DETECTOR DE VELOCIDADE DE CONDOMÍNIO UTILIZANDO TECNOLOGIA INFRAVERMELHO DE BAIXO CUSTO

Luiz Guilherme Tordin[1] e Giovana A. J. Almeida[1]
Fábio Andrijauskas[2]
Universidade São Francisco
lgtordin@yahoo.com.br

[1]Aluno do Curso de Engenharia da Computação, Universidade São Francisco; Campus - Itatiba
[2]Professor Orientador Fábio Andrijauskas, Curso de Engenharia da Computação, Universidade São Francisco; Campus-Itatiba.

Resumo: A imprudência de alguns motoristas no trânsito é uma realidade e infelizmente há vítimas, muitas vezes fatais, dessas imprudências. Nos condomínios não é diferente, assim como no trânsito, ocorrem muitas imprudências. Os detectores de velocidade são dispositivos que surgiram como forma de diminuir as imprudências e conscientizar os motoristas, porém esses detectores têm um custo muito elevado e muitos condomínios não têm esse sistema e acabam optando por sistemas mais baratos, tais como lombadas de borracha, placas, entre outros. Devido a esse alto custo, será proposto o desenvolvimento de um protótipo de detector de velocidades para ser usado em condomínios, de modo a permitir não só o controle, mas também a gestão da velocidade e dos infratores, através de um aplicativo integrado com banco de dados. Esse trabalho irá detalhar esse detector de velocidade, explicado desde o armazenamento de informações de velocidade dos veículos no banco de dados, até a integração entre hardware e software. Também será apresentado um diferencial no qual foi inserido ao detector de velocidade um método de detecção de padrão de velocidade de forma individual para cada veículo para permitir uma gestão além de apenas o controle de velocidade. Esse projeto tem por objetivo permitir a detecção de velocidade e a geração de relatórios dessas velocidades com foco em se construir um detector de baixo custo e acessível a quaisquer condomínios. Por fim, este trabalho irá comparar com um projeto real em termos de custo para avalizar os resultados atingidos e fazer testes no protótipo a fim de se verificar sua eficiência.

Palavras-chave: Detector de velocidade, Arduíno, Java, MySQL.

1 – Introdução

Existem diversos dispositivos com a função de se controlar velocidade no trânsito, visando garantir que os motoristas tenham consciência do limite de velocidade pré-estabelecido e a identificação e punição de motoristas infratores visando acima de tudo garantir a segurança dos condutores e pedestres (Ubisse, 2017). Esses dispositivos podem ser desde meios baratos, tais como lombadas de borracha ou placas de limite de velocidade, funcionando mais como meios de conscientização, mas não sendo eficazes no controle da velocidade, até meios mais caros como a fiscalização eletrônica. A fiscalização eletrônica de velocidades é um meio muito eficiente para se controlar a velocidade dos veículos, devido a ser um sistema que funcionam 24 horas e permite a identificação instantânea do infrator devido ao registro fotográfico, auxiliando e muito os agentes de trânsito. Por ser um meio autêntico, é improvável sua falsificação, tornando um meio seguro de se fazer o controle de velocidades (Yamada, 2005).

Os dispositivos para controlar velocidade podem ser manuais, necessitando de um operador para controla-lo, ou automáticos. Nos dispositivos manuais, podem ser aplicadas à tecnologia RADAR (*Radio Detection and Ranging),* emitindo ondas de rádio que, ao colidir com o alvo, são refletidas de volta ao aparelho, permitindo assim calcular a velocidade em

função do tempo de ida e volta das ondas de rádio. Outra tecnologia é a LIDAR (*Light Detection and Ranging),* que consiste em um princípio similar ao radar, porém com ondas infravermelhas ao invés de ondas de rádio (Ubisse, 2017). A fiscalização eletrônica pode ser feita por radares fixos permitindo o registro de fotos, hora e data da infração e com as informações armazenadas em uma central fixa ao poste (Yamada, 2005). Também podem ser feitas com radares estáticos, com registros feitos por ondas de rádio, com ultra som ou com laser. São montados em um tripé, dependendo de um operador para monitorá-lo (Yamada, 2005). A lombada eletrônica e a bandeira eletrônica também são exemplos de fiscalização eletrônica, com o diferencial de serem aparelhos robustos e visíveis (Yamada, 2005). Apesar de serem meios de fiscalização eficazes, o grande problema desse tipo de fiscalização é seu alto custo devido a todo o aparato tecnológico e/ou a necessidade de um operador especializado para acompanhamento somado ao fato de que as empresas que alugam esse tipo de equipamento para agencias de trânsito incluem custos de manutenção que devem ser feitas periodicamente. No capítulo de resultados e discussões, esse custo será mais detalhado.

Com o alto custo dos radares de velocidade, o objetivo deste trabalho é desenvolver um protótipo de detector de velocidades para condomínios, de baixo custo. Para isso, o protótipo de detector de velocidades para condomínios foi desenvolvido em três partes: a primeira contendo hardware no qual irá coletar os dados dos veículos e fazer o cálculo da velocidade e segunda parte é o software no qual funcionará com interface de comunicação entre usuários e sistema, receberá as informações vindas do hardware e será responsável por fazer toda a comunicação backgroud com o banco de dados e a terceira parte é o banco de dados no qual irá armazenar todas as informações de cadastros dos veículos, moradores e registros do hardware de moto a permitir a emissão de relatórios, calcular padrões de velocidade de modo a permitir que o radar opere de maneira inteligente.

Para este trabalho, levaram-se em consideração alguns trabalhos já realizados sobre o tema, de modo a se analisar os pontos fortes e pontos fracos de cada um. Os trabalhos pesquisados serão mais bem discutidos no capítulo Revisão Bibliográfica. Como diferencial e ponto positivo com relação aos trabalhos estudados, será desenvolvido um aplicativo para funcionar em conjunto com o hardware (medidor de velocidades) a fim de que seja possível acessar relatórios de infração, utilizando para isso um banco de dados para armazenar informações e um aplicativo acessível para os usuários poderem emitir seus relatórios.

Este trabalho será dividido em cinco capítulos: Revisão Bibliográfica, iniciando com a apresentação dos trabalhos sobre o tema, seguido por uma fundamentação teórica a respeito dos principais conceitos utilizados neste trabalho, tais como cinemática de um ponto material, linguagens de programação que servirão para fazer o aplicativo e introdução à banco de dados, Metodologia, destacando os componentes de hardware e programas para codificação utilizadas na construção do software, Resultados e Discussões, em que serão apresentados os métodos de comunicação entre hardware, software e banco de dados, métodos para tornar o radar inteligente, resultados obtidos, testes realizados para a validação do protótipo e comparação de custo para analisar se o objetivo foi atingido e Conclusão, no qual será feito uma síntese dos resultados obtidos e sugestões de trabalhos futuros.

2 - Revisão Bibliográfica

Este capítulo irá apresentar os trabalhos já existentes sobre o tema que foram usados como ponto de partida para o início deste trabalho, irá também introduzir os conceitos de cinemática de um ponto material, Java e banco de dados, entendimento dos quais fazem parte deste trabalho. Iniciando com os trabalhos sobre o tema, a seguir serão apresentados três trabalhos dos quais serão analisados e discutidos através do levantamento de seus pontos fortes e/ou fracos, analise da qual será aplicada ao protótipo deste trabalho.

O primeiro trabalho analisado é uma monografia de licenciatura em engenharia de eletrônica e de telecomunicações, intitulada 'Desenvolvimento de um sistema automático de detecção de excesso de velocidade na via pública' (Ubisse, 2017). Esse trabalho consiste em utilizar dois sensores de presença LDR (*Light Dependent Resistor*) para detectar a veículo, posicionados a uma certa distância de modo com que a diferença de tempo de acionamento dos dois sensores e a distância de posicionamento entre eles sejam usados para calcular a velocidade do veículo. Caso a velocidade seja superior ao permitido, uma câmera JPEG 2M captura a imagem e as informações são armazenadas em um módulo Ethernet Shield. O ponto negativo deste trabalho é relacionado ao seu módulo de armazenamento, no qual as imagens capturadas pela câmera são salvas em uma memória externa, inserida no módulo Ethernet Shield, fazendo com que o armazenamento seja dependente desse cartão de memória. Juntando-se a falta de algum banco de dados de cadastro de veículos ou metodologias de buscas em bancos de dados de órgãos competentes, os relatórios ficam restritos somente a um banco de dados de imagens, fazendo com que relatórios para aplicações de multas tenham que ser feitos manualmente ou por algum outro software externo. O autor (Ubisse, 2017), sugere para trabalhos futuros utilizar um chip para capturar informações dos veículos, utilizando para isso a tecnologia RFID (*Radio Frequency IDentification),* eliminando-se assim a necessidade da câmera para captura de imagens. Neste trabalho, foram feitas algumas alterações durante sua concepção, que são a alteração do sensor infravermelho para a detecção de velocidades pelo sensor LDR, devido à demora de resposta do sensor infravermelho em emitir a informação, e o microcontrolador Arduino Mega foi substituído pelo Arduíno Uno. Porém, há um outro trabalho que conseguiu utilizar o sensor infravermelho para medir a velocidade do veículo, fazendo isso através de um projeto de semáforo inteligente, que será mostrado a seguir.

