802.11 da IEEE. Atualmente diversos dispositivos do dia-a-dia utilizam a tecnologia Wi-Fi, tais como: celulares, computadores, consoles de jogos eletrônicos, impressoras e até lâmpadas, tomadas inteligentes e ar condicionados. Uma organização sem fins lucrativos foi criada em 1999 chamada de Wi-Fi Alliance, que tem como objetivo aprimorar e manter os protocolos utilizados nas redes sem fio Wi-Fi, além de certificar dispositivos eletrônicos para que o consumidor tenha a garantia de uma boa experiência. Entretanto, o processo de certificação pode ser custoso, fazendo com que muitos desses dispositivos tenham conformidade com a norma IEEE, porém não sejam certificados.

Um dispositivo chamado de roteador, ilustrado na Figura 5 é usualmente utilizado para criar uma rede local e ao mesmo tempo conectar-se a um modem de internet, fazendo o intermédio entre a comunicação local e a rede mundial de computadores. Tanto o roteador quanto o dispositivo final devem possuir suporte ao padrão 802.11 da IEEE, possibilitando uma grande mobilidade aos dispositivos sem perda de acesso a internet de

alta velocidade.

Figura 5 – Interação entre dispositivos eletrônicos e roteador Wi-Fi.



Fonte: Autor.

2.4 Internet das coisas - IoT

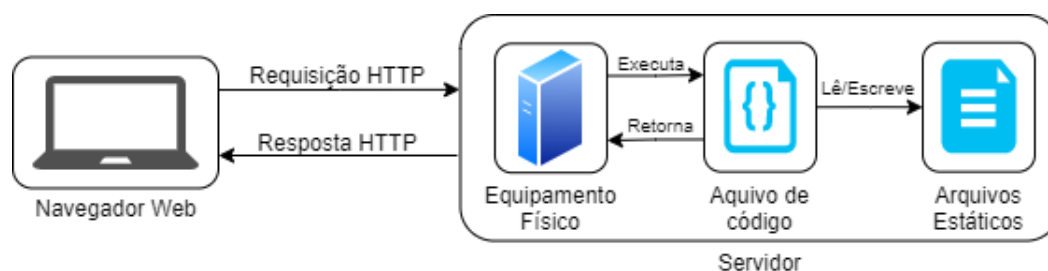
A internet das coisas é um paradigma de comunicação recente que visiona a utilização de objetos do dia-a-dia em conjunto com microcontroladores, que, equipados com um protocolo de comunicação adequado, permite que esses objetivo acessem e façam parte da internet global. Por permitir que diversos tipos de dispositivos acessem a internet, como, por exemplo, eletrodomésticos, relógios, sistemas de segurança, sensores, atuadores, veículos automotivos e outros, uma gama de dados antes inacessíveis ficam disponíveis para utilização em diversas aplicações. Essas informações necessitam de um uma estrutura capaz de armazená-las e processá-las, para que alguma decisão possa ser tomada de acordo com o domínio da aplicação (ZANELLA et al., 2014).

Uma característica importante da Internet das Coisas é a capacidade de integração entre os dispositivos, permitindo que uma decisão mais fundamentada e complexa possa ser tomada. Dessa forma, é necessário que haja um meio de centralização das informações coletadas. Será descrito nas subseções seguintes algumas tecnologias que possibilitam a centralização e o processamento dessas informações.

2.4.1 Servidor Web

O servidor web é o conjunto de código (*software*) e equipamento físico (*hardware*) que trabalhando em junto conseguem entregar arquivos através do protocolo HTTP. Os arquivos são salvos no *hardware* e podem ser do tipo HTML, CSS, Javascript, arquivos de imagem, etc, e são entregues ao usuário final a partir da internet. O diagrama da Figura 6 ilustra de forma simplificada o processo de interação de um navegador web e um servidor. Uma vez que o servidor recebe a requisição HTTP, um determinado código é executado e então um conjunto de arquivos estáticos, tal como um modelo HTML, é acessado e ajustado conforme a requisição enviada. Os arquivos finais são enviados na resposta HTTP, que é processada e renderizada pelo navegador web (MOZILLA, 2019).

Figura 6 – Interação entre um navegador web e um servidor.



Fonte: Autor.

2.4.2 Python e Django

Python é uma linguagem de programação publicada pela primeira vez em 1991 pelo programador Guido van Rossum. É divulgada como uma linguagem de fácil aprendizado tanto para programadores experientes quanto para iniciantes (FOUNDATION, 2020). Devido a sua baixa barreira de entrada, tem ganhado grande notoriedade ao longo dos anos. De fato, a revista IEEE Spectrum listou Python como a linguagem de programação mais utilizada no mundo em 2019, justificando sua popularidade devido ao grande número de bibliotecas especializadas disponíveis para a linguagem.

Além disso, Python é uma linguagem do tipo interpretada, ou seja, executa o código livremente sem a necessidade de traduzi-lo para linguagem de máquina previamente. Também é orientada a objetos, um paradigma que trata variáveis de programação como um elemento modelável onde é possível definir atributos e métodos.

Dentro da diversidade de ferramentas disponíveis especialmente para python, Django é uma *framework* para desenvolvimento *web*, ou seja, um conjunto de bibliotecas que foram agregadas de forma a facilitar o desenvolvimento de um *site*. Entretanto, é importante notar que esta *framework* introduz um paradigma *modelo-template-view*, o que significa, em termos práticos, que uma aplicação deve ser projetada adaptando-se a sintaxe e estrutura de arquivos exigidas pelo Django.

2.4.3 Banco de dados relacional

O banco de dados relacional é uma coleção de dados com relacionamentos pre-definidos. Os dados são salvos em uma tabela onde a coluna guarda um tipo de dado e cada linha representa um novo registro. Cada linha possui um campo único chamado de chave principal e podem possuir chaves estrangeiras, que ligam uma tabela a outra, possibilitando que dados sejam obtidos de maneira recursiva (AMAZON, 2020).

Um relacionamento pode ser definido entre as tabelas do banco de dados. DevMedia (2020) define os tipos de relacionamento possíveis:

- **Relacionamento 1..1 (um para um):** cada uma das duas entidades envolvidas referenciam obrigatoriamente apenas uma unidade da outra;
- **Relacionamento 1..n (um para n):** uma entidade pode referenciar diversas outras, porém as outras entidades referenciadas apontam apenas para uma. É o caso, por exemplo, de um sistema de um site online onde um usuário pode inserir diversos comentários, mas um comentário só pode ser feito por um usuário;
- **Relacionamento n..n (n para n):** uma entidade referencia diversas outras, enquanto que as entidades referenciadas também podem apontar para mais de uma outra entidade. É o caso de um sistema que registra a autoria de um artigo, onde um autor pode publicar diversos artigos, assim como os artigos podem ter sido publicados por diversos autores.

2.5 Sistemas Comerciais Disponíveis

O autor identificou empresas de tecnologia de informação ^{1,2,3} que oferecem soluções de internet das coisas relacionadas a veículos automotivos. Entretanto, não foi identificado uma solução que proponha o registro de utilização veicular por usuários motoristas. As soluções comerciais disponíveis, em geral, focam na rastreabilidade do veículo e na análise de desempenho de condução, tal como o consumo de combustível.

¹ WorldTech. Disponível em <<https://rfid.worldtech.com.br/?gclid=EAIaIQobChMI6LCM15XI6AIVxQmRCh3ptw>>. Acesso em 01 de Abril de 2020

² Fractal. Disponível em: <<https://www.fractal.com/pt/blog/internet-das-coisas-iot-para-gerenciamento-de-frota-de-veiculos>>. Acesso em 01 de Abril de 2020.

³ dti. Disponível em: <<https://dtidigital.com.br/blog/gestao-de-frotas-com-iot/>>. Acesso em 01 de abril de 2020.

3 Métodos e ferramentas

Neste Capítulo será apresentado a concepção do sistema, o tipo e papel dos usuários, projeto do banco de dados, assim como o requisito de funcionamento geral do sistema, onde o comportamento esperado de cada componente será detalhado, assim como suas interações com as demais partes.

De forma geral, existem dois módulos principais no projeto: o sistema físico de autenticação e o sistema virtual de servidor, que inclui a interface web de usuário e o banco de dados.

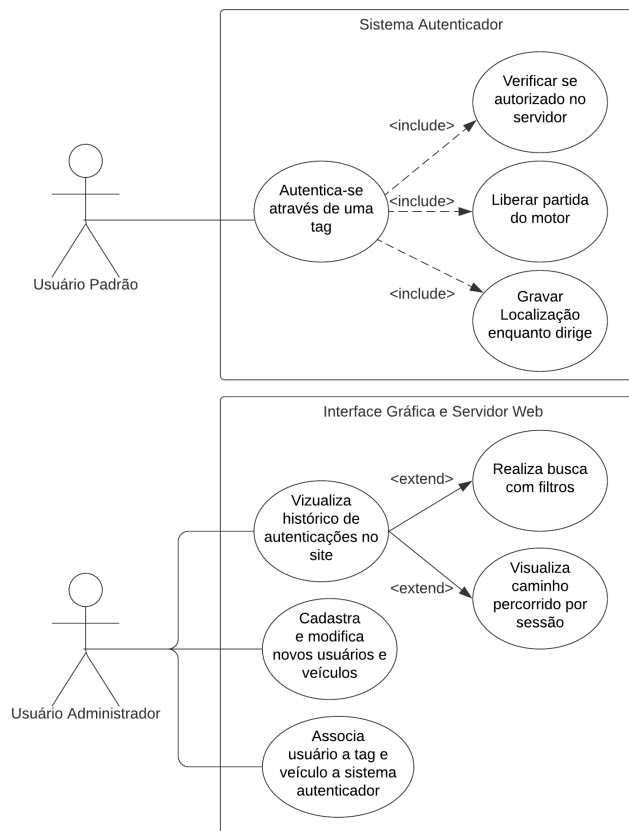
3.1 Sistema de autenticação veicular

Este trabalho propõe um sistema de verificação de usuário motorista no qual deve-se aproximar um cartão RFID a um leitor que estará presente no veículo. Após a leitura, o sistema autenticador irá consultar um servidor web para verificar se aquele determinado cartão está autorizado para dar partida ao motor. Caso o usuário esteja autorizado, a partida é liberada, e então o sistema acionará um relé, conectando a bateria do carro ao motor de partida e permitindo que o usuário gire a chave de ignição e parta o motor de combustão interna. Caso o usuário não esteja autorizado, o sistema não acionará o relé, e então a bateria do carro estará fisicamente desconectada do motor de partida, impossibilitando a partida do motor de ignição.

O sistema também deve obter a localização geográfica a partir de um módulo GPS e enviar para um servidor, onde será salvo em um banco de dados.

Com base nesses requisitos, foram criados dois tipos de usuários: o usuário motorista e o usuário administrador. O papel que cada um desempenha é descrito na Figura 7. A palavra *include* indica que quando a ação à esquerda for executada, a ação à direita também será executada, enquanto que a palavra *extend* indica uma execução opcional, a critério do usuário. Observa-se que o usuário motorista precisa somente aproximar a tag RFID ao leitor, e, caso ele esteja registrado e autorizado no sistema, a partida do motor será liberada. O sistema ficará obtendo a posição geográfica continuamente e enviando essa informação ao servidor web para que seja salvo no banco de dados.

Figura 7 – Diagrama de casos de uso do sistema de autenticação veicular.



Fonte: Autor.

O usuário administrador é o segundo tipo de usuário e possui acesso a um site web no qual é possível adicionar novos usuários veiculares e associá-los a uma tag RFID. Esse mesmo usuário pode também visualizar o histórico de tentativas de autenticação feitas pelos usuários motoristas, além de administrar e visualizar toda a base de dados.