O segundo trabalho trata-se de um artigo realizado na Universidade de Feira de Santana, Bahia, intitulado "Desenvolvimento de um Semáforo Inteligente Utilizando Arduino e Sensores Infravermelhos" (Santos, 2019), consiste em um protótipo de um sistema de semáforos com LED que também possui controle de velocidade, no qual a velocidade é capturada quando o semáforo estiver no vermelho. A velocidade é calculada através de dois sensores infravermelho de 5V, em função da distância e tempo de passagem do veículo. Os dados são exibidos em um Display LCD 16x2. Esse projeto, apesar de funcional não possui ferramentas de armazenamento de informações nem identificação dos veículos, não sendo possível gerar nenhum tipo de relatório. Este trabalho, apesar de apresentado no Scielo, não está totalmente finalizado, com isso, apenas será em contraponto ao primeiro trabalho no qual apresenta sensores infravermelhos de 5V com resultado satisfatório. Diferentemente dos dois projetos anteriores, o próximo trabalho faz o controle de velocidades através de RFID.

Escrito através de uma monografia para certificação em engenharia da Computação, intitulada de 'Controle de velocidade em longos trechos por RFID' (Savi, 2011), o próximo projeto apresentado usa sensores RFID, com leitor da marca *Phgidget* , para se calcular a velocidade e fazer a identificação dos veículos. A velocidade é calculada em função da distância do posicionamento dos leitores RFID e do tempo que o veículo leva para passar por esses sensores. Para isso há etiquetas RFID nos veículos que identificam a passagem pelos leitores RFID fazendo também a identificação. Esse trabalho possui um software desenvolvido em linguagem JAVA com o NetBeans IDE 7.0.1 para mostrar as informações de velocidade, tempo de travessia, tempo de passagem pelas etiquetas e também a permissão de usuários em configurar a velocidade mínima e a distância entre as etiquetas. O autor (Savi, 2011) sugere para trabalhos futuros fazer uma integração do monitoramento de velocidades com fotos do veículo infrator, fazer relatórios em tempo real e análise de licenciamento do veículo. Para este trabalho, será levado em consideração o ponto negativo destacado pelo autor relacionado à falta de relatórios em tempo real, pois, apesar de possuir um software e etiquetas RFID nos veículos, esses recursos poderiam ter sido melhor explorados com a execução de um cadastro de veículos através da criação um banco de dados para armazenamento dos relatórios emitidos. Outro ponto

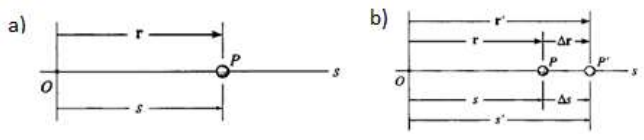
negativo deste trabalho foi seu alto custo, considerado em U$ 127,54 dólares, o que resulta algo em torno de R$ 620,00 aproximadamente (considerando cotação do dólar de 14/03/2020). Como ponto positivo vale destacar o emprego dos sensores RFID não só na identificação dos veículos, mas também no cálculo de velocidades, eliminando assim a necessidade de se ter outro sensor para cálculo de velocidades.

Em muitos condomínios há a presença de um leitor RFID na portaria para a identificação dos moradores. Utilizando a sugestão de (Ubisse, 2017) e aproveitando a existência de leitores RFID nos condomínios, nosso trabalho irá implementar essa tecnologia para a identificação dos veículos, eliminando assim a necessidade de se gastar com quaisquer outros meios, o que irá elevar o custo do protótipo de detector de velocidades. Para a velocidade, (Savi, 2011) utilzia também os leitores RFID para detector a velocidade, porém isso ocasionaria a necessidade de se comprar um segundo leitor para ser instalado na rua, pois apenas um leitor seria insuficiente para a detecção da velocidade. Comprar mais um leitor RFID deixaria o projeto muito caro, pois são dispositivos caros. Pensando no custo, para o detector de velocidades, será seguido o modelo de trabalho de (Santos, 2019), no qual foi obtido sucesso em se detectar a velocidade. Como dispositivo integrador do RFID e dos sensores infravermelho, tem-se as placas Arduino UNO e Arduino MEGA usadas nos trabalhos analisados, no qual o Arduino UNO apresenta menor custo, sendo R$ 64,90 (Electro, s.d.) contra R$ 99,90 (Electro, s.d.) da placa Arduino Mega 2560 R3. Há uma outra alternativa mais barata que e a placa Arduino nano, com custo aproximado de R$ 38,00 (Electro, s.d.). Analisando os três trabalhos, um ponto fraco em comum é a falta da existência de um banco de dados de armazenamento das informações, no qual será possível também a emissão de relatórios. Com isso, como diferencial, será feito um banco de dados e um aplicativo, de modo a se gerar relatórios para serem exibidos de forma intuitiva aos usuários do protótipo. Para esse fim, será usado um banco de dados relacional, sendo acessível através de uma interface de usuário. Para programação do software e do banco de dados, serão utilizadas as linguagens de programação JAVA e SQL, sendo escolhido por motivos de conhecimento prévio das ferramentas. As IDEs para a programação serão melhor discutidas no capítulo Metodologia e Resultados e Discussões.

Na utilização dos sensores infravermelhos para se detectar a velocidade, serão necessários dois sensores posicionados a uma certa distância entre eles. O tempo de detecção de um sensor para o outro em função da distância servirá para cálculo da velocidade. Esse cálculo é feito seguindo a cinemática de um ponto material. Considerando-se um ponto material, que se desloca ao longo de um trecho reto, que neste trabalho poderá ser verificado por um carrinho que se desloca por uma rua (trecho reto), é possível verificar que, a cada instante, esse carrinho possui uma velocidade, aceleração e uma posição. A posição é definida, em uma trajetória reta, pelo ponto em que o carrinho se encontra (Hibeler, 2005).

Com a saída da inércia do carrinho P sobre a trajetória retilínea em direção a uma nova posição P', pode-se entender o conceito de deslocamento, que é definido como a mudança de sua posição (Hibeler, 2005). A Figura 1 mostra a posição 'r' vetorial do carrinho 'P', a uma distância 's' de sua origem 'O' e o carrinho 'P' se deslocando para 'P'' provocando uma mudança em sua posição inicial 's', demonstrada por 'Δs' e em sua posição vetorial 'r', demonstrada por 'Δr'.

Figura 1: a) Posição do carrinho, b) Deslocamento do carrinho



Fonte: (Hibeler, 2005).

Todo deslocamento ‘Δs’ ocorre um intervalo de tempo ‘Δt’. A relação entre essas duas grandezas é a velocidade média, que pode ser calculada pela Equação (1) (Hibeler, 2005).

$$Vméd = \frac{\Delta s}{\Delta t} \quad (1)$$

O cálculo da velocidade é a principal variável para o detector de velocidades, que neste trabalho será feita por hardware, porém essa informação deverá ser armazenada e o banco de dados é a melhor maneira de se salvar informações com segurança.

Para a criação do software deste trabalho, será necessário primeiro desenvolver o banco de dados. Um banco de dados é um meio no qual podem-se armazenar dados, podendo-se dar como exemplo simples de aplicação a agenda de um telefone celular, onde pode-se guardar nome, telefone, endereço, etc. ou seja, várias informações que podem ser acessadas, alteradas, inseridas e excluídas a qualquer momento. Os bancos de dados são constituídos de três elementos: campos, o qual é o espaço que se destina a inserção de dados, registros, que é formado por um ou vários campos indicando um conjunto de informações de determinado objeto e tabelas, no qual contêm vários registros formando assim o banco de dados (Ferrari, 2007). Na criação de tabelas de banco de dados, cada registro deve ser único, para que exista diferenciação dos campos e registros nas tabelas, evitando-se redundâncias. Cada registro deve ser inserido de forma cronológica à inserção de dados e respeitando-se o tamanho e tipo de caracteres presentes nos campos (Ferrari, 2007). Em um banco de dados existem tabelas diferentes que podem se relacionar entre si. Esses relacionamentos são feitos través de chave externas primárias e podem ser de vários tipos: Um para um, quando os registros na tabela mãe correspondem a um registro na tabela filha, um para muitos, quando um registro na tabela mãe correspondem a muitos registros na tabela filha, muitos para um, quando muitos registros na tabela mãe correspondem a um registro na tabela filha e muitos para muitos quando muitos registros na tabela mãe correspondem a muito registros na tabela filha (Ferrari, 2007).

O banco de dados em SQL possui uma linguagem de programação diferenciada devido a ser unicamente utilizada na gestão de dados do banco em tabelas, no qual deve-se informar ao computador quais dados devem ser manipulados, independente do software ou hardware utilizados. A linguagem SQL baseia-se nas *views*, ou seja, em tabelas virtuais, que podem ser criadas através de *queries*, ou seja, perguntas para pesquisas de dados no SQL do qual se deseja obter. As queries originam views novas com os dados desejados, ou podem ser trabalhados para execução de cálculos, criação de novas tabelas, etc. (Ferrari, 2007). Para acesso ao banco de dados, é necessária uma interface para impedir que usuários possam alterar ou ter acesso ao banco de dados. A interface para ser usado nas consultas SQL será programada em Java.

Java é uma linguagem de programação de alto nível orientada a objetos, muito similar ao C++, com a finalidade de criar desde dispositivos para desktop até softwares robustos para aplicações comerciais (Sobral, 2008). O JDK (*Java Development Kit*) é um conjunto de ferramentas e programas gratuitos que servem para compilar, interpretar e utilizar dados e programas em linguagem JAVA. Para utilizar o JDK, deve-se primeiramente escrever o código em linguagem JAVA em uma IDE, que no caso deste trabalho será a NetBeans 8.0, e em seguida compilar esse código usando o compilador javac e executá-lo utilizando o interpretador java. Tanto o compilador, quanto o interpretador já estão embutidos dentro da IDE (Sobral, 2008).