Quando uma nova tag precisa ser cadastrada, e uma vez que cada tag RFID possui um número único, é necessário que o administrador aproxime essa tag a um sistema leitor, que envia o número de identificação para o servidor web e salva essa informação no banco de dados. Dessa forma, o administrador pode associar um usuário motorista a uma tag existente.

A associação de usuários a uma tag é útil pois permite a visualização fácil de uma autenticação. Por exemplo, caso o usuário motorista “Pedro” seja associado a uma tag de número X, uma vez que for identificado a aproximação de uma tag de número X pelo sistema autenticador, é possível inferir que o usuário “Pedro” está tentando se autenticar

para dar partida no motor do veículo. Outra vantagem de realizar essa associação virtual é a possibilidade de trocar facilmente qual usuário está associado a qual tag em casos de perda de tag, por exemplo.

3.2 Sistema leitor de autenticação veicular

Visto a necessidade de centralizar as informações obtidas de forma rápida e remota, faz-se necessário que o sistema físico possua algum tipo de comunicação confiável. Uma vez que o sistema será instalado em um automóvel, é necessário que essa comunicação abranja a maior área possível. Sabendo disso, é fácil perceber que a rede de telefonia móvel 3G ou superior atende a essa exigência.

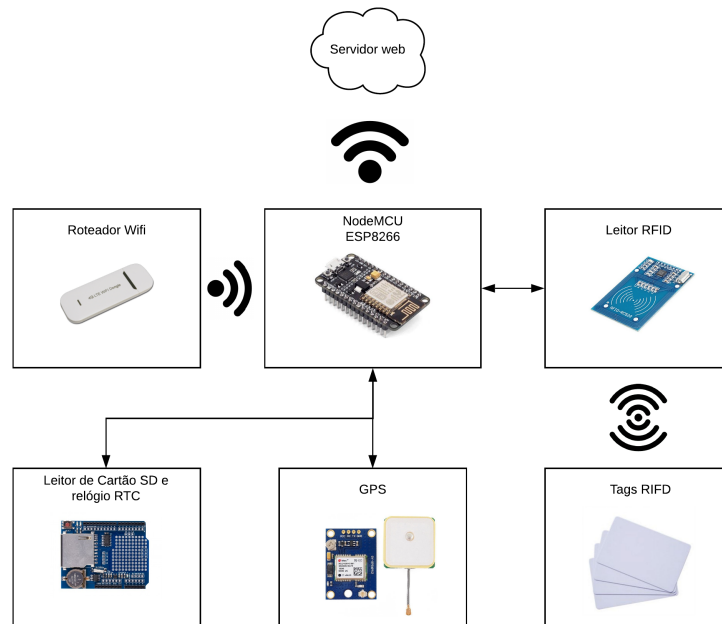
As opções encontradas pelo autor de conectividade móvel para microcontroladores se limitaram a rede 2G, entretanto. Visto que essa rede não oferece velocidade suficiente para o constante envio de localização, além de possuir baixa confiabilidade comparado à rede 3G, foi decidido, então, que o microcontrolador deveria possuir conectividade WiFi e um modem adaptador deveria ser utilizado em conjunto. Uma outra vantagem observada em utilizar um modem adaptador separado é o fato de que esse pode ser facilmente substituído a fim de aproveitar uma melhor cobertura que uma região geográfica pode oferecer, uma vez que há uma grande discrepância das faixas de frequências utilizadas pela rede móvel entre os diversos países do mundo.

Pensando nisto, é proposto um sistema físico ilustrado na Figura 8. O modem adaptador é genérico pois depende da região em que o sistema é utilizado. Com a conectividade garantida, o microcontrolador ESP8266 foi utilizado uma vez que este possui suporte para redes WiFi. O funcionamento de cada um dos módulos será detalhado nas seções seguintes. É importante notar que, quando integrado a um veículo, o sistema será alimentado pela bateria do carro apenas quando a chave estiver na penúltima posição, antes da posição de partida do motor.

3.2.1 ESP8266 e NodeMCU

O ESP8266 é um microcontrolador do tipo sistema-em-um-chip que possui suporte a comunicação sem fio (*WiFi*) de 2,4 GHz com suporte a WPA/WPA2 e protocolo 802.11 b/g/n. Possui diversas funcionalidades, tais como: entrada e saída digital (*GPIO*), comu-

Figura 8 – Diagrama de componentes do sistema leitor de autenticação.

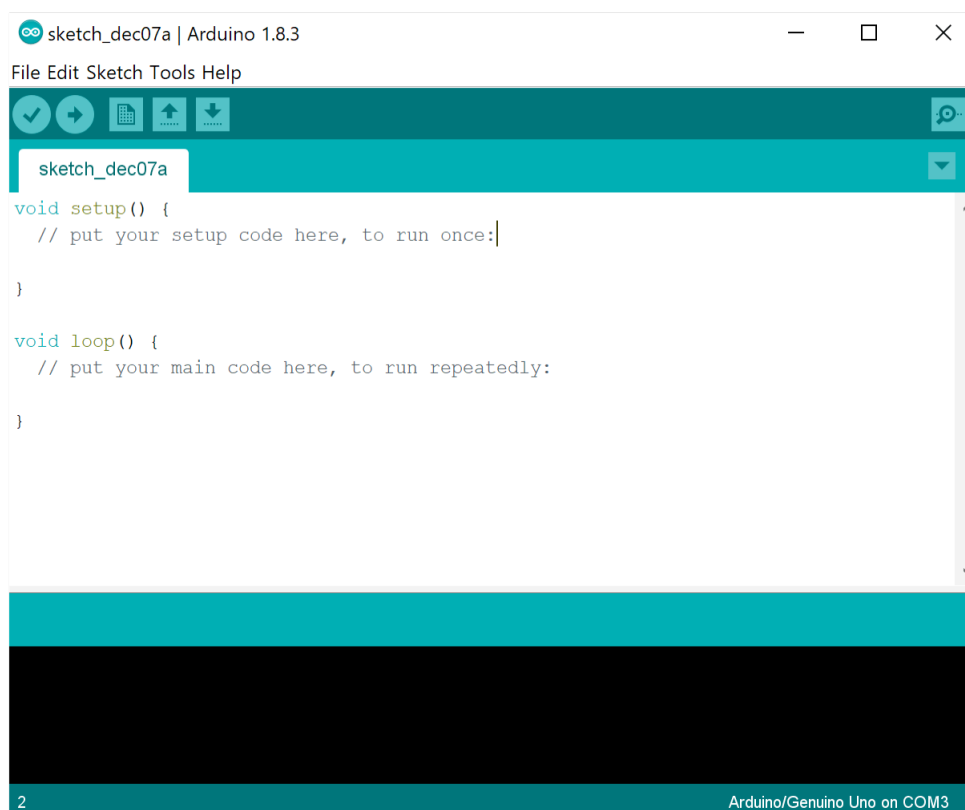


Fonte: o Autor.

nicação I2C, SPI e UART, modulação por largura de pulso (PWM), 64 KBytes de RAM para instruções, 96 KBytes de RAM para dados, conversor analógico-digital de 10 bits, entre outras funcionalidades. A unidade de processamento é do tipo 106 micro LX3 da fabricante Tensilica, enquanto o módulo completo é feito pela fabricante chinesa Espressif (ESPRESSIF, 2020).

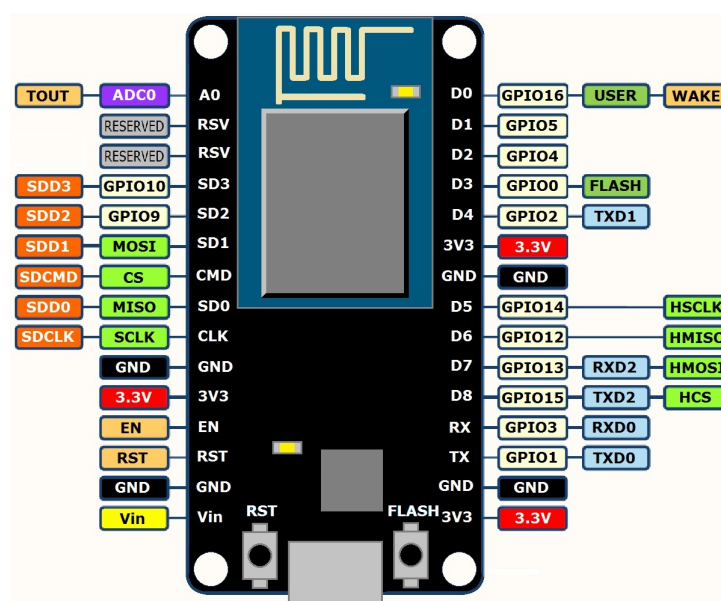
O NodeMCU é um *firmware* desenvolvido para o ESP8266 que facilita o processo de prototipagem. É implementado na linguagem de programação C e inicialmente desenvolvido para ser utilizado com a linguagem Lua, porém posteriormente adaptado para programação no ambiente de desenvolvimento *Arduino IDE* que utiliza a linguagem C/C++ como mostra a Figura 9.

O *firmware* NodeMCU é comumente embarcado em uma placa de desenvolvimento, conforme ilustrado na Figura 8. Essa placa possui os circuitos de condicionamento necessários para utilizar as funcionalidades do microcontrolador ESP8266, além de fazer possível o carregamento de código por meio de um cabo USB, que é conectado diretamente ao computador. O diagrama de pinos é mostrado na Figura 10 e é importante pois é através dele que é possível localizar quais pinos dão acesso a determinadas funções do microcontrolador, tal como comunicação SPI.

Figura 9 – Ambiente de desenvolvimento *Arduino IDE*.

Fonte: Arduino.cc.

Figura 10 – Diagrama de pinos da placa de desenvolvimento baseada em Node-MCU.



Fonte: Arduining.com, adaptado.

3.2.2 Tag e Leitor RFID MFRC522

O dispositivo MFRC522 é um circuito integrado para leitura/escrita de tags RFID do tipo ISO 14443 MIFARE. Além disso, suporta comunicação do tipo SPI, I2C ou UART e é produzido pela fabricante *NXP Semiconductors*. É apresentado na Tabela 1 um resumo das especificações elétricas que devem ser consideradas durante o projeto do sistema.

Tabela 1 – Especificações elétricas de operação do Leitor RFID MF.

| | |
|-----------------------------|------------------|
| Frequência de Operação | 13,56 MHz |
| Comunicação do hóspede | SPI / I2C / UART |
| Faixa de tensão operacional | 2,5 V a 3,3 V |
| Corrente de operação máxima | 13-26 mA |
| Corrente de operação mínima | 10 μ A |
| Tensão de entrada lógica | Tolerante a 5V |
| Alcance de leitura | 5 cm |

Fonte: *NXP Semiconductors*.

A tag utilizada foi do tipo passiva devido ao custo reduzido e ao fato de que um curto alcance de comunicação ser uma funcionalidade desejada, uma vez que isso aumenta a segurança do sistema, já que o usuário poderá se identificar apenas quando dentro do veículo.

3.2.3 Leitor de cartão SD e relógio RTC

O componente mostrado na Figura 8 é uma placa que possui um leitor de cartão SD formatado em FAT16 ou FAT32 com funcionamento em 3.3V e Utiliza a interface do tipo SPI para comunicação. Além disso, um relógio em tempo real do tipo RTC DS1307 também é incluso na placa, possibilitando que a informação de data e hora não seja perdida mesmo quando a alimentação do circuito principal é cessada, uma vez que uma bateria do tipo CR1220 alimenta o circuito integrado todo o tempo, tendo vida útil de, em média, cinco anos. A utilização desse módulo faz-se necessária pois o sistema deve manter um cópia local do histórico de autenticação e de posição geográfica para que, em último caso, possa ser resgatado manualmente.