A estrutura de desenvolvimento das aplicações JAVA é sempre feita através das classes, no qual são inseridos os atributos, os métodos e o corpo principal do programa. Um dos métodos importantes, presentes nas aplicações java, é o ‘main ()’. Esse método é solicitado quando o programa é compilado. Ao se inserir o método main, deve-se também declarar o objeto do array de string, através da variável ‘args’ (Sobral, 2008). Na programação em java, os atributos e métodos são contidos dentro das classes. As classes também são responsáveis por estruturar os objetos da programação, que por sua vez são as instâncias das classes responsáveis pela permissão de manipulação dos métodos e atributos de um objeto. Os pacotes contêm as classes, funcionando como um container, fornecendo ao compilador métodos de acesso as classes

necessárias para a compilação do código (Sobral, 2008). Um ponto importante da programação em java é o conceito de herança, que usa parte de uma outra classe (herda atributos e métodos) para abreviação da linguagem de programação. Para que uma classe herde uma outra classe, deve-se usar a palavra 'extends' (Sobral, 2008).

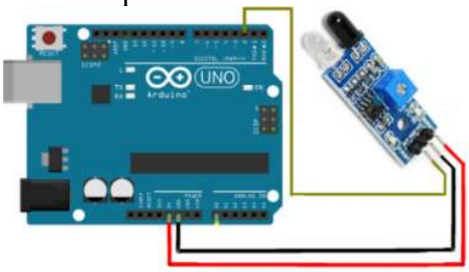
3 – Metodologia

A metodologia para este trabalho iniciou-se com a pesquisa dos trabalhos relacionados ao tema com a análise de três trabalhos já discutidas no capítulo Revisão Bibliogrpafica através do levantamento de ideias funcionais para o protótipo através do levantamento de componentes de hardware que serão descritos neste capítulo. Com os componentes de hardware levantados, precisou ser testado também se funcionamento através de uma simulação prévia para assim iniciar a montagem do protótipo. Com relação ao software, foi usado conhecimento prévio das ferramentas de programação em Java e SQL para definição dos componentes. Hardware e software serão trabalhados de maneira paralela até a integração final que será discutida no capítulo Resultados e Discussão. Este capítulo irá mostrar então os dispositivos de hardware utilizados neste trabalho em função do que foi levantado e comentado no capitulo Revisão Bibliográfica com uma breve teoria a respeito de cada um e também irá indicar as IDEs utilizadas para os softwares destacando com detalhes as versões utilizadas.

3.2 – Hardware

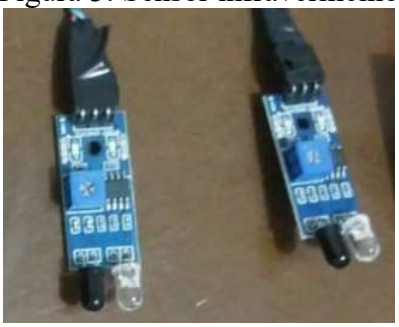
Para a detecção da velocidade será usado o sensor infravermelho reflexivo. Esse sensor é composto por dois LEDs infravermelhos, um responsável por emitir luz e o outro LED por receber luz infravermelha. Para que ocorra a recepção de luz, é necessário a presença de algum objeto que reflita a luz infravermelha emitida. O sensor tende a perder eficiência quanto maior for a distância pois assim torna-se mais difícil a recepção da luz infravermelha devido à intensidade dessa luz ser menor. O sensor possui três pinos, um de alimentação com 3,3 ou 5V, um para aterramento e um para enviar as leituras, digitalmente, para outro dispositivo e também possui um resistor, chamado de *trimpot*, que controla a quantidade de luz absorvida pelos LEDs (Athos Electronics, s.d.). A Figura 2 e a Figura 3 mostram o sensor infravermelho reflexivo e sua conexão à placa Arduíno UNO e o sensor infravermelho utilizado neste trabalho.

Figura 2: Sensor infravermelho conectado à placa Arduíno Uno.



Fonte: (Athos Electronics, s.d.)

Figura 3: Sensor infravermelho



Fonte: Própria.

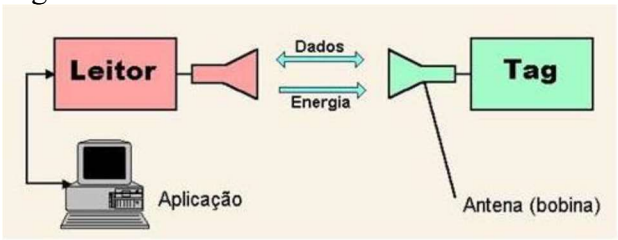
Para mostrar a velocidade e o ID do veículo de modo visível, será usado o Módulo LDC (Liquid Crystal Display 16x2) que é um componente que apresenta caracteres alfanuméricos e que permite a visualização de informações através de um display de capacidade variável de 8 a 40 caracteres por linha e de 1 a 4 linhas. Para conectar com o Arduino, o LCD possui de 14 a 16 pinos dependendo de possuir ou não o blacklight, ou luz de fundo, para permitir que a leitura possa ser feita em ambientes com pouca ou sem luz (Puhlmann, 2015). A Figura 4mostra o Display LCD com capacidade 16x2.

Figura 4: LCD 16x2.

Fonte: Própria.

Na identificação dos veículos será usado os sistemas de identificação RFID (Radio Frequency Identification) no qual cada veículo receberá um TAG de identificação única. Os RFIDs são sistemas automáticos de identificação que se baseiam na eletrônica para armazenar informações e transportá-las através de ondas de rádio. São compostos por três componentes principais, sendo eles a TAGs, os leitores e os computadores (Soares, 2018). A Figura 5 mostra, esquematicamente, o funcionamento de um sistema de identificação RFID.

Figura 5: Funcionamento de um sistema RFID.



Fonte: (Soares, 2018)

As TAGs, conhecidas como etiquetas RFID são compostas por um chip, que armazena os dados, e uma resistência, que atua como uma antena. podendo funcionar de forma passiva, não possuindo fonte de energia de alimentação, necessitando de comunicação com o leitor para poder transmitir informação ou pode funcionar de forma ativa, não precisando estar conectado a uma fonte de alimentação, devido à presença de fonte própria de energia. (Soares, 2018). Os TAGs têm seus dados capturados pelos leitores RFID, contendo mecanismos de segurança, gestão e controle dos TAGs para a efetuação da captura dos dados. Os leitores RFID transmitem energia em rádio frequência através de uma ou várias antenas, sendo capturadas pela etiqueta que a converte em energia elétrica por meio de indução. (Soares, 2018). A Figura 6 mostra o TAG RFID usado neste trabalho.

Figura 6: TAG RFID

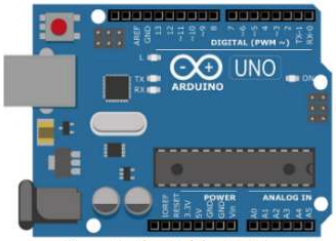
Fonte: Própria.

Para gerenciar todos esses dispositivos é necessário um microcontrolador. O Arduino é um microcontrolador que pode se tornar um minicomputador no qual é possível programar e processar dispositivos externos conectados através de periféricos de entrada e saídos. A aplicação do Arduíno pode ser desde um simples acender de luzes em um intervalo de tempo, utilizando para isso apenas o hardware, quanto ao envio de dados para um site ou dispositivo

móvel, provenientes de algum sensor conectado a ele, utilizando para isso o software e o hardware (McRoberts, 2011).

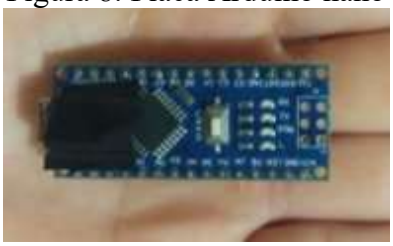
Os componentes principais da placa Arduíno UNO são as conexões de rede, conexão elétrica de alimentação, podendo ser usado cabo USB conectado ao computador, a presença do microcontrolador ATMEGA328, que possui 8 bits, família AVR, arquitetura RISC e um encapsulamento DIP28, 32KB de memória Flash, 2KB de memória RAM e 1KB de memória EEPROM (Soares, 2018). Já a placa Arduino nano possui microcontrolador ATMEGA328 de 5V e 16MHz com 14 saídas ou entradas digitais. A Figura 7 mostra a placa Arduino UNO e a Figura 8 mostra a placa Arduino nano usada neste trabalho.

Figura 7: Placa Arduino UNO.



Fonte: (Oliveira, 2018).

Figura 8: Placa Arduino nano



Fonte: Própria

Para a programação da plana Arduino nano, será usada a IDE de desenvolvimento integrado Code Arduino 1.8.5 que pode ser usado nos sistemas operacionais Windows, OSX e Linux, com linguagem de nível avançado C/C++ com algumas particularidades do microcontrolador usadas pelo ambiente de programação Open Source Wiring (Soares, 2018).

3.3 Software

No software, foi utilizado para a programação do banco de dados a a linguagem SQL para fazer o banco de dados, através do aplicativo MySQL Workbench 8.0.

Para o desenvolvimento da interface do usuário e dos métodos de acesso ao banco de dados foi utilizado a linguagem Java através da IDE NetBeaans. O NetBeans é um ambiente de desenvolvimento gratuito, criado pela empresa Oracle, que integra todos os componentes essenciais para desenvolvimento nas linguagens Java, HTML5, PHP e C/C++, suportando os sistemas operacionais Windows, MAC, Linux e Solaris (Oracle, 2020). Para este trabalho, será usado a versão NetBeans IDE 8.0.

Para a emissão dos relatórios de registros de velocidade, será utilizado uma ferramenta altamente comunicável com o NetBeans chamada iReport. O iReport é uma ferramenta de interface gráfica para a emissão de relatórios desenvolvida em integralmente em Java e altamente comunicável com o NetBeans e a comandos SQL. O JasperReport é a biblioteca que torna a emissão de relatórios possíveis que é usada pelo iReport e pode ser importada ao NetBeans. (MMHOST, s.d). Para a emissão de relatórios, foi usado o iReport na versão 5.6.0 e suas bibliotecas JasperReport.

De modo a permitir a comunicação entre o banco de dados e a interface, é necessário a existência de um servidor ao qual disponibilizará o endereço do banco de dados e a porta ao qual a conexão entre a interface e o banco. Para isso utilizou-se o XAMPP. O XAMPP é um pacote contendo os mais importantes códigos abertos atuais, contendo, principalmente, banco de dados MySQL que será usado para este trabalho, além de outras linguagens tais como PHP, Perl e FTP, e está presente nos sistemas operacionais Windows, MAC, Linux e Solaris. O XAMPP permite que sistemas rodem localmente no endereço http://127.0.0.1, permitindo que sistemas rodem em um servidor sem complicativos (TechTudo, 2012). Para este trabalho será usado o XAMPP Control Panel versão 3.2.4.

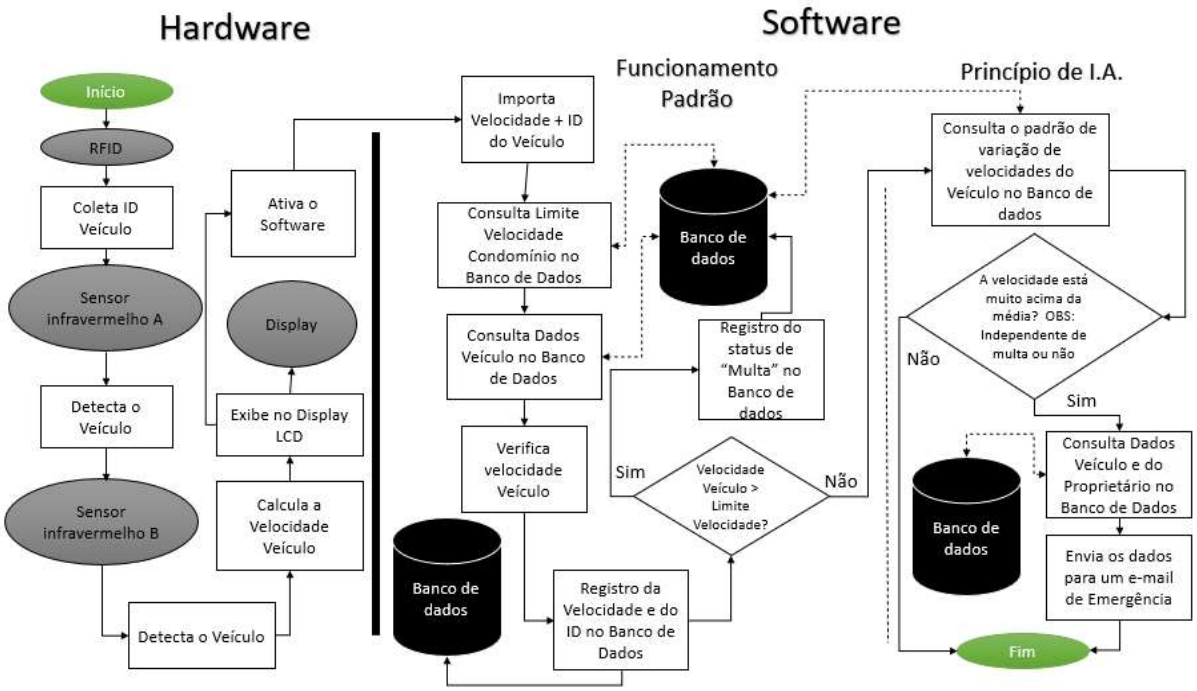
Para a verificação do funcionamento do hardware, antes de sua construção será feito uma simulação para verificar possíveis problemas antecipadamente e verificar as conexões funcionais antes de se partir para a construção do protótipo real. Para isso será usado o software de Proteus. O software Proteus foi desenvolvido para os campos didáticos e profissionais e engloba todas as aplicações analógicas e digitais, perimindo a montagem e esquemas, simulações e layouts. Possuindo uma capacidade maior de simulação de circuitos que outros softwares existentes, o Proteus torna-se um dos melhores softwares de aplicação analógica e digital. (ANACOM ELETRÔNICA, 2010). Para este trabalho, será usada a versão do Proteus 7.6.

4 - Resultados e Discussões

Este capítulo irá apresentar a modelagem do banco de dados, a programação do software em Java, a construção do hardware e a programação do microcontrolador e também o método de detecção de padrão de velocidade. Posteriormente será apresentado a integração entre software e hardware e os testes realizados para validação do protótipo.

O projeto será desenvolvido com duas maneiras de manipulação de dados, uma através do software no qual será feita pelo usuário de modo que que o usuário consiga acessar apenas a interface do aplicativo e não o banco de dados diretamente, podendo fazer o cadastro, exclusão e alteração de moradores do condomínio e de seus respectivos veículos e fazer a consulta dos registros de velocidade através de emissão de relatórios. A outra maneira será feita pelo hardware, no qual será inserido o valore da velocidade dos veículos através da detecção da passagem do veículo pelos sensores infravermelho e tag RFID. A Figura 9 exemplifica, através de um fluxograma, como será a manipulação dos dados dentro do protótipo.

Figura 9: Diagrama de acesso a dados do projeto



Fonte: Própria

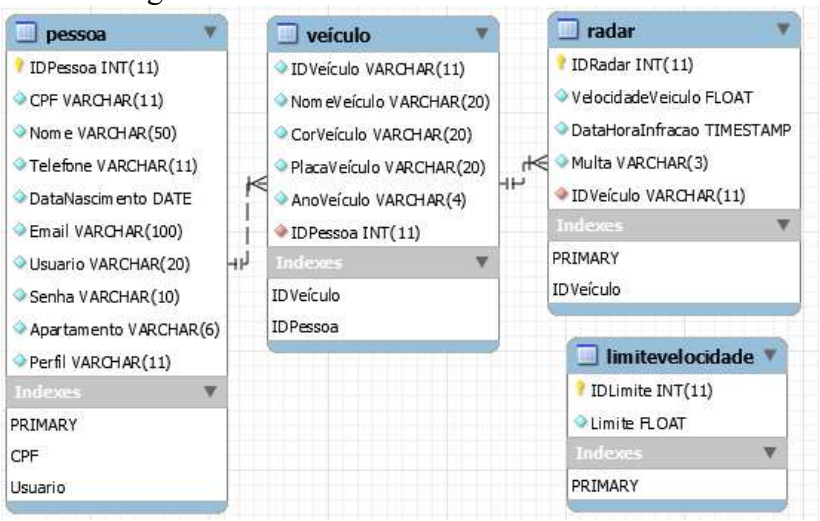
O veículo irá passar pelo TAG RFID onde será identificado seu ID, e em seguida irá passar pelos dois sensores infravermelhos no qual irá detectar a velocidade e converter, através da equação 1, em km/h. Os dois dados: ID e velocidade serão importados através do Arduino para o NetBeans que chamará uma função onde irá detectar o limite de velocidade e o padrão de velocidade do veículo através de consultas automáticas ao banco de dados e irá inserir os

dados importados do Arduino no banco de dados com o acréscimo da informação de multa ou não multa. Já para os padrões de velocidade, o software irá verificar se a velocidade que o veículo passou é superior as médias normais em um determinado parâmetro e se esse parâmetro for ultrapassado uma mensagem de emergência irá ser enviada para um e-mail de emergência. A seguir será melhor detalhado o desenvolvimento do protótipo iniciando-se pelo software com a programação inicial do banco de dados.

4.1 Desenvolvimento do Banco de dados

A programação do banco de dados foi feita em linguagem SQL criando a databade 'TCCProjetoRadarCondomínio' e utilizando o software MySQL. O banco de dados foi desenvolvido de maneira relacional onde as tabelas possuem chaves estrangeiras para interligação das informações. A idealização para o banco de dados consistiu em se criar quatro tabelas: Tabela 'Pessoa', no qual será usado para cadastro das informações pessoais uteis dos moradores e dos funcionários do condomínio, tabela 'Veículo', no qual serão cadastradas as informações úteis dos veículos a tabela 'Radar', no qual serão cadastradas as informações úteis importadas diretamente do hardware como ID e velocidade do veículo e a Tabela 'LimiteVelocidade', no qual será cadastrado o limite de velocidade do condomínio. A Figura 10 mostra o respectivo modelo de entidade relacionamento do banco de dados criados.

Figura 10: Modelo Entidade Relacionamento



Fonte: Própria

Com relação a Figura 10, pode-se verificar a criação de quatro tabelas: Pessoa, Veículo e Radar e LimiteVelocidade. Dentre as tabelas, pode-se destacar a tabela Radar, ao qual armazenará os dados do hardware, possuindo assim os registros 'IDRadar' incrementado automaticamente a cada registro para diferenciação entre as várias coletas de dados, o registro 'VelocidadeVeiculo', que irã armazenar a velocidade coletada do veículo como FLOAT, o registro 'DataHoraInfracao' ao irá coletar a hora do lançamento no banco de dados para registro de dia e hora da passagem do veículo, o registro 'multa' que irá registrar um VARCHAR de 'Sim' ou 'Não' definido status de multa ou não multa que verificara a velocidade coletada, através da programação no NetBeans, e irá compara-la com o limite de velocidade da tabela 'LimiteVelocidade'. A tabela Radar possui uma chave estrangeira diretamente ligada a Tabela Veículos, que por sua vez possui chave estrangeira diretamente ligada a Tabela Pessoa, permitindo assim o uso da função INNER JOIN para junção de informações de tabelas distintas, o que será importante para a geração de relatórios discutidas no capítulo Resultados e Discussões.