3.2.4 Algoritmo do microcontrolador

Para que o sistema leitor realize suas tarefas com robustez, foi desenvolvido o fluxograma, ilustrado na Figura 11, com o algoritmo que deve ser seguido durante o desenvolvimento do código. O fluxograma foi dividido em seções para que comentários possam ser feitos.

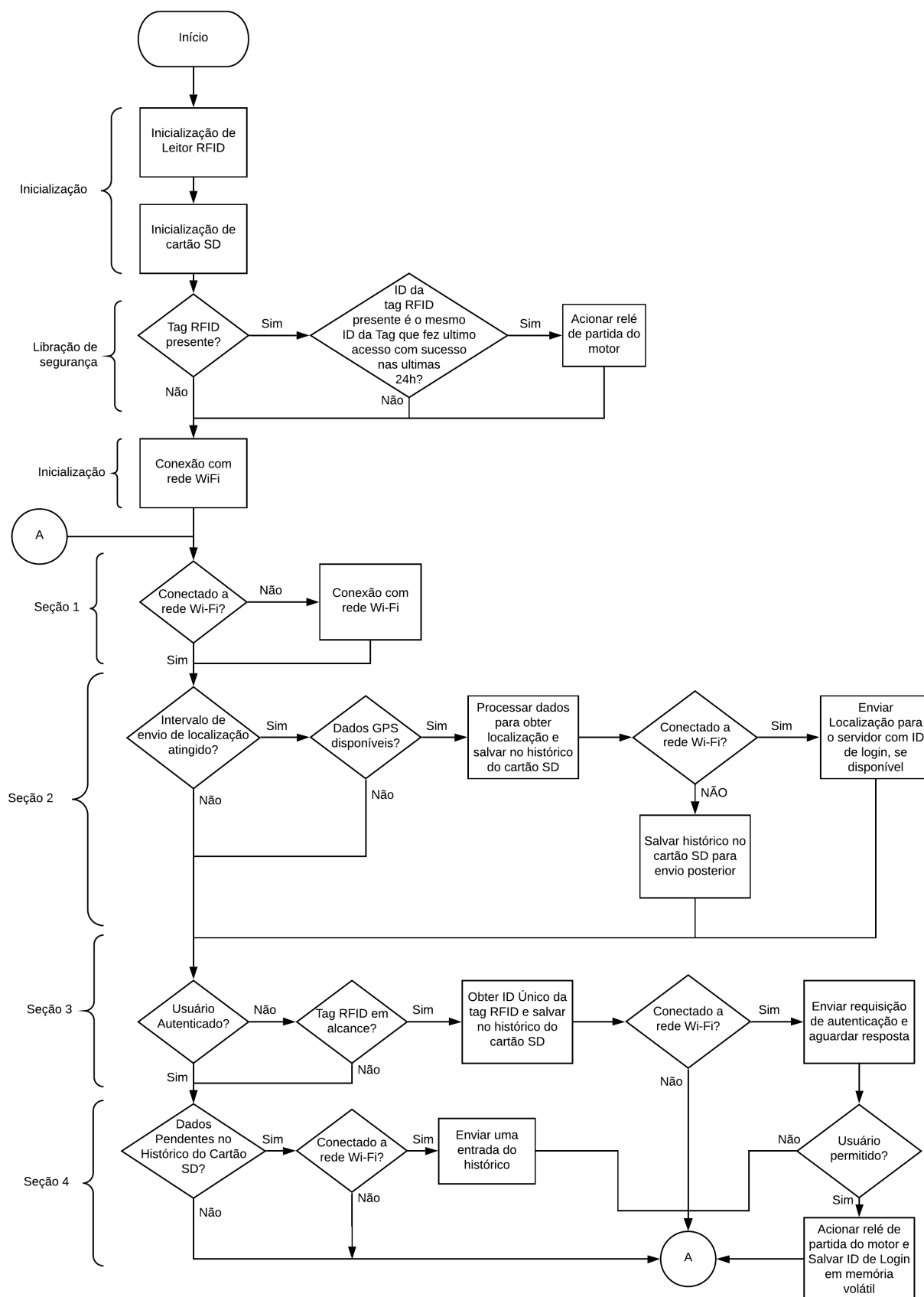
Ao ser ligado, o microcontrolador deve inicializar a comunicação com o leitor RFID e cartão SD. Antes de tentar se conectar com a rede Wi-Fi, o microcontrolador deve verificar se existe alguma tag RFID próxima para leitura. Caso exista, o cartão SD deve ser consultado e o ID, data e hora da última autenticação feita com sucesso deve ser obtida. Caso a Tag próxima do sistema tenha o mesmo ID da tag consultada no cartão SD, e o login tenha sido feito nas últimas 24 horas, o microcontrolador irá energizar o relé que conecta a bateria do carro ao motor de partida, permitindo que o usuário motorista conclua o acionamento do motor a combustão. Essa funcionalidade assegura que o motor a combustão possa ser ligado o mais rápido possível após uma falha, já que a indisponibilidade de torque em um veículo pode colocar em risco a vida do condutor. Isso também assegura que o condutor ainda possa partir o motor caso ele autentique-se e conduza o veículo para um lugar remoto, onde não há cobertura de rede móvel 3G ou superior, ou ainda mesmo em uma falha temporária da rede móvel.

Após a liberação por segurança, o microcontrolador deve tentar conectar-se com a rede Wi-Fi e verificar continuamente se continua conectado, repetindo o procedimento de conexão novamente caso contrário. Esse procedimento é ilustrado na Seção 1 do fluxograma.

Na Seção 2 do fluxograma, detalha-se o procedimento de envio de localização ao servidor. O microcontrolador deve enviar a posição geográfica com uma periodicidade pré-definida. Quando esse período é atingido, o microcontrolador comunica-se com o módulo GPS, que envia a posição por meio dos pinos TX e RX. O microcontrolador processa as informações recebidas do módulo GPS e envia ao servidor, caso a conexão Wi-Fi esteja estabelecida, ou salva localmente no cartão SD para envio posterior, caso contrário. Dessa forma, os dados não são perdidos durante momentos de instabilidade da rede móvel.

O sistema detecta que o cartão RFID está presente quando o microcontrolador atinge a posição ilustrada na Seção 3 do fluxograma. Assumindo que o laço de repetição

Figura 11 – Algoritmo utilizado para desenvolver o código do microcontrolador.



Fonte: Autor.

é executado diversas vezes em um segundo, o microcontrolador irá detectar a presença da tag RFID assim que o usuário realizar a aproximação da tag. O microcontrolador escreve no cartão SD o ID único da tag e a hora que a aproximação foi realizada e, caso conectado a rede Wi-Fi, envia ao servidor. O servidor recebe o ID único da tag lida e retorna ao microcontrolador o ID de login salvo no banco de dados e a confirmação informando se a autenticação foi válida ou não. Caso o servidor retorne que tag é válida, o microcontrolador aciona o relé que conecta a bateria do carro ao motor de partida, além de salvar em uma variável interna que a autenticação foi feita com sucesso, e então irá pular a Seção 3 do algoritmo nos laços seguintes da execução.

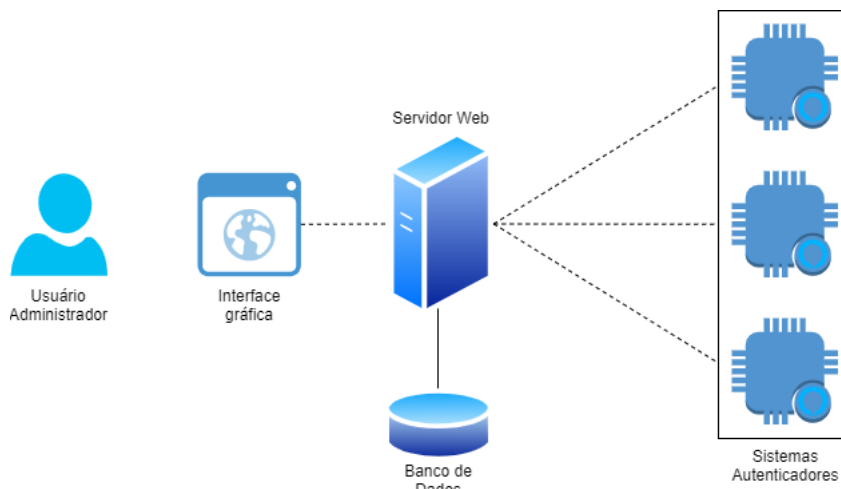
Por fim, a Seção 4 do algoritmo verifica se existe algum dado de posição geográfica pendente para envio no cartão SD que foi escrito previamente durante a Seção 2. O código representado pela Seção 4 só será executado quando uma autenticação for feita com sucesso, pois é priorizado a leitura da tag RFID, uma vez que o envio de informações ao servidor bloqueia momentaneamente a execução de código do microcontrolador.

3.3 Sistema virtual: servidor

O servidor tem como objetivo fazer o intermédio entre o usuário administrador, sistemas autenticadores e banco de dados, com o objetivo de centralizar todas as informações. O diagrama da Figura 12 ilustra essa interação. Os sistemas autenticadores adquirem os dados, que são enviados diretamente ao servidor web através da estrutura de internet móvel. Então, o servidor armazena as informações recebidas em um banco de dados e o usuário acessa uma página web que dá acesso a um menu de navegação para que, então, a informação desejada seja exibida.

A Figura 13 é um diagrama de sequência que ilustra as operações que o usuário administrador pode fazer enquanto navega na página web. Para registrar uma tag RFID, o usuário administrador deve aproximar a tag a um sistema leitor para que seja feita a leitura de ID único, que é enviado ao servidor e salvo automaticamente. Os sistemas leitores são configurados para enviar o endereço MAC próprio automaticamente quando conectados a uma rede Wi-Fi, portanto, não necessitam ser registrados manualmente. Para criar um usuário novo, o usuário administrador deve clicar no botão “Adicionar Usuário” e então navegador web irá fazer uma solicitação ao servidor, que redireciona

Figura 12 – Diagrama de interação entre os componentes físicos, servidor web e usuário administrador.