4.2 Programação do aplicativo

Para a programação do aplicativo, foi usado a linguagem Java, usando o software NetBeans para programação e compilação. Primeiramente, criaram-se três pacotes, o primeiro pacote, denominado 'TCC.USF.ConexãoBancodeDados', será usado para fazer a integração entre o Aplicativo (Java) e o banco de dados (MySQL), o pacote 'TCC.USF.Icones' será usado para os ícones e figuras que deverão aparecer nas telas do aplicativo e o pacote 'TCC.USF.Telas' será usado para a programação de todas as telas do aplicativo.

A programação do software foi iniciada com o método de integração entre o banco de dados e o aplicativo que é intermediado pelo servidor de banco de dados (XAMPP) em uma conexão local pela porta 3306. Para se fazer essa conexão. dentro do pacote 'TCC.USF.ConexãoBancodeDados' foi criada a classe 'Conexao'. Essa classe descrita para acesso ao banco de dados é mostrada na Figura 11.

Figura 11: Codificação para acesso ao banco de dados

```
1 public class Conexao {
2   public static Connection conectorBD() {
3     java.sql.Connection conexaoBD = null;
4     String driver = "com.mysql.jdbc.Driver";
5     String url = "jdbc:mysql://localhost:3306/TCCProjetoRadarCondominio";
6     String user = "root";
7     String password = "";
```

Fonte: Própria

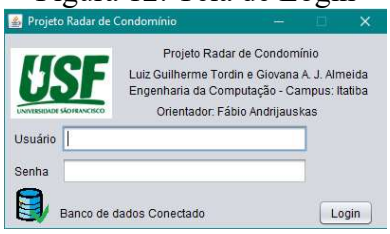
Através da Figura 11 pode-se verificar a classe publica 'Conexao' (linha 1), que possui o método 'conectorBD' (linha 2), e a variável 'conexaoBD' (linha 3) inicialmente instanciada como nula e que receberá a conexão com o banco de dados através de um Try-Catch no qual irá buscar as demais variáveis criadas, sendo elas o 'driver' (linha 4), a 'url' de onde está o banco de dados (linha 5), o usuário 'user' configurado no banco de dados (linha 6) e a senha 'password' se houver para acesso ao banco de dados. Para a programação da classe 'Conexao' foi preciso também importar a biblioteca 'MySQL Connections – Conector/J versão 5.1.49', para ser possível a interpretação da linguagem SQL dentro do Java

Dentro do pacote 'TCC.USF.Telas' criaram-se uma classe para cada tabela do banco de dados (Figura 10), sendo elas 'TelaMulta' para a tabela Radar, 'Tela Pessoa' para a tabela Pessoa, 'TelaMulta' para a tabela LimiteVelocidade e 'TelaVeiculo' para a tabela Veículo, além de outras duas classes sendo elas 'TelaLogin' e 'TelaPrincipal'. Para a programação das classes criadas, primeiramente fez a importação do método de comunicação com o banco de dados em todas as telas. Esse método será usado sempre que um dado precisa ser consultado no banco de dados, ou seja, sempre que terá a necessidade de se executar uma Query SQL. Para isso, deverá ser importado o pacote 'TCC.USF.ConexãoBancodeDados' em todas as classes no qual se deseja executar uma Query SQL e chamar a variável 'conexaoBD'. Esse método consiste em se usar uma três variável: 'pst' para a fazer o PreparedStatement, ou seja, chamar um conjunto de bibliotecas no qual será possível usar códigos SQL internamente no NetBeans, 'rs' para o ResultSet, ou seja, exibição do resultado do comando SQL e 'sql' que receberá o código SQL que será usado no banco de dados. Após configurado as três variáveis, deverá ser usado o tratamento de erros Try-Catch para retornar o resultado esperado da consulta ou então retornar o erro em caso de exceção informando o motivo pelo qual o erro ocorreu.

Para a conexão com o banco de dados, tomando como exemplo o método de consulta de usuário e senha para o login no aplicativo é criado uma função no qual irá receber as variáveis de conexão com o banco de dados, para isso criou-se então a função Login (linha 1), no qual possui a variável 'sql' (linha 2) através de uma String de consulta SQL. Após receber a variável 'sql' é chamado o método Try-Catch 9linha 3) e dentro foram inseridas as variáveis 'pst' para ser possível executar as consultas SQL dentro do Java (linha 4), seguidas das variáveis 'pst' que irão buscar as informações inseridas nos campos de login e senha com as informações contidas no banco de dados, com a variável 'rs' para retorno do resultado. A classe de Login

(TelaLogin) será onde o usuário insere seu nome de usuário e a senha, que serão coletados do banco de dados da tabela Pessoa (Figura 10), através do método de Login onde o LabelUsuario receberá o nome de usuário e o LabelSenha receberá a senha. A Figura 12Figura 12 mostra como ficou a interface de login criada em cima da classe 'TelaLogin'.

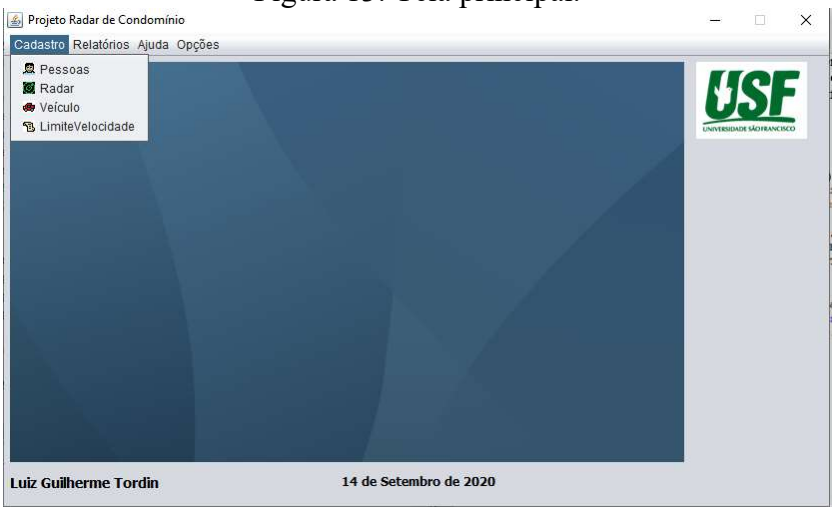
Figura 12: Tela de Login



Fonte: Própria

A classe principal (TelaPrincipal) será onde o usuário terá acesso a todas as funcionalidades do aplicativo, dependendo do seu perfil de usuário. A Figura 13 mostra a interface principal, onde serão chamadas as demais classes (TelaMulta, TelaPessoa, TelaRadar, TelaVeículo), o menu de relatórios e os menus de suporte (Ajuda e Opções).

Figura 13: Tela principal.



Fonte: Própria

Foram configurados dois tipos de perfis de usuário, um para funcionário e outro para morador. Os funcionários terão acesso a 100% dos itens do aplicativo, presentes na tela principal e o morador terá acesso apenas a algumas funcionalidades, das quais se destacam a emissão de relatórios de velocidade. A Figura 14 mostra o método de acesso às telas da classe principal em função do perfil do usuário.

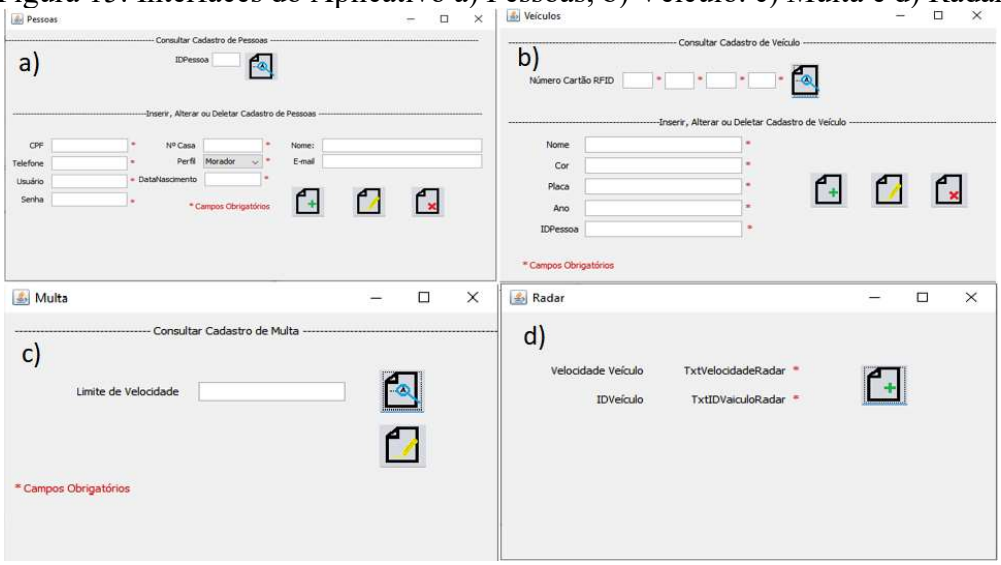
Figura 14: Configuração de acesso vinculado ao perfil de usuário na tela principal.

```
String perfilusuario = rs.getString(10);
    if (perfilusuario.equals("Funcionario")){
    Tela_Principal TLPrincipal = new Tela_Principal();
    TLPrincipal.setVisible(true);
    TLPrincipal.MenuCadastro.setEnabled(true);
    TLPrincipal.LabelUsuario.setText(rs.getString(3));
    this.dispose();
    } else {
        Tela_Principal TLPrincipal = new Tela_Principal();
        TLPrincipal.setVisible(true);
        TLPrincipal.LabelUsuario.setText(rs.getString(3));
        this.dispose();
    }
```

Fonte: Própria

O método que faz esse controle de acesso vinculado ao perfil do usuário está presente dentro da classe ‘TelaLogin’ no método de login e é chamado sempre que a variável ‘rs’ possuir resultado verdadeiro, ou seja, quando houver um usuário cadastrado no banco de dados. A configuração de acesso, que pode ser vista na (Figura 14) consiste em se criar a variável ‘perfilusuario’ (linha 1) que irá consultar o perfil do usuário dentro do banco de dados sendo Funcionário ou Morador. Se for funcionário (linha 2), será chamada a tela principal (linha 3) e selecionado os itens de menu, vistos na Figura 13, como visíveis (linhas 4,5 e 6) e depois irá encerrar a tela de Login (linha 7). Caso o perfil seja morador (linha 8), será chamado a tela principal (linhas 10 e 11) e encerrado a tela de login (linha 12). Como padrão as telas que possuem restrição de acesso já são configuradas como invisíveis sendo habilitadas somente a perfis de Funcionário. As demais classes: TelaMulta, TelaPessoa, TelaVeículo e TelaRadar também possuem suas respectivas interfaces que podem ser vistas na Figura 15.

Figura 15: Interfaces do Aplicativo a) Pessoas, b) Veículo. c) Multa e d) Radar



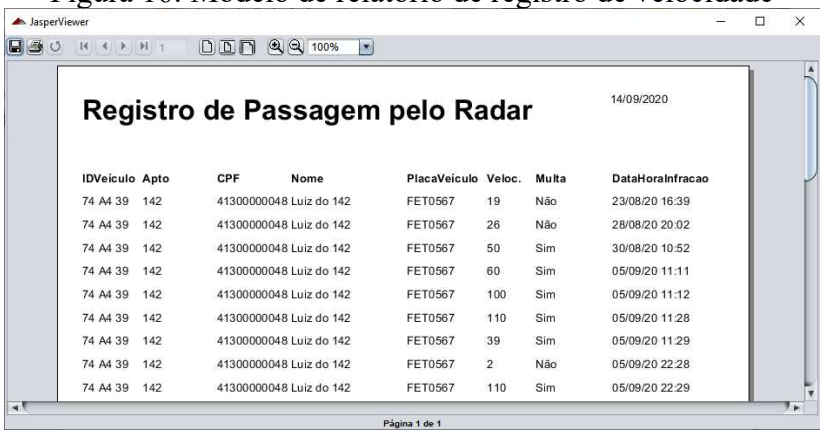
Fonte: Própria

As interfaces TelaMulta, TelaPessoa, TelaRadar e TelaVeículo serão usadas para fazer todo o cadastro necessário para inserir, alterar, consultar ou deletar do banco de dados. A interface TelaMulta está configurada apenas para alteração e consulta do valor do limite de velocidade da tabela ‘LimiteVelocidade’ do banco de dados, não permitindo que sejam criados dois limites de velocidade para o condomínio, mas sim, apenas um que pode ser editado e/ou consultado. A interface TelaPessoa é onde o funcionário poderá cadastrar todas as informações dos moradores na tabela Pessoa do banco de dados, alterá-las, consulta-las ou deletá-las. A interface TelaVeículo é onde o funcionário poderá cadastrar todas as informações dos veículos na tabela Veículo do banco de dados e também poderá alterá-las, consulta-las ou deletá-las. Tanto na TelaPessoa quanto na TelaVeiculo, cada campo presente na interface corresponde respectivamente aos campos configurados nas tabelas do banco de dados, conforme visto na Figura 10. Por fim, a interface TelaRadar é onde será coletado as informações provenientes do hardware como velocidade e ID do veículo, sendo inseridas de maneira automática, cujo método será melhor detalhado no capítulo 4.4.

No menu relatórios (Figura 13), é possível emitir três tipos de relatórios, sendo eles: moradores cadastrados, veículos cadastrados e registros de velocidade. Todos os relatórios foram criados com o auxílio do iReport e suas bibliotecas JasperReports importadas ao NetBeans. Para os relatórios de moradores foi criada a consulta geral na tabela Pessoa com o comando SQL ‘SELECT * FROM Pessoa’ retornando todos os registros da tabela Pessoa existente no banco de dados e para os veículos foi criado também uma consulta geral com o comendo SQL ‘SELECT * FROM Veículo’ retornando todos os dados da tabela veículo

existente no banco de dados. Para o relatório de multas, foi preciso juntar as informações das tabelas foi preciso fazer um 'SELECT' nas três tabelas: Radar, Veiculo e Pessoa juntando as informações através das chaves estrangeiras que puderam ser observadas na Figura 10. Essas informações são juntadas através de um 'INNER JOIN' interligando essas chaves estrangeiras. Os três relatórios exibem um modelo que pode ser impresso ou salvo em PDF. A Figura 16 mostra o modelo de relatório emitido pelo aplicativo com o exemplo do relatório de registros de velocidade.

Figura 16: Modelo de relatório de registro de velocidade

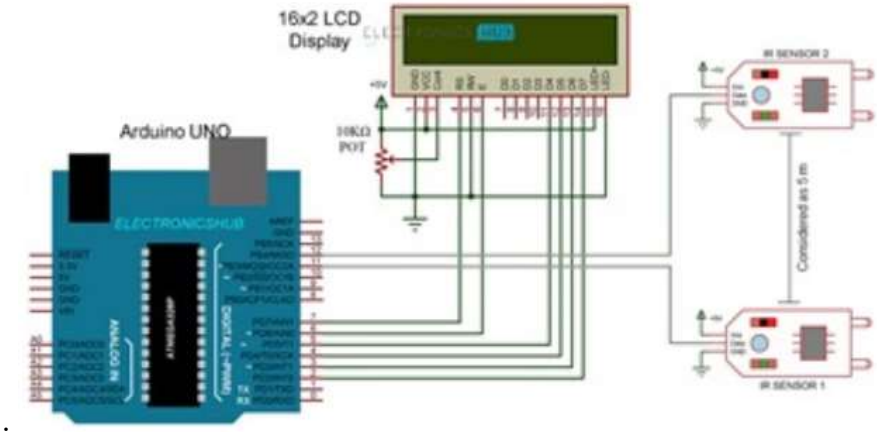


Fonte: Própria

4.3 – Modelagem do Hardware

Primeiramente, fez-se uma simulação de montagem, utilizando o software Proteus, para verificar e analisar se o projeto irá funcionar antes de iniciar a montagem do protótipo. Foi usado na simulação a placa Arduíno UNO conectada a ela um Dysplay LCD e dois sensores infravermelhos. A Figura 17 mostra como ficou a simulação feita no Proteus.

Figura 17: Simulação da montagem do protótipo



.

Fonte: Própria

Após a simulação, fez-se a montagem do leitor de TAG RFID e dos sensores infravermelhos instalando-os na placa Arduíno nano. Após a montagem iniciou-se a programação da placa Arduino nano usando sua IDE própria Arduíno 1.8.5. Primeiramente foi feito os includes das bibliotecas dos hardwares: Display LCD (LiquidCrystal.h), RFID (MFR522,h) e do microcontrolador da placa Arduíno (SPI.h), e também foi definido as entradas para os dois sensores infravermelho, através dos pinos 9 e 10 do Arduíno e os set e reset do modulo RFID com o SS_PIN e RST_PIN.

Para atender ao fluxo de dados da Figura 9, relacionado a parte de hardware, foram criadas as variáveis globais do Arduino, sendo elas: 'TimePrimeiro' que irá coletar o tempo de passagem no primeiro sensor infravermelho, 'TimeSegundo' que irá coletar o tempo de passagem no segundo sensor infravermelho, 'diff' que irá subtrair os tempos coletados pela variável de tempo 'TimeSegundo' pela variável 'TimePrimeiro', tendo como retorno o tempo de passagem entre os dois sensores infravermelho, a variável 'speedConst' que fornece a distância (em cm) entre os dois sensores infravermelho e a variável 'velocidade' que irá calcular a velocidade média de passagem do veículo pelos sensores infravermelho. Usando a equação 1, pode-se elaborar o cálculo da variável velocidade, tendo como resultado a velocidade média em cm/milissegundos.

$$Vméd = \frac{\Delta s}{\Delta t} = \frac{speedConst}{diff} = \frac{cm}{millisegundo}$$

Para transformar a velocidade no padrão km/h, foi criada a variável 'velocidade_real' no qual irá multiplicar o resultado obtido na variável velocidade por 36, resultando no padrão km/h. O método para cálculo da velocidade foi inserido dentro da função 'loop' e pode ser visto na Figura 18.