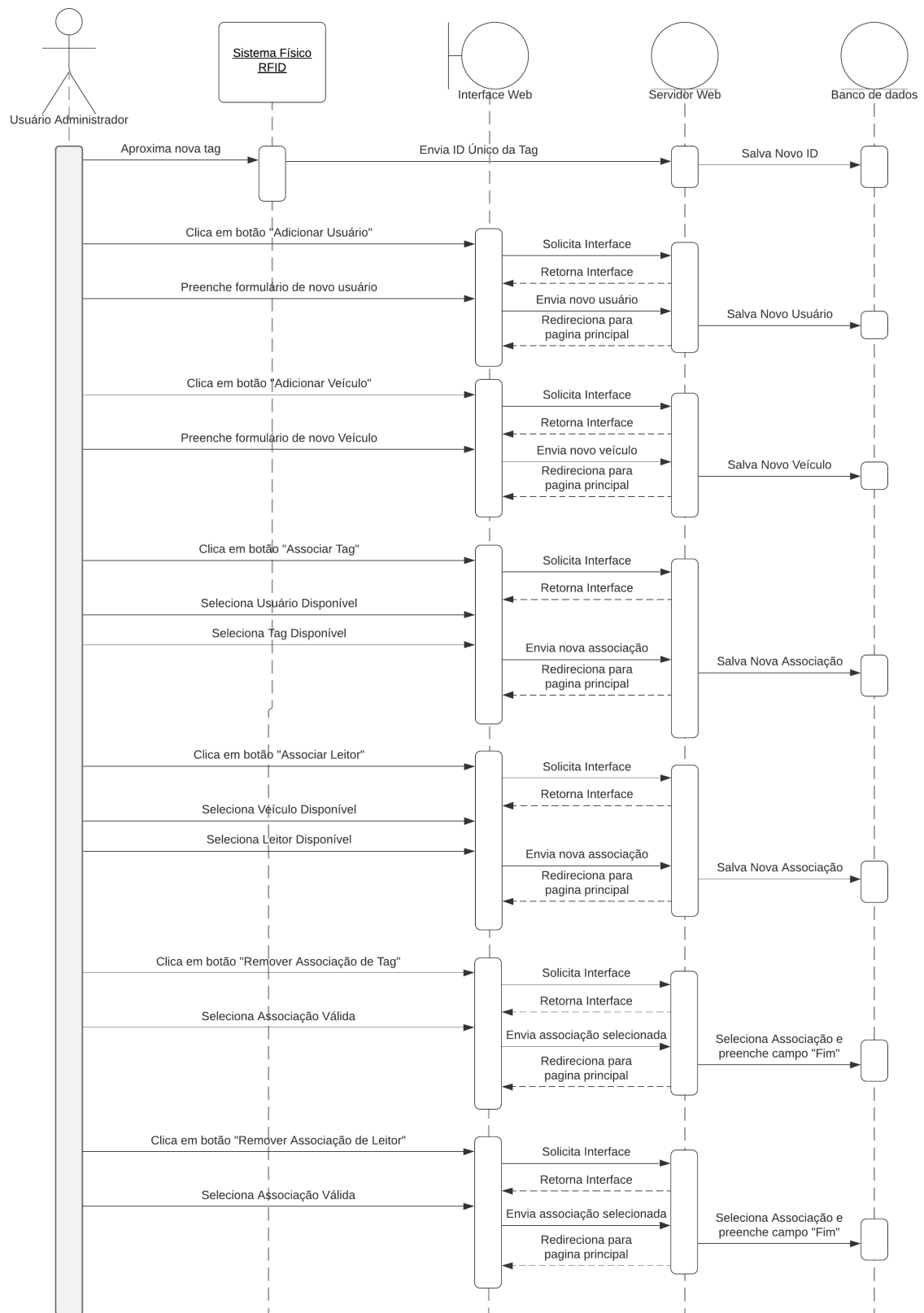
Fonte: Autor.

para uma página que contém um formulário onde o administrador pode inserir o nome do usuário, documento de identificação (opcional) e informações extras (opcional). Quando o usuário deseja registrar um novo veículo, ele deve clicar no botão “Adicionar Veículo”, e será redirecionado para o formulário adequado, onde poderá preencher as informações de “Identificação de Placa” e “Descrição” para que possa ser identificado mais facilmente. Para associar um usuário motorista a uma tag, o administrador deve clicar no botão “Associar Tag”, e então será redirecionado para um formulário, onde deverá selecionar um usuário disponível e uma tag disponível, ou seja, sem associações válida. Quando o usuário administrador desejar terminar uma associação de tag ou leitor, ele deverá clicar nos botões “Remover Associação de Tag” e “Remover Associação de Leitor”, onde será redirecionado a um formulário simples que exibe uma lista de associações válidas.

O diagrama do modelo entidade-relacionamento é mostrado na Figura 14, na qual a linha que leva uma tabela a outra representa um tipo de relacionamento, conforme mencionado na Fundamentação Teórica. O traço perpendicular representa relacionamento “um”, enquanto a ramificação em três traços representam o relacionamento “n”. A função de cada entidade será comentada a seguir.

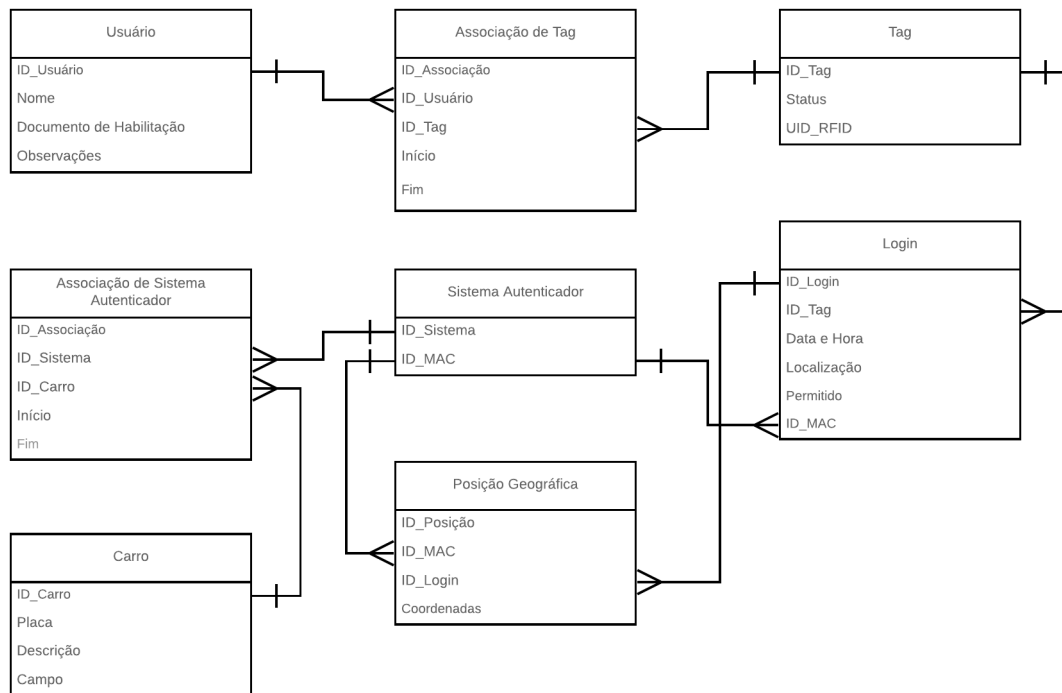
- **Usuário:** Guarda informações do usuário motorista. O nome deve ser inserido obrigatoriamente, mas a inserção do número de documento de habilitação e observações é opcional, servindo como conveniência para o administrador. O usuário possui

Figura 13 – Diagrama de sequência das operações feitas pelo usuário administrador.



Fonte: Autor.

Figura 14 – Modelo entidade-relacionamento do banco de dados.



Fonte: Autor.

zero a n associações de tag, porém apenas uma ativa ao mesmo tempo.

- Associação de Tag:** Guarda qual usuário está associado com qual tag. O campo início e fim são obrigatórios e sinalizam a duração da validade da associação. No entanto, o administrador não precisa se preocupar em preencher esses dados quando realizar uma nova associação, pois o servidor os preenche automaticamente com a data atual e uma data consideravelmente distante. Um usuário motorista será permitido dar partida no motor de combustão quando o campo de início for anterior a data atual e o campo de fim for posterior à data atual. Quando o administrador desejar que um determinado usuário não consiga se autenticar no veículo, o servidor preencherá a data Fim com a data atual.
- Tag:** Guarda o ID único da tag RFID. O campo *ID_Tag* é auto-incrementado e serve também para indicar um número legível para o administrador, que pode imprimir este número diretamente no cartão físico. Além disso, possui o campo *Status*, que pode assumir o valor “Ativado” ou “Desativado”, onde o administrador pode suspender temporariamente o *login* de uma determinada tag, fazendo com que

as autenticações sejam negadas. Para retomar, basta ativar a tag novamente.

- **Login:** Guarda as informações de tentativas de login dos sistemas autenticadores. Quando o usuário motorista aproxima a tag RFID, o sistema faz a leitura do número único da tag e automaticamente envia essa informação ao servidor, juntamente com o endereço MAC do módulo Wi-Fi. O servidor, então, armazena essas informações na tabela Login, juntamente com a hora do servidor. Então, a tabela de Associação de tags é consultada para determinar se o usuário está cadastrado e liberado. Essa decisão é salva no campo “Permitido” a fim de reduzir o número de consultas posteriores.
- **Sistema Autenticador:** Da mesma forma da tabela Tag, os sistemas autenticadores são automaticamente cadastrados quando conectados com o servidor, no qual o seu endereço MAC é salvo e um número legível é dado para o administrador.
- **Associação de Sistema Autenticador:** Quando um sistema autenticador é instalado no carro, é necessário que seja indicado qual carro utiliza qual sistema autenticador, e então essa informações é salva nesta tabela, facilitando a utilização do usuário administrador, que conseguirá ver diretamente o veículo utilizado.
- **Carro:** Cadastra as informações do carro, tal como o número da placa, no campo “Placa”, ou detalhes do modelo e/ou ano, no campo “Descrição”, que será exibida na interface de usuário para que o usuário administrador consiga identificar um veículo facilmente.
- **Posição geográfica:** Salva as aquisições de posição geográfica enviadas pelo GPS, onde cada ponto será salvo como uma linha da tabela. Dado um login, é possível resgatar todos os pontos de uma determinada sessão e, então, traçar uma rota no mapa.

4 Resultados

Neste Capítulo será abordado a execução do código desenvolvido para implementar o servidor web. Também será mostrado um esquemático do circuito utilizado para atender as especificações de requisito do módulo leitor, e então a execução do código embarcado no microcontrolador será comentado. Os resultados obtidos serão relatados com o objetivo de validar o correto funcionamento do projeto, além de assegurar a integração entre os módulos do sistema.

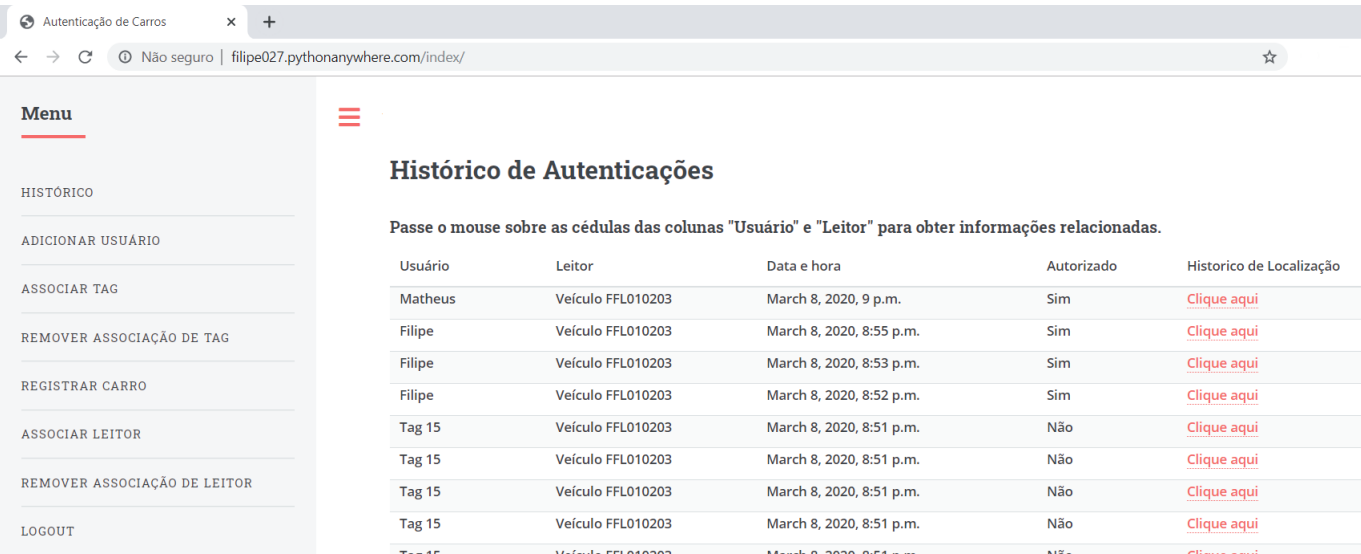
4.1 Servidor Web

Ao acessar a página web, o usuário administrador deve inserir o login e senha correta para ter acesso as demais funcionalidades do site. É possível observar que um menu está disponível a esquerda, porém as subseções do site necessitam que o login seja feito previamente. Uma vez que o usuário administrador faça a autenticação em uma dessas subseções, todas as outras ficam disponíveis para o mesmo navegador web. Caso o administrador deseje encerrar a seção, um botão de logout está disponível no menu de navegação.