Figura 18: Métodos para conversão da velocidade

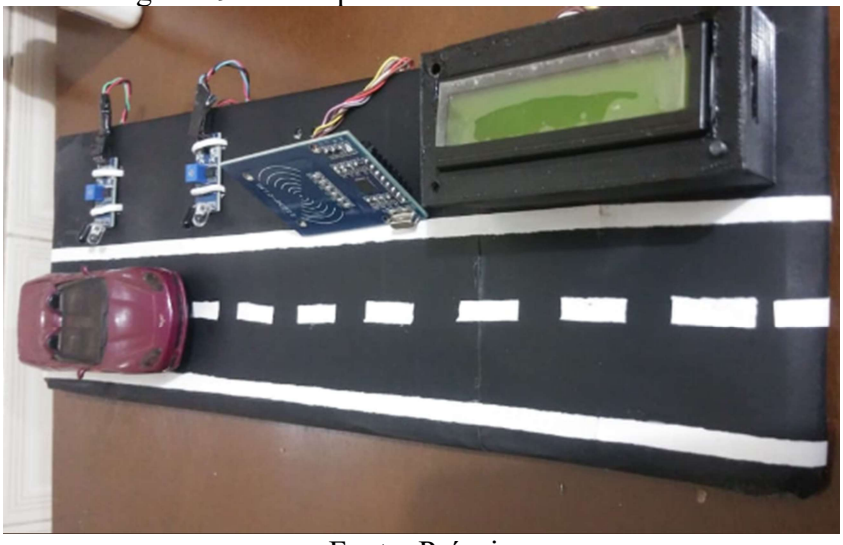
```
1 timeSegundo = millis();
2 diff = timeSegundo - timePrimeiro;
3 velocidade = speedConst / diff;
4velocidade_real = (velocidade*360)/10;
```

Fonte: Própria

Na Figura 18, pode ser visto a definição do tempo em milissegundos (linha 1), o tempo de passagem entre os dois sensores infravermelho 'timePrimeiro' e 'timeSegundo' pela variável 'diff' (linha 2), a velocidade média em função da equação (1) e das variáveis 'diff' e 'speedConst' (linha 3) e a velocidade real que deverá ser enviada para o NetBeans (linha 4). Com isso todo o fluxo de dados da Figura 9 na parte de dados é atendido e os resultados podem ser enviados para o software. Para isso foi criada uma nova variável 'conteudo' que recebera o ID do TAG no qual será o mesmo Id Veículo da tabela veículo e, juntamente com a variável 'velocidade-real' serão exportadas para o NetBeans.

No protótipo final, os dois sensores infravermelho foram colocados a uma distância de 7,5cm de modo que a variável 'speedConst' tenha valor 7,5. A Figura 19 mostra como ficou o protótipo com os sensores infravermelhos, Display LCD e Leitor TAG colocados em sequência.

Figura 19: Protótipo final detector de velocidade



Fonte: Própria

4.4 – Comunicação entre Software e Hardware

Para a integração entre software e hardware, primeiramente foi feita a importação no NetBeans da biblioteca ‘rxtx-2.1.7.jar’ com seus arquivos de seriais presentes no pacote ‘gnu.io’ para estabelecimento da comunicação entre o NetBeans e o Arduino. Após instaladas as bibliotecas necessárias, iniciou-se a programação da lógica de comunicação, criando a classe ArduinoSerial no pacote ‘TCC.USF.ConexãoBancodeDados’. A Figura 20 mostra o método de comunicação entre Arduino e NetBeans.

Figura 20: Método de comunicação entre Arduíno e NetBeans

```
1 public class ArduinoSerial implements SerialPortEventListener {
2     private SerialPort serialPort;
3     private final String namePort;
4     public ArduinoSerial(String portName) {
5         this.namePort = portName;   }
```

Fonte: Própria

Para a lógica de comunicação entre Arduino e NetBeans na Figura 20, a classe ‘ArduinoSerial’ (linha 1) chama a porta serial do Arduíno através da variável ‘serialPort’ (linha 2), definine a variável para o nome da porta ‘namePort’ (linha 3) e cria uma constante ‘arduinoSerial’ em função do nome da porta ‘namePort’ (linha 4) e seleciona esse nome (linha 5) para estabelecer a comunicação. Após criada a classe, criou-se método ‘initialize’ onde irá estabelecer o método de comunicação entre o NetBeans e o Arduíno através de um ‘TRY-CATCH’ Figura 21 mostra a tentativa de conexão com a porta do Arduino.

Figura 21: Método 'try' para estabelecimento de conexão com a porta do Arduino

```
1 try {
2     serialPort = (SerialPort) portId.open(this.getClass().getName(),
          TIME_OUT);
3     serialPort.setSerialPortParams(DATA_RATE,
4             SerialPort.DATABITS_8,
5             SerialPort.STOPBITS_1,
6             SerialPort.PARITY_NONE);
7     input = new BufferedReader(new InputStreamReader(serialPort.getInputStream()));
8     output = serialPort.getOutputStream();
9     serialPort.addEventListener(this);
```

Fonte: Própria

O método irá fazer a tentativa, através de um ‘try-catch’ (linha 1) e em seguida é definido a ID da variável ‘serialPort’ visto na Figura 20 (linha 2) e é passado então os parâmetros (linha 3): DATABITS -8 (linha 4), STOPSITS-1 (linha 5) e PARITY-NONE (linha 6) para a variável ‘serailPort’. Em seguida é chamada a variável ‘imput’ que irá fazer a leitura dos dados vindos do hardware (linha 7), a variável ‘output’ que irá enviar dos dados recebidos do hardware (linha 8) e configurado o novo evento para ser recebido pela classe em questão (linha 9). Caso a conexão seja estabelecida, a variável ‘serialPort’ receberá a conexão com o Arduino, senão é mostrado um erro de exceção.

Após estabelecido o método de comunicação entre Arduino e NetBeans, é necessário criar a função, dentro da classe ‘TelaRadar’ que irá fazer toda a fluxo de dados visto na Figura 9 relacionado à software, incluindo o projeto de rdar inteligente, que será melhor discutido no capítulo 4.5. Primeiramente na classe ‘TelaRadar’, similar ao que foi feito com banco de dados, é necessário importar a classe ArduinoSerial para se estabelecer a conexão com o Arduino. Para essa classe, as classes ArduinoSerial e ConexaoBD são importadas de maneira conjunta dentro da classe principal da ‘TelaRadar’. Após importadas as conexões, é necessário importar as variáveis ‘conteudo’ e ‘velocidade_real’ do Arduino. Para a variável ‘conteudo’, foi necessário um tratamento para coletar apenas os campos desejados da String no Netbeans usando o SUBSTRING(3,14) para coletar apenas o ID sem espaços extras ou caracteres que estavam sendo importados também via hardware.

Após a importação, os valores coletados aparecem na interface Radar e em seguida é chamado o método 'InserirBD' no qual irá inserir os valores das duas variáveis no banco de dados e executar uma consulta na tabela 'LimiteVelocidade' no qual irá gerar status de multa ou não multa dentro da própria tabela Radar do banco de dados dependendo se a velocidade do radar foi superior al limite pré estabelecido.

4.5 – Princípio de radar inteligente

Para o projeto do radar de condomínio, foi inserido um método para transformar o radar em um radar inteligente. O radar verifica e armazena o registro de velocidade de todos os veículos que passam por ele, permitindo assim a verificação de padrões de velocidade de determinado veículo e, ao passar com uma velocidade muito acima do padrão, um alerta é enviado como uma forma de emergência, pois algo pode estar errado com o motorista naquele momento ou naquele dia, permitindo assim que a gerência do condomínio possa tomar alguma providência antecipada.

Esse método também foi inserido classe 'TelaRadar', após o método descrito no capítulo 4.4, e consiste em se comparar o valor da velocidade coletada pelo hardware com uma consulta no banco de dados, especificamente na tabela radar, da média das velocidades anteriores do veículo em questão pela função 'SELECT AVG(VelocidadeVeiculo) FROM Radar WHERE IDVeiculo =?', sendo que no lugar da '?' será o IDVeiculo coletado via hardware.

Se a velocidade do veículo não for anormal, a velocidade é registrada normalmente conforme fluxograma da Figura 9. Porém, ainda levando em consideração o fluxograma, caso o radar registre uma velocidade anormal, uma consulta SQL é chamada dentro do método de registro da velocidade que irá levantar os dados do veículo e do morador juntando-os através da chave estrangeira IDPessoa com um INNER JOIN nas tabelas Pessoa e Veículo para trazer os campos desejados como: 'Nome', 'apartamento', 'CPF', 'Telefone', 'Email', 'DataNascimento' e 'IDVeiculo'. Após ser feito a consulta SQL, será chamado a função para emissão do e-mail de emergência com os dados levantados do banco de dados. Primeiramente foi necessário importar as bibliotecas commons-emial-1.5.jar e mail-1.4.7.jar para ser possível a comunicação entre NetBeans e e-mail. A Figura 22 mostra o método para envio do e-mail de notificação.

Figura 22: Host e Port para o envio do e-mail

```
1 String meuEmail = "tccprojetoradarcondominio@gmail.com";
2 String minhasenha = "[redacted]";
3 SimpleEmail email = new SimpleEmail();
4 email.setHostName("smtp.gmail.com");
5 email.setSmtpPort(465);
6 email.setAuthenticator(new DefaultAuthenticator(meuEmail, minhasenha));
7 email.setSSLOnConnect(true);
```

Fonte: Própria

Na Figura 22, é primeiramente definido a variável 'meuEmail' para coletar o endereço de e-mail (linha 1), a variável 'minhaSenha' para coletar a senha do e-mail (linha 2), a variável 'email' que receberá a classe SimpleEmail e suas bibliotecas (linha 3), definido o host 'smtp.gmail.com' para o Gmail (linha 4) e da porta 465 (linha 5) para então ser feita a autenticação (linha 6) e estabelecimento da conexão (linha 70 para ser enviado e-mail de notificação.

4.6 – Testes e viabilidade do projeto

O primeiro teste a ser feito foi se o método 'inserirBD' dentro da classe 'TelaRadar' estava funcionando. Foi então feito uma série de passagens pelo sensor infravermelho, totalizando quarenta passagens intercalando dois TAGs diferentes e verificado se estava

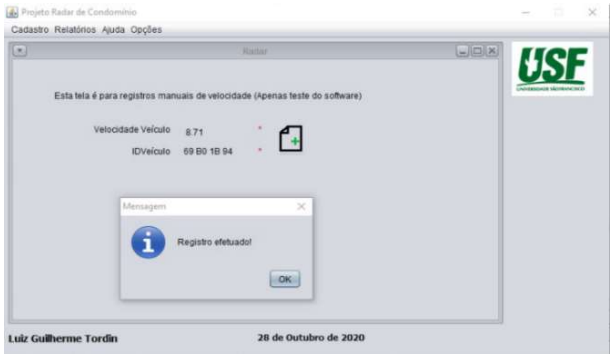
ocorrendo a inserção no banco de dados através de uma consulta na tabela Radar dentro do MySQL. A Figura 23 mostra o resultado da inserção dos valores dentro da tabela Radar em uma consulta no MySQL após as quarenta passagens do carrinho e a Figura 24 mostra a interface 'TelaRadar' após uma inserção bem sucedida.