A página principal está acessível através do botão “Histórico”. Uma tabela é exibida nesta página, onde é cada linha representa uma tentativa de login feita em um sistema autenticador. Na primeira coluna é detalhado qual Usuário fez a tentativa de login, caso exista uma associação de tag válida no momento em que o login foi feito, ou número da tag, caso contrário. Na segunda coluna, é exibido em qual veículo foi feito a autenticação veicular caso exista associação de leitor válida ou, caso não exista, o número do leitor salvo no banco de dados. Mostra-se na terceira coluna a data e hora em que o login foi feito. Caso o usuário motorista obtenha sucesso na autenticação, a quarta coluna exibirá “Sim”, caso contrário, a coluna exibirá “Não”. Por fim, está disponível na quinta coluna um link para uma página externa que mostra o histórico de localização capturado durante condução do motorista.

Ao acessar o link da quinta coluna, o usuário é redirecionado para um mapa estático que mostra o caminho percorrido pelo motorista, como ilustrado na Figura 16. Os pontos

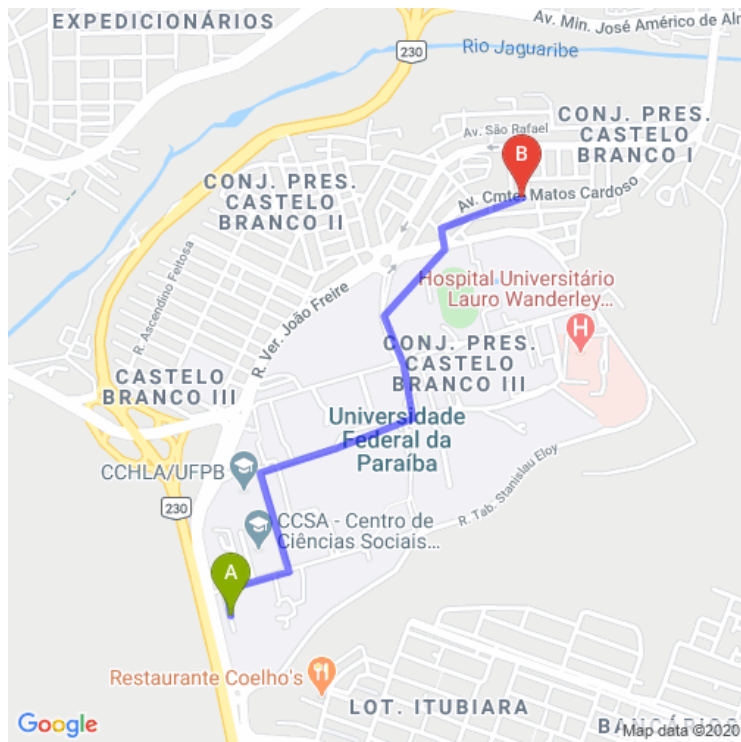
Figura 15 – Página inicial do site.



Fonte: Autor.

de localização coletados pelo sistema leitor são conectados em linha reta, e um marcador de cor verde e com letra “A” denota o ponto de início da seção, enquanto outro marcador de cor vermelha e com letra “B” denota o ponto final da seção, ou seja, a última posição geográfica recebida.

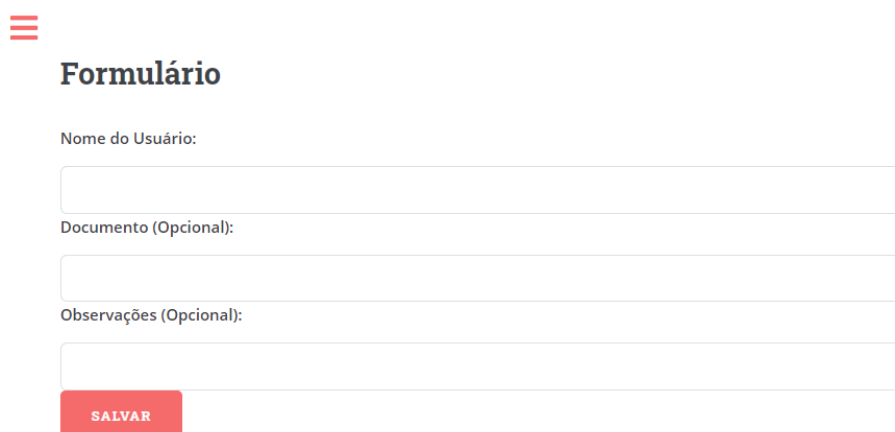
Figura 16 – Mapa estático com o caminho percorrido pelo veículo durante uma sessão.



Fonte: Autor.

Ao clicar no botão “Adicionar Usuário”, disponível no menu de navegação, o administrador acessa um formulário onde três campos estão disponíveis, conforme ilustrado na Figura 17. No primeiro campo é possível preencher o nome do usuário motorista que será utilizado na primeira coluna da tabela da página principal (Figura 15). Os segundos e terceiros campos são detalhes que ajudam a identificar o condutor e serão salvos no banco de dados para uso posterior, porém o preenchimento é opcional.

Figura 17 – Formulário de registro de usuário.



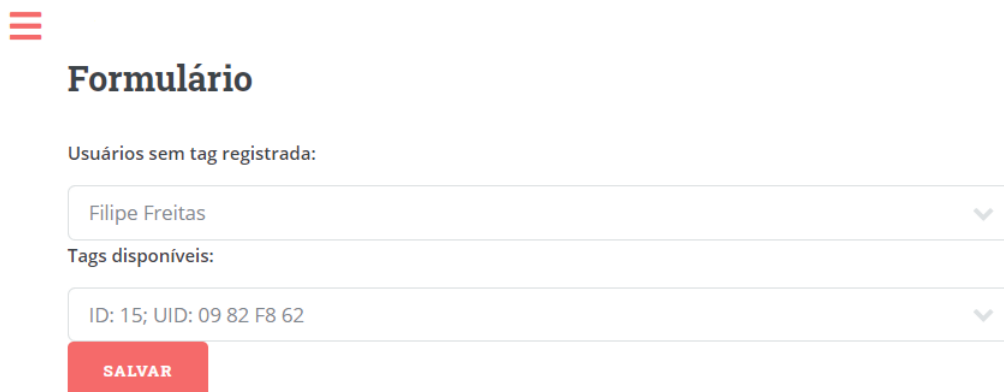
O formulário, intitulado "Formulário", possui um ícone de menu hambúrguer no canto superior esquerdo. Ele contém três campos de entrada de texto, cada um precedido por um rótulo: "Nome do Usuário:", "Documento (Opcional):" e "Observações (Opcional):". Abaixo dos campos, há um botão vermelho com o texto "SALVAR" em branco.

Fonte: Autor.

Quando o administrador aproximar uma tag RFID a um sistema leitor que esteja conectado a internet, o ID único da tag é salvo automaticamente no banco de dados. Após cadastrar o usuário motorista, é possível associar um usuário a uma tag. Essa associação é importante pois permite a liberação da partida do motor quando a tag selecionada é aproximada ao sistema leitor. Para realizar a associação, o administrador deve clicar no botão “Associar Tag”, disponível no menu de navegação e então um formulário será exibido, conforme a Figura 18 com dois campos do tipo opção suspensa, ou seja, que só aceitam valores pré-determinados. O primeiro campo chamado de “Usuário sem tag registrada” mostra os usuários cadastrados que não possuem uma associação válida, enquanto o segundo campo com nome “Tags disponíveis” mostra as tags cadastradas que não possuem um usuário associado. Após selecionar as opções conforme julge apropriado, o administrador deve clicar no botão “Salvar”, e então uma associação será salva no banco de dados.

Quando for necessário encerrar a associação de um usuário a uma tag, por exemplo, em caso de rescisão de contrato de um usuário motorista, o administrador deve clicar

Figura 18 – Formulário de associação de tags.



O formulário, intitulado "Formulário", possui um ícone de menu hambúrguer no canto superior esquerdo. Ele contém duas seções de seleção: "Usuários sem tag registrada:" com uma caixa de texto contendo "Filipe Freitas" e uma seta para baixo; e "Tags disponíveis:" com uma caixa de texto contendo "ID: 15; UID: 09 82 F8 62" e uma seta para baixo. Abaixo dessas seções, há um botão vermelho com o texto "SALVAR" em branco.

Fonte: Autor.

no botão “Remover associação de tag”, disponível no menu de navegação. Neste formulário, apenas um campo está disponível que mostra as associações consideradas válidas. Após selecionar qual associação deve ser encerrada, o administrador deve clicar no botão “Salvar”. É importante notar que esta ação não remove nenhuma informação do banco de dados, mas apenas preenche o campo “Fim” da tabela “Associação de Tag” (Figura 14), preservando o histórico de autenticações.

Figura 19 – Formulário de encerramento de associação de tags.

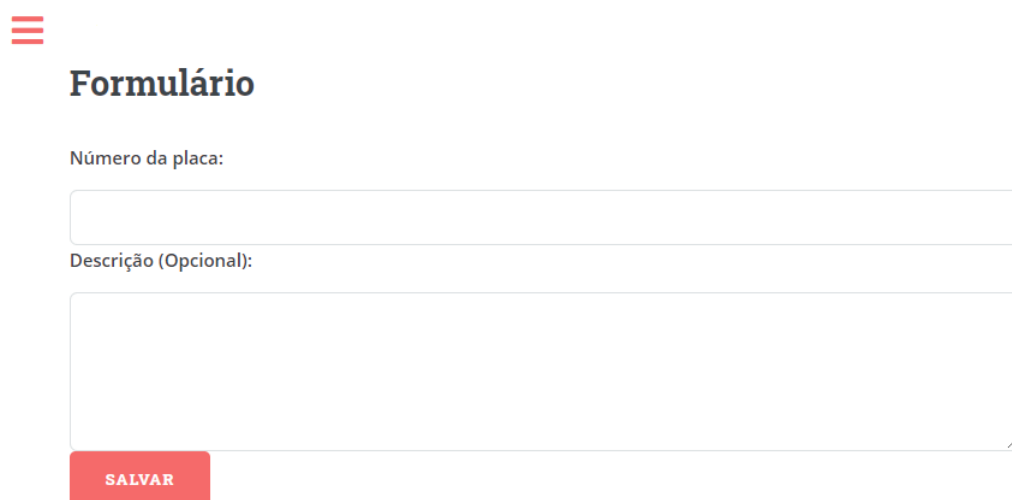


O formulário, intitulado "Formulário", possui um ícone de menu hambúrguer no canto superior esquerdo. Ele contém uma seção de seleção: "Associações ativas:" com uma caixa de texto contendo "Tag 15 -> Filipe Freitas" e uma seta para baixo. Abaixo dessa seção, há um botão vermelho com o texto "SALVAR" em branco.

Fonte: Autor.

De forma semelhante a adição de um novo usuário, o administrador pode registrar um novo veículo no site. Esta opção está disponível ao clicar no botão “Adicionar carro”, e então um formulário será exibido, como ilustrado na Figura 20. Dois campos estão disponíveis para o administrador preencher, o primeiro é a placa do veículo, informação que será exibida na segunda coluna da página principal (Figura 15) e o segundo é onde quaisquer informações extras podem ser inseridas. Essas informações são salvas no banco de dados e ficam disponíveis para uso em qualquer página do site.

Figura 20 – Formulário de registro de um novo veículo.

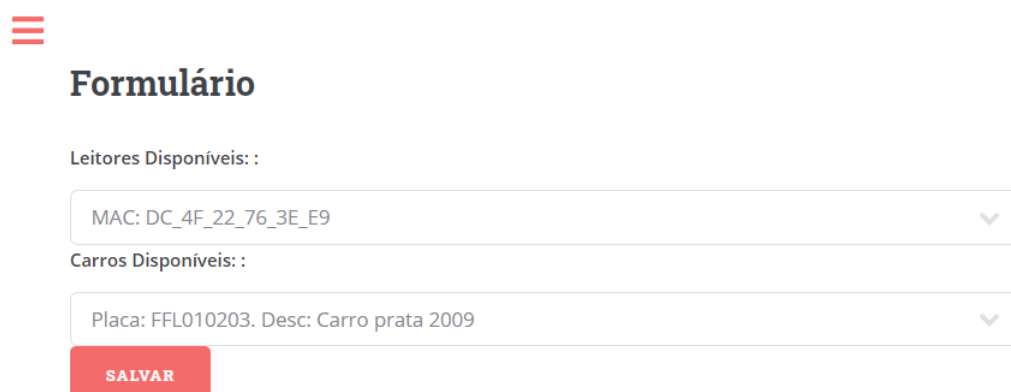


O formulário possui um ícone de menu hambúrguer no canto superior esquerdo. O título "Formulário" está em negrito. Abaixo dele, há o rótulo "Número da placa:" seguido de um campo de entrada de texto. Logo abaixo, há o rótulo "Descrição (Opcional):" seguido de um campo de entrada de texto maior. No canto inferior direito do campo de descrição, há um ícone de cursor. No canto inferior esquerdo do formulário, há um botão vermelho com o texto "SALVAR" em branco.

Fonte: Autor.

O administrador também pode associar um veículo a um leitor, fazendo com que seja possível exibir a informação de um veículo na segunda coluna da tabela da página principal (Figura 15). Semelhante a associação de tag, o administrador pode associar um sistema leitor a um veículo. O leitor é identificado através do seu endereço MAC, que é enviado para o servidor quando conectado a internet e salvo automaticamente no banco de dados. Após selecionar o leitor e o veículo, o usuário administrador deve clicar no botão “Salvar”. Ilustra-se na Figura 21 o formulário que o administrador acessa.

Figura 21 – Formulário de associação de um leitor a um veículo.



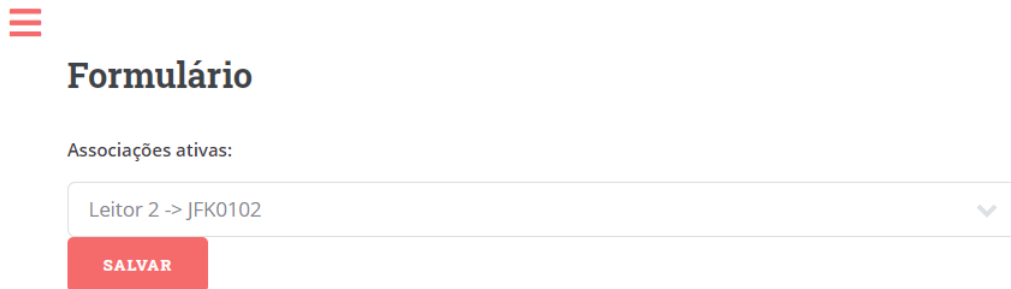
O formulário possui um ícone de menu hambúrguer no canto superior esquerdo. O título "Formulário" está em negrito. Abaixo dele, há o rótulo "Leitores Disponíveis: :" seguido de um campo de seleção com o valor "MAC: DC_4F_22_76_3E_E9" e uma seta para baixo. Logo abaixo, há o rótulo "Carros Disponíveis: :" seguido de um campo de seleção com o valor "Placa: FFL010203. Desc: Carro prata 2009" e uma seta para baixo. No canto inferior esquerdo do formulário, há um botão vermelho com o texto "SALVAR" em branco.

Fonte: Autor.

A ultima opção disponível para o usuário administra é “Remover associação de leitor”, que deve ser usada quando, por exemplo, para fazer a transferência de um sistema

leitor de um veículo para outro. O processo semelhante ao formulário “Remover associação de tag”, no qual o administrador deve selecionar qual associação deseja encerrar a clicar no botão “Salvar”. O formulário é ilustrado na Figura 22.

Figura 22 – Formulário de encerramento de associação de leitor.

A interface do formulário para encerrar uma associação de leitor. No topo, há um ícone de menu hambúrguer vermelho à esquerda e o título "Formulário" em negrito. Abaixo, o texto "Associações ativas:" precede uma caixa de seleção com o texto "Leitor 2 -> JFK0102" e uma seta para baixo. Logo abaixo da caixa de seleção, há um botão retangular vermelho com o texto "SALVAR" em letras maiúsculas brancas.

Fonte: Autor.

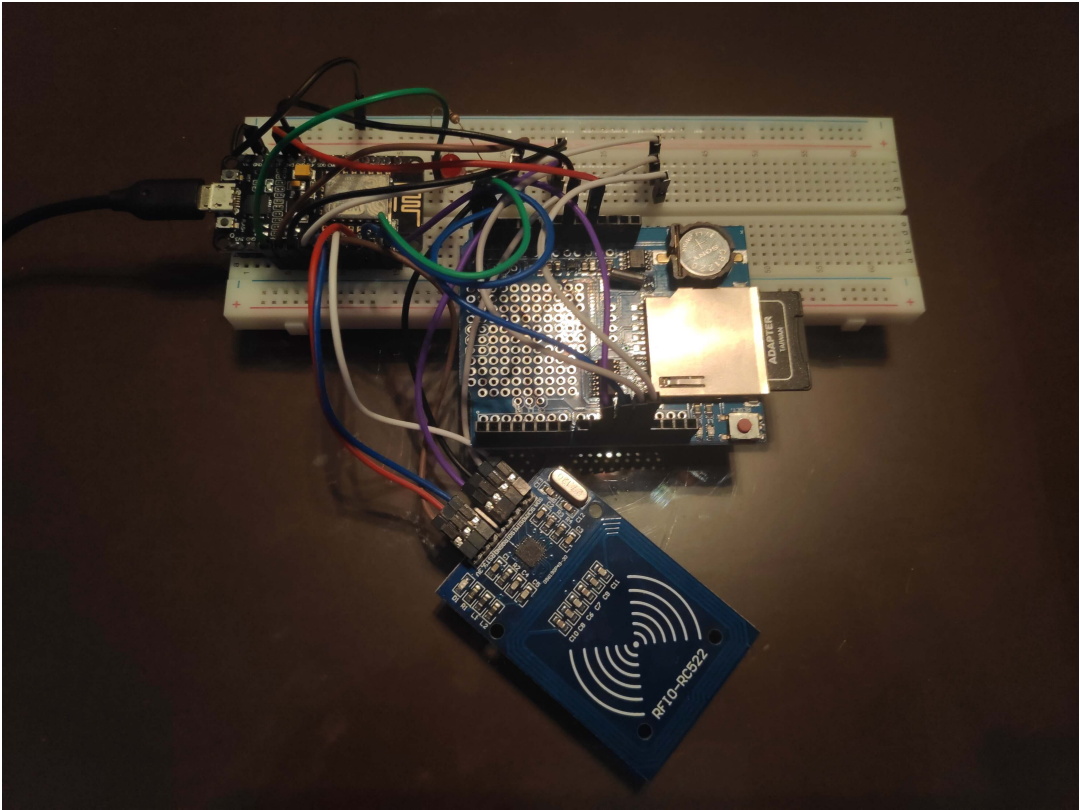
4.2 Sistema Autenticador

O sistema autenticador proposto na Seção 3.2 foi construído em forma de protótipo em uma matriz de contato, que pode ser observado na Figura 23.

O diagrama de circuito pode ser observado na Figura 24. Um *shield* de Arduino que agraga um leitor de cartão SD e Relógio em Tempo Real (RTC) foi utilizado para protótipo por motivo de fácil disponibilidade para o Autor. Um ponto importante a se destacar é que tanto o leitor de cartão SD quanto o módulo RTC presentes no *shield* de Arduino possuem faixa de tensão de alimentação entre 3,3 V a 5V. A trilha que alimenta esses circuitos integrados é a que está rotulada como “5 V” no shield, porém é alimentado diretamente pelo pino de 3,3 V do microcontrolador, uma vez que esse nível de tensão é suficiente para os circuitos integrados do leitor de cartão SD e módulo RTC.

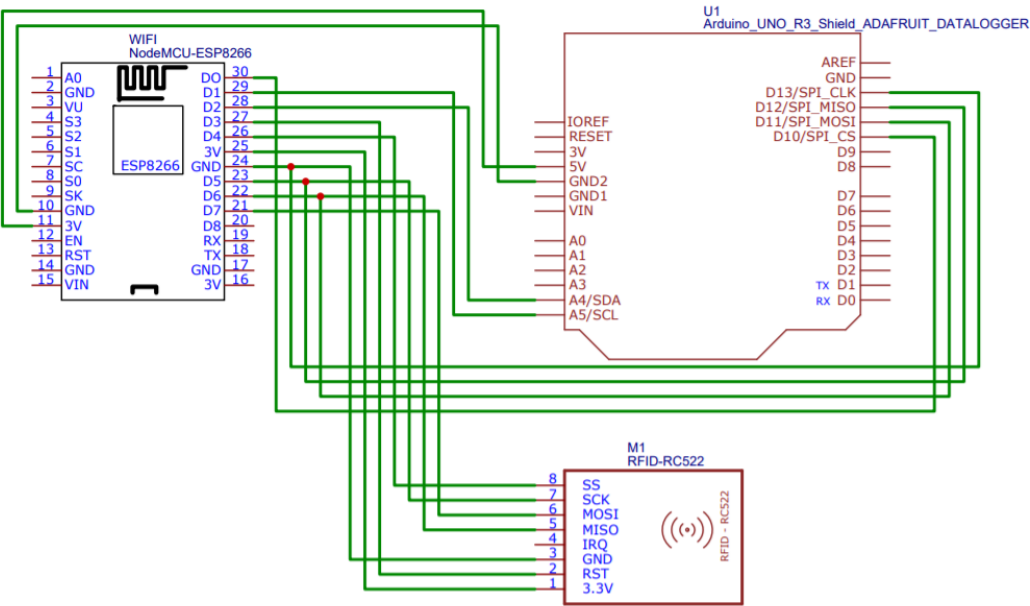
Para garantir o funcionamento dos componentes, cada módulo foi testado separadamente e todos funcionaram conforme esperado. Ao realizar o teste em conjunto, entretanto, observou-se que o módulo GPS não mais enviava a informação de posição. O autor voltou a testar o módulo GPS separadamente e notou-se que nenhuma informação era recebida pelo microcontrolador. Assumiu-se então que houve falha no componente. Após a aquisição de outro módulo GPS do mesmo modelo, desde o início não houve sucesso em recebimento de informações de localização, tanto em teste separado quanto em conjunto. Por falta de acesso a outro módulo GPS, foi decidido prosseguir o projeto sem utilizar

Figura 23 – Protótipo desenvolvido.



Fonte: Autor.

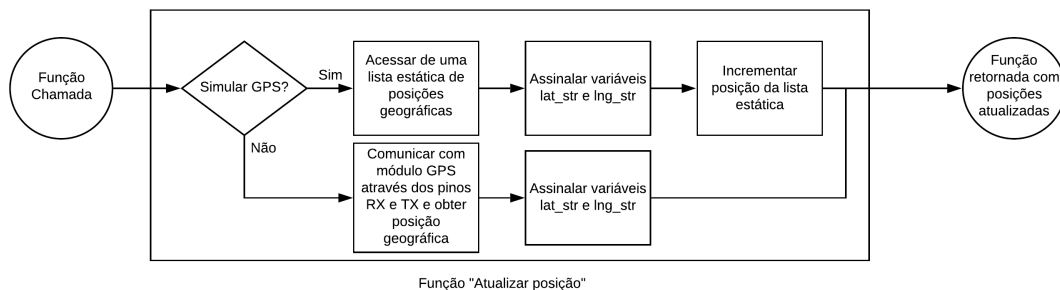
Figura 24 – Esquemático de circuito elétrico do sistema leitor.