Figura 23: Dados inseridos pelo hardware na tabela Radar

IDRadar	VelocidadeVeiculo	DataHoraInfracao	Multa	IDVeículo
27	8.71	2020-10-28 20:40:48	Não	69 B0 1B 94
28	8.71	2020-10-28 20:57:59	Não	69 B0 1B 94
29	8.71	2020-10-28 20:58:02	Não	69 B0 1B 94
30	9	2020-10-28 20:58:09	Sim	69 B0 1B 94
42	8.9	2020-10-28 21:12:34	Sim	74 A4 39 22
43	8.9	2020-10-28 21:12:36	Sim	74 A4 39 22
44	8.9	2020-10-28 21:12:38	Sim	74 A4 39 22
45	-1	2020-10-28 21:13:19	Não	69 B0 1B 94
46	-1	2020-10-28 21:13:23	Não	69 B0 1B 94
47	9.31	2020-10-28 21:13:36	Sim	69 B0 1B 94
48	9.31	2020-10-28 21:13:39	Sim	69 B0 1B 94
49	9.31	2020-10-28 21:13:42	Sim	69 B0 1B 94

Fonte: Própria

Figura 24: Interface 'TelaRadar' com inserção bem sucedida



Fonte: Própria

Analisando a Figura 23, pode-se verificar que alguns valores não são coletados, aparecendo resultado de erro "-1". Com isso, mais alguns testes foram feitos para verificar a distância máxima que os componentes conseguem detectar e a velocidade máxima e mínima que pode ser detectado também. A Tabela 1 mostram os resultados dos testes para velocidade máxima e mínima que o hardware consegue detectar bem como as distâncias máximas tanto para o infravermelho quanto para o Arduino.

Tabela 1: Testes realizados no protótipo

Descrição	Valor	Unidade
Velocidade máxima	9,31	Km/h
Velocidade mínima	8,71	Km/h
Distância máxima sensor infravermelho	3,5	cm
Distância máxima RFID	2	cm

Fonte: Própria

Analisando a Tabela 1, não seria possível, em um projeto real, utilizar os componentes descritos neste trabalho para o TAG RFID e para os sensores infravermelho, tendo a necessidade de serem adaptados por hardwares equivalentes mais potentes com distância maior de detecção. Para isso, seria necessário substituir o TAG RFID por uma Antena RFID UHF Linear de longo alcance e os sensores infravermelho por sensores infravermelhos de barreira. Isso acarretará na elevação do custo do projeto real. A Tabela 2 apresenta o custo estimado para o projeto de radar proposto.

Tabela 2: Custo estimado do projeto

Tipo	Descrição	Preço
RFID	Antena RFID UHF Linear 12,5dbi de longo alcance	R$ 1190,00
Radar	Radar Infravermelho	R$ 1280,00

Fonte: Própria

Em uma consulta com empresa locadora de radares para rodovias (cujo nome não houve autorização de divulgação), o preço médio de locação de um radar varia de R$ 2800,00 para radar intrusivo a R$ 3500,00 para não intrusivo. O leitor RFID é um componente de alto custo, porém, neste trabalho, seria utilizando o leito já existente nos condomínios, tornando

desnecessária a aquisição desse item. Com isso, o custo estimado total de locação do radar seria em função da antena RFID e do Radar infravermelho, com custo total de R$ 2470,00. Comparando com o radar não intrusivo da empresa de locação, o custo seria cerca de 29,43% menor.

5 – Conclusões

Analisando os resultados obtidos a integração entre software e hardware e as funcionalidades relacionadas a software funcionam bem e atingiu as expectativas. Com relação ao funcionamento do hardware, já era esperado que a distância seria pequena para os componentes usados no protótipo, porem como era apenas um protótipo com o objetivo de validar um possível projeto real, foi escolhido propositalmente. Porém, com relação a velocidade máxima e mínima, esperava-se um range maior com pelo menos a velocidade mínima chegando a um range maior. Com isso, foi necessário propor soluções de melhorias para um projeto real a fim de cobrir essas lacunas, no qual consistiu em se indicar componentes de hardware mais potentes e com range maior. Com relação ao custo, é possível verificar que o trabalho apresenta resultado satisfatório, apresentando um custo 29,43% menor de locação que os radares locados atualmente por fabricantes especializados.

Através do fluxo de dados, é possível verificar a complexidade da manipulação dos dados presentes neste trabalho que, mesmo trabalhando com apenas quatro tabelas, foi preciso a criação de diversos métodos e classes, dos quais os principais foram discutidos no capítulo Resultados e discussões. Em uma possível comercialização desse radar, a locação seria o melhor jeito de garantir ao cliente um radar eficiente, pois o tanto o software quanto o hardware precisariam de manutenções e atualizações periódicas.

Para projetos futuros, há alguns pontos que poderiam ser melhorados. Um deles seria a substituição ou mesmo extensão do software feito no NetBeans por um aplicativo desenvolvido para celulares. Para isso poderia ser usado o Android Studio e a portabilidade não seria algo complexo, pois a linguagem permaneceria o java, necessitando apenas de algumas integrações extras como bibliotecas, alterações de métodos, etc. Ainda com o aplicativo, outro ponto seria transferir o banco de dados, que é localhost para um servidor em nuvem que poderia ser acessado diretamente com o aplicativo onde o morador estiver. Por fim, um outro ponto importante que poderia ser feito é um método extra para o radar inteligente de avisos quando um morador quer receber alguma mensagem de que, por exemplo “seu filho chegou com o seu carro da faculdade as 23h50” através de um SMS. Para isso, seria necessário criação de métodos de envio de SMS, similares aos métodos de envio de e-mail já discutidos neste trabalho.

6 – Referencial Bibliográfico

ANACOM ELETRÔNICA. (2010). *Treinamento Proteus VSM.* n.a: LABSIS COMÉRCIO DE EQUIPAMENTOS EDUCACIONAIS LTDA.

Athos Electronics. (s.d.). *Sensor infravermelho de obstáculo reflexivo e Arduino*. Acesso em 18 de Maio de 2020, disponível em Site da Athos Electronics: https://athoselectronics.com/sensor-infravermelho-de-obstaculo-reflexivo-e-arduino/

Electro, A. (s.d.). (A. Electro, Editor) Acesso em 26 de Maio de 2020, disponível em Site Arduo Electro: https://www.arduoeletro.com

Ferrari, F. A. (2007). *Crie banco de dados em MySQL.* São Paulo - SP - Brasil: Digerati Books.

Hibeler, R. C. (2005). *Mecânica para Engenharia.* São Paulo, SP: Pearson Education do Brasil, 10ª Edição.

McRoberts, M. (2011). *Arduino Básico.* São Paulo, SP- Brasil: Novatec Editora Ltda.

MMHOST. (s.d). *Jasper iReport - Ferramenta para desenvolvimento e geração de relatórios utilizando Java.* Fonte: http://mz.pro.br/LPII/LPII_170313_IReport_I.pdf

Oliveira, C. L. (2018). *Aprenda Arduino - Uma abordagem prática.* Duque de Caixas, RJ - Brasil: Katzen.

Oracle. (2020). *Site da Oracle*. Acesso em 22 de Junho de 2020, disponível em Oracle: <https://www.oracle.com/br/tools/technologies/netbeans-ide.html>

Puhlmann, H. F. (2015). *Embarcados*. Acesso em 18 de Maio de 2020, disponível em Site Embarcados: https://www.embarcados.com.br/modulo-de-display-lcd/

Santos, A. R. (2019). *Desenvolvimento de um Semáforo Inteligente Utilizando Arduino e Sensores Infravermelhos.* Feira de Santana, Bahia: Curso de Engenharia de Computação. Universidade Estadual de Feira de Santana.

Savi, A. F. (2011). *Controle de velocidade em longos trechos por RFID.* Brasília, DF: Centro Universitário de Brasília - UniCEUB. Curso de Engenharia da Computação.

Soares, T. C. (2018). *Registro de Frequência de alunos com Arduíno e RFID.* Maceió: Curso de ciências da computação, Faculdade da cidade de Maceió. .

Sobral, D. B. (2008). *Programação.* Florianópolis, SC - Brasil: Pearson Education.

TechTudo. (2012). *TechTudo*. Acesso em 22 de Junho de 2020, disponível em Site da TechTudo: <https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2012/02/o-que-e-xampp-e-para-que-serve.html>

Ubisse, P. P. (2017). *Desenvolvimento de um sistema automático de detecção de excesso de velocidade na via pública.* Maputo, Moçambique: Licenciatura em engenharia eletrônica e de telecomunicações. Escola Superior de Ciências Náuticas, Departamento de rádio.

Yamada, M. G. (2005). *Impacto dos radares fixos na velocidade e na acidentalidade em trecho da rodovia Washington Luís.* São Carlos, SP: Dissertação (Mestrado em Engenharia Civil). Escola de Engenharia de São Carlos, Universidade de São Paulo.

7 – Relatório de plágio

O relatório de plágio deste trabalho está disponível no link a seguir:
https://drive.google.com/file/d/1t58cgGdFpa-1O_4Xpu_zPFnw8QBPYwXA/view