Fonte: Autor.

esse módulo, porém simulando os dados de um GPS através de uma lógica implementada no microcontrolador, tal como ilustrado na Figura 25. Uma técnica de caixa branca foi utilizada com o objetivo de que as entradas e saídas da função de atualizar a posição geográfica permanecessem inalteradas, entretanto modificando o funcionamento interno para que a fonte de informação de latitude e longitude seja uma lista estática salva na memória de programa do microcontrolador, em vez de utilizar o módulo GPS. Uma vez que o módulo GPS não estará disponível, faz-se necessário utilizar o módulo RTC para que a informação de data e hora ainda possa ser registrada no cartão de memória mesmo quando a comunicação com a internet não estiver disponível.

Figura 25 – Fluxograma da função de atualizar posição geográfica.



Fonte: Autor.

Sabendo que a interface de desenvolvimento Arduino conta com um monitor serial que consegue se comunicar diretamente com a placa de desenvolvimento utilizada por meio de um cabo USB, foi inserido no código fonte comandos que imprimem a situação de funcionamento dos componentes. O código fonte do microcontrolador está disponível para consulta no apêndice deste documento.

Dessa forma, o sistema foi iniciado com um cartão RFID já aproximado e o resultado imprimido no monitor serial é exibido na Figura 26, na qual foi inserida uma numeração à esquerda que será referenciada na lista a seguir.

- **Linha 1:** Sinaliza que o RTC foi encontrado e está em funcionamento;
- **Linha 2:** Indica se a partida do motor foi liberada ou não quando o critério de segurança é atendido;
- **Linha 3:** É feita uma tentativa de conexão a rede Wi-Fi durante a inicialização do sistema e esta linha indica se a conexão foi feita com sucesso;

5 Conclusão

As tecnologias utilizadas no projeto foram apresentadas no Capítulo 2 para familiarizar os leitores com os conceitos e nomenclaturas utilizadas nos capítulos seguintes. Foi definido no Capítulo 3 a metodologia a ser aplicada durante o desenvolvimento do projeto a fim de garantir a reprodutibilidade do sistema. Alguns diagramas UML foram desenhados para facilitar o entendimento da interação entre os componentes.

Por fim, no Capítulo 4 foram apresentados os resultados. O site para uso do administrador apresentou comportamento esperado, uma vez que foi possível cadastrar e salvar no banco de dados as demais configurações de utilização dos sistemas leitores, usuários veiculares e tags RFID. A interação entre o servidor e componente ESP8266 não apresentou problemas, já que todas as requisições do tipo GET e POST foram executadas sem retorno de código de erro. O sistema leitor, que foi desenvolvido em caráter de prova de conceito, apresentou falhas com origens desconhecidas no módulo GPS em dois componentes físicos distintos. Dessa forma, foi necessário simular dados de localização geográfica para que o restante do sistema não fosse afetado. Os outros módulos físicos apresentaram comportamento esperado, ou seja, foi possível fazer leitura da tag RFID, ler e escrever dados no cartão de memória e resgatar informações de data e hora através do componente RTC.

A segurança do usuário motorista foi levando em consideração durante todo o desenvolvimento do projeto. Uma vez que a indisponibilidade de torque é um fator que pode colocar em risco a vida do motorista, foi implementado uma lógica que libera a partida do motor de forma muito mais rápida e sem depender de internet. Entretanto, esta liberação só será permitida para o último usuário autenticado dentro das últimas 24 horas.

Para o fim de trabalhos futuros, pretende-se estudar com mais profundidade o ponto de corte ideal entre a bateria do carro e o motor de partida do veículo. Devido a complexidade dos diversos módulos presentes no carro, tal como o sistema de alarme e os sistemas eletrônicos que controlam a partida do motor, a decisão de ponto de corte não é trivial, pois impacta diretamente o funcionamento do veículo. Além disso, pretende-se

substituir e integrar um novo módulo GPS para que dados reais sejam utilizados, avaliar os possíveis impactos de segurança da comunicação entre o sistema leitor e o servidor web e, por fim, projetar uma placa de circuito impresso. Uma vez que o sistema for integrado a um veículo, deve-se realizar testes em ambientes controlados para validar o funcionamento do sistema e garantir a segurança dos usuários.

Nota-se que a sociedade moderna necessita de soluções cada vez mais tecnológicas para trazer agilidade às tarefas do dia-a-dia. Nas empresas, o compartilhamento de veículos entre funcionários é problemático pois exige um controle de utilização. Há ainda outros setores que podem se beneficiar do sistema proposto, tais como os aplicativos de entrega à domicilio, uma vez que poderia existir uma frota de veículos especiais pertencentes à empresa e compartilhada pelos seus entregadores. Dessa forma, o sistema traria segurança e simplicidade no gerenciamento veicular da empresa.

Referências

- 99/IPSOS. *Como o brasileiro entende o transporte urbano*. 2018. Disponível em: <<https://99app.com/como-o-brasileiro-entende-o-transporte-urbano/>>. Citado na página 12.
- AIAFA. *Frota de veículos corporativos deve crescer no Brasil*. 2018. Disponível em: <<https://br.aiafa.com/frota-de-veiculos-corporativos-deve-crescer-no-brasil/>>. Citado na página 12.
- AMAZON. *O que é banco de dados relacional?* 2020. Disponível em: <<https://aws.amazon.com/pt/relational-database/>>. Citado na página 22.
- BRAIN, M.; HARRIS, T. *How GPS Receivers Work*. 2006. Disponível em: <<https://electronics.howstuffworks.com/gadgets/travel/gps.htm>>. Citado na página 19.
- CRELIER, C. Número de pessoas que trabalham em veículos cresce 29,2%. *Agencia de Noticias IBGE*, 2019. Disponível em: <<https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/26424-numero-de-pessoas-que-trabalham-em-veiculos-cresce-29-maior-alta-da-serie>>. Citado na página 12.
- DEVMEDIA. *Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER)*. 2020. Disponível em: <<https://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>>. Citado na página 22.
- ESPRESSIF. *ESP8266EX*. 2020. Disponível em: <<https://www.espressif.com/en/products/hardware/esp8266ex/overview>>. Citado na página 27.
- FOUNDATION, P. S. *Python For Beginners*. 2020. Disponível em: <<https://www.python.org/about/gettingstarted/>>. Citado na página 21.
- GARFINKEL, S.; HOLTZMAN, H. *Understanding RFID Technology*. [S.l.], 2005. Citado na página 17.
- LOMBARDI, M. et al. Time and frequency measurements using the global positioning system. *Cal Lab: The International Journal of Metrology*, v. 8, p. 26–33, 07 2001. Citado na página 18.
- MAHMOOD, A.; JAVAID, N.; RAZZAQB, S. A review of wireless communications for smart grid. *Renewable and Sustainable Energy Review*, v. 41, p. 248–260, jan 2015. Citado na página 19.
- MOZILLA. *O que é um servidor web (web server)?* 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Common_questions/o_que_e_um_web_server>. Citado na página 21.
- WANG, H. A historical review and bibliometric analysis of gps research from 19912010. *Scientometrics*, 2012. Citado na página 17.

WEINSTEIN, R. Rfid: a technical overview and its application to the enterprise. *IT Professional*, v. 7, n. 3, p. 27–33, 2005. Citado na página 17.

ZANELLA, N. B. A. et al. Internet of things for smart cities. *IEEE Internet of Things Journal*, v. 1, n. 1, p. 22–32, feb 2014. Citado na página 20.

Apêndices

APÊNDICE A – Código do microcontrolador

```

1  #include "RTCLib.h"
2  #include <ESP8266WiFi.h>
3  #include <WiFiClient.h>
4  #include <ESP8266WebServer.h>
5  #include <ESP8266HTTPClient.h>
6  #include <TinyGPS++.h>
7  #include <SoftwareSerial.h>
8  #include <SPI.h>
9  #include <MFRC522.h>
10 #include <SD.h>
11
12 //region GLOBAL_CONFIGURATION
13 bool debug = true;
14 bool simulate_gps = true;
15 #define SS_PIN 2
16 #define RST_PIN 0
17 //endregion
18
19 //region VARIABLE_INITIALIZATION
20 RTC_DS1307 rtc;
21 MFRC522 mfrc522(SS_PIN, RST_PIN);    // Create MFRC522 instance.
22 int out = 0;
23 //const char *ssid = "APT 902 2g"; //ENTER YOUR WIFI SETTINGS
24 //const char *password = "mansaolaranja";
25 String ssid;
26 String password;
27 int CS_SD = 16; // DO
28 const char *host = "http://filipe027.pythonanywhere.com/";
29 TinyGPSPlus gps;
30 SoftwareSerial ss(4, 5) ;
31
32 float latitude , longitude;
33 int year , month , day, hour , minute , second;
34
35 String date_str , time_str , lat_str , lng_str;
36
37 int pm;
38 int counter;
39 File file;
40
41 String rfid_tag;

```

```
42 String server_time;
43 String login_id = "null";
44 String signedin = "dnd";
45 unsigned long t0;
46 File login_root;
47 File location_root;
48 bool scanning_location_root = true;
49 File login_file;
50 File location_file;
51 String mac;
52 DateTime now;
53
54 //endregion
55
56 void setup() {
57
58     if(debug == false){
59         pinMode(1, OUTPUT);
60     }
61     lock_relay();
62
63     mac = WiFi.macAddress();
64     mac.replace(":", "_");
65
66     counter = 0;
67     SPI.begin();          // Initiate SPI bus
68     mfr522.PCD_Init();    // Initiate MFRC522
69     delay(1000);
70     Serial.begin(9600);
71     //ss.begin(9600);
72
73     if (! rtc.begin()){
74         log_debug("RTC não encontrado");
75     }
76     else{
77         log_debug("RTC OK");
78     }
79
80     //Serial.println("Iniciando cartao SD...");
81     int sd_tries = 0;
82     while (!SD.begin(CS_SD))
83     {
84         //Serial.println("Falha na inicializacao do SD!");
85         sd_tries +=1;
86         delay(100);
87         if(sd_tries >=10){
88             break;
89         }
90     }
```

```
91
92   if (sd_tries <10){
93       read_ssid();
94   }else{
95       ssid = "admin";
96       password = "123456";
97   }
98   long init_backup_timeout = millis();
99   rfid_tag = "";
100  while (millis() - init_backup_timeout <= 5000){
101      if ( mfrc522.PICC_IsNewCardPresent()){
102          rfid_tag = read_rfid();
103          break;
104      }else{
105          delay(1);
106      }
107  }
108  if (rfid_tag != ""){
109      if (sdcard_check_backup(rtc.now().secondstime(), rfid_tag)){
110          unlock_relay();
111          log_debug("Rele liberado por criterio de segurança");
112      }
113  }
114  else{
115      lock_relay();
116  }
117
118  connect_to_wifi();
119  if (WiFi.status() == WL_CONNECTED)
120      log_debug("Conectado a rede Wi-Fi");
121  else
122      log_debug("Falha ao conectar");
123
124
125  if( wifi_connected() ){
126      server_time = get_server_time();
127      log_debug("Hora do servidor: " + server_time);
128      //Server time format = 2020-03-07-22-59-58
129      int year = server_time.substring(0,4).toInt();
130      int month = server_time.substring(5,7).toInt();
131      int day = server_time.substring(8,10).toInt();
132      int hour = server_time.substring(11,13).toInt();
133      int minute = server_time.substring(14,16).toInt();
134      int second = server_time.substring(17,19).toInt();
135      rtc.adjust(DateTime(year, month, day, hour, minute, second));
136      String now_str = get_time_str();
137      log_debug("Hora do RTC ajustado para: " + now_str);
138  }
139
```

```
140     if (sd_tries<10){
141         login_root = SD.open("/L/");
142         location_root = SD.open("/C/");
143         login_root.rewindDirectory();
144         location_root.rewindDirectory();
145         login_file = login_root.openNextFile();
146         location_file = location_root.openNextFile();
147     }
148     t0 = millis();
149
150 }
151
152 void loop() {
153     if ( ! wifi_connected()){
154         connect_to_wifi();
155     }
156     if (millis() - t0 >= 5000){
157         if (update_gps()){
158
159             sdcard_log_location(login_id, lat_str + "_" + lng_str, get_time_str());
160             if( wifi_connected() ){
161                 log_debug("Requisicao de localização HTTP enviada com dados: "
162                     "id=" +login_id + "&pos=" + lat_str + "_" + lng_str + "&time="
163                     + get_time_str() + "&mac=" + mac );
164                 post_location(lat_str + "_" + lng_str);
165             }
166             else{
167                 log_debug("Dados salvos no cartão SD para envio posterior: "
168                     "id=" +login_id + "&pos=" + lat_str + "_" + lng_str + "&time="
169                     + get_time_str()+ "&mac=" + mac);
170                 sdcard_log_location_pending(login_id, lat_str
171                     + "_" + lng_str, get_time_str());
172             }
173         }
174         t0 = millis();
175     }
176
177     if (signedin == "alw"){
178         if (scanning_location_root){
179             if (location_file.available())
180             {
181                 if (wifi_connected()){
182                     sdcard_upload_location_pending();
183                 }
184             }
185             else{
186                 String name_file = location_file.name();
187                 location_file.close();
188                 SD.remove("/C/" + name_file);
```

```
189         location_file = location_root.openNextFile();
190         if ( !location_file ){
191             location_file.close();
192             scanning_location_root = false;
193         }
194     }
195 }
196 else{
197     rewind_locdir_with_timeout();
198 }
199 }
200 else{
201     if ( ! mfrc522.PICC_IsNewCardPresent())
202     {
203         return;
204     }
205     else{
206         rfid_tag = read_rfid();
207         log_debug("Leitura do cartão RFID" + rfid_tag);
208         if( wifi_connected() and rfid_tag != "" ){
209             String result = request(rfid_tag);
210             signedin = result.substring(0,3);
211             login_id = result.substring(4);
212             if (signedin == "alw") {
213                 log_debug("Autenticacao liberada pelo servidor");
214                 sdcard_store_backup(rtc.now().secondstime(), rfid_tag);
215                 unlock_relay();
216                 log_debug("Rele de partida liberado");
217             }
218             sdcard_log_login(login_id, signedin, get_time_str());
219         }
220     }
221 }
222 }
223
224
225 }
226
227 //region GPS_UPDATE
228 String locations[] = {
229     "-07.142862_-034.850775",
230     "-07.142106_-034.850902",
231     "-07.141590_-034.849069",
232     "-07.138721_-034.849917",
233     "-07.138009_-034.847697",
234     "-07.137141_-034.845460",
235     "-07.135592_-034.845722",
236     "-07.134171,-034.846300",
237     "-07.132237,-034.844527",
```



```
238 "-07.131625,-034.844631",
239 "-07.130739,-034.842246",};
240 int i_loc = 0, loc_len=10;
241 bool update_gps() {
242     if (simulate_gps){
243         if(i_loc <= loc_len){
244             lat_str = locations[i_loc].substring(0,10);
245             lng_str = locations[i_loc].substring(11);
246             i_loc++;
247             return true;
248         }else{
249             return false;
250         }
251     }
252     else {
253         if (Serial.available() > 0){
254             while (Serial.available() > 0){
255                 if ( gps.encode( Serial.read() )){
256                     if (gps.location.isValid())
257                     {
258                         latitude = gps.location.lat();
259                         lat_str = String(latitude , 6);
260                         longitude = gps.location.lng();
261                         lng_str = String(longitude , 6);
262                     }
263
264                     if (gps.date.isValid())
265                     {
266                         day = gps.date.day();
267                         month = gps.date.month();
268                         year = gps.date.year();
269                         hour = gps.time.hour();
270                         minute = gps.time.minute();
271                         second = gps.time.second();
272                     }
273                 }
274             }
275             return true;
276         }
277         else{
278             return false;
279         }
280     }
281 }
282 //endregion
283
284 //region HTTP_REQUESTS
285 String request(String tag){
286     HTTPClient http;    //Declare object of class HTTPClient
```

```
287
288
289 //Post Data
290 String postData = "tag=" + tag + "&location=1353&mac=" + mac;
291
292 http.begin("http://filipe027.pythonanywhere.com/post/");
293 http.addHeader("Content-Type", "application/x-www-form-urlencoded");
294
295 int httpCode = http.POST(postData); //Send the request
296 String payload = http.getString(); //Get the response payload
297
298
299
300 http.end(); //Close connection
301 return payload;
302 }
303
304 String post_location(String pos){
305     HTTPClient http; //Declare object of class HTTPClient
306
307
308 //Post Data
309 String postData = "id=" + login_id + "&pos=" + pos
310     + "&time=" + get_time_str() + "&mac=" + mac;
311
312 http.begin("http://filipe027.pythonanywhere.com/post-loc/");
313 http.addHeader("Content-Type", "application/x-www-form-urlencoded");
314
315 int httpCode = http.POST(postData); //Send the request
316 String payload = http.getString(); //Get the response payload
317
318 http.end(); //Close connection
319 log_debug(payload);
320 return payload;
321 }
322
323 String post_to_url(String path, String postData){
324     HTTPClient http;
325     http.begin("http://filipe027.pythonanywhere.com" + path);
326     http.addHeader("Content-Type", "application/x-www-form-urlencoded");
327     postData.trim();
328     log_debug(postData);
329     int httpCode = http.POST(postData); //Send the request
330     String payload = http.getString(); //Get the response payload
331
332     http.end(); //Close connection
333     return payload;
334 }
335
```

```
336 String get_server_time(){
337     HTTPClient http;
338
339     http.begin("http://filipe027.pythonanywhere.com/time/"
340         + WiFi.macAddress() + "/");
341     int httpCode = http.GET();
342     return http.getString();
343     http.end();
344 }
345 //endregion
346
347 //region UTILITY_FUNCTIONS
348 void log_debug(String message){
349     if (debug) {
350         Serial.println(message);
351     }
352 }
353
354 void unlock_relay(){
355     if(debug == false){
356         digitalWrite(1, HIGH);
357     }
358 }
359
360 void lock_relay(){
361     if(debug == false){
362         digitalWrite(1, LOW);
363     }
364 }
365
366 bool wifi_connected(){
367     return (WiFi.status() == WL_CONNECTED);
368 }
369 }
370
371 void connect_to_wifi(){
372     WiFi.mode(WIFI_OFF);
373     delay(1000);
374     WiFi.mode(WIFI_STA);
375
376     WiFi.begin(ssid.c_str(), password.c_str());
377     // Wait for connection
378     int timeout = 0;
379     while (WiFi.status() != WL_CONNECTED) {
380         delay(500);
381         timeout +=1;
382         if(timeout == 25){
383             break;
384         }
385     }
```

```
385     }
386 }
387
388 String read_rfid(){
389     if ( ! mfrc522.PICC_ReadCardSerial()){
390         return "";
391     }
392     String content= "";
393     byte letter;
394     for (byte i = 0; i < mfrc522.uid.size; i++)
395     {
396         content.concat(String(mfrc522.uid.uidByte[i]
397             < 0x10 ? " 0" : " "));
398         content.concat(String(mfrc522.uid.uidByte[i], HEX));
399     }
400     content.toUpperCase();
401     content = content.substring(1);
402     return content;
403 }
404
405 File backup_file;
406 void sdcard_store_backup(long seconds, String UID){
407     backup_file = SD.open("backup.txt", FILE_WRITE);
408     backup_file.seek(0);
409     backup_file.println(String(seconds));
410     backup_file.println(UID);
411     backup_file.close();
412 }
413
414 bool sdcard_check_backup(long seconds, String UID){
415     backup_file = SD.open("backup.txt", FILE_READ);
416     backup_file.seek(0);
417     long backup_seconds = backup_file.readStringUntil('\n').toInt();
418     String sd_uid = backup_file.readStringUntil('\n');
419     sd_uid.trim();
420
421     //86400 seconds = 24h
422     if( seconds - backup_seconds <= 86400 and UID == sd_uid){
423         log_debug("unlocked by backup");
424         return true;
425     }
426     else{
427         log_debug("backup denied");
428         return false;
429     }
430 }
431
432 long rewind_timer = 0;
433 const int REWIND_TIMEOUT = 60000;
```

```
434 void rewind_locdir_with_timeout(){
435     if (millis() - rewind_timer >= REWIND_TIMEOUT){
436         location_root.rewindDirectory();
437         location_file = location_root.openNextFile();
438         if (location_file){
439             scanning_location_root = true;
440         }
441         rewind_timer = millis();
442     }
443 }
444
445
446 void read_ssid(){
447     if (SD.exists("ssid.txt")){
448         File ssid_file = SD.open("ssid.txt", FILE_READ);
449         ssid = ssid_file.readStringUntil('\n');
450         password = ssid_file.readStringUntil('\n');
451         ssid.trim();
452         password.trim();
453     }
454 }
455
456 String syear, smonth, sday, shour, sminute, ssecond;
457 String get_time_str(){
458     now = rtc.now();
459     syear = String(now.year());
460     if(now.month() < 10){
461         smonth = String("0") + String(now.month());
462     }
463     else{
464         smonth = String(now.month());
465     }
466     if(now.day() < 10){
467         sday = String("0") + String(now.day());
468     }
469     else{
470         sday = String(now.day());
471     }
472     if(now.hour() < 10){
473         shour = String("0") + String(now.hour());
474     }
475     else{
476         shour = String(now.hour());
477     }
478     if(now.minute() < 10){
479         sminute = String("0") + String(now.minute());
480     }
481     else{
482         sminute = String(now.minute());
```

```
483     }
484     if(now.second() < 10){
485         ssecond = String("0") + String(now.second());
486     }
487     else{
488         ssecond = String(now.second());
489     }
490
491     return syear + "-" + smonth + "-" + sday
492         + "-" + shour + "-" + sminute + "-" + ssecond ;
493
494 }
495 //endregion
496
497 //region SD_FUNCTIONS
498 void grava_cartao_SD(String data)
499 {
500     delay(10);
501     //Abre arquivo no SD para gravacao
502     file = SD.open("sd04.txt", FILE_WRITE);
503     //Le as informacoes de data e hora
504     file.println(data);
505     file.close();
506 }
507
508 // Informação do data logger, arquivo não será apagado
509 void sdcard_log_location(String login_id, String position, String time){
510     File file = SD.open("LOCLOG.txt", FILE_WRITE);
511     file.println("id=" + login_id + "&pos="
512         + position + "&time=" + time + "&mac=" + mac);
513     file.close();
514 }
515
516 // Informação do data logger, arquivo não será apagado
517 void sdcard_log_login(String server_id, String auth, String time){
518     File file = SD.open("LINLOG.txt", FILE_WRITE);
519     file.println("server_id=" + server_id
520         + "&auth=" + auth + "&time=" + time);
521     file.close();
522 }
523
524 void sdcard_log_location_pending(String login_id, String position, String time){
525     File file = SD.open("/C/" + login_id + ".txt", FILE_WRITE);
526     file.println("id=" + login_id + "&pos=" + position
527         + "&time=" + time + "&mac=" + mac);
528     file.close();
529 }
530
531 void sdcard_upload_location_pending(){
```

```
532     String data = location_file.readStringUntil('\n');
533     log_debug("Localização pendente enviada para o servidor: " + data);
534     post_to_url("/post-loc/", data);
535 }
536 //endregion
```