FATEC SANTO ANDRÉ

CAÍQUE DE CAMPOS MENDONÇA

ERIC ESTEVAN PAGIATO

RICARDO CELSO MICHELETTO JUNIOR

CONTROLE DE DISTÂNCIA NO *AUTONOMOUS PLATOON*

SANTO ANDRÉ

JUNHO - 2018

CAÍQUE DE CAMPOS MENDONÇA

ERIC ESTEVAN PAGIATO

RICARDO CELSO MICHELETTO JUNIOR

CONTROLE DE DISTÂNCIA NO *AUTONOMOUS PLATOON*

Monografia apresentada como requisito parcial para à obtenção do título de Tecnólogo em Mecatrônica Industrial, Fatec Santo André.

Orientador: Prof. Me. Murilo Zanini de Carvalho

Co-orientador: Prof. Me. Eliel Wellington Marcelino

SANTO ANDRÉ

JUNHO - 2018

M539c
Mendonça, Caíque de Campos
Controle de distância no Autonomous Platoon / Caíque de Campos Mendonça, Eric Estevan Pagiato, Micheletto Junior, Ricardo Celso. - Santo André, 2018. – 120f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Mecatrônica Industrial, 2018.

Orientador: Prof. Me. Murilo Zanini de Carvalho

1. Mecatrônica. 2. Navegação. 3. Veículos. 4. Sensores. 5. Comboio autônomo. 6. Automação. 7. I. Pagiato, Eric Estevan. II. Micheletto Junior, Ricardo Celso III. Controle de distância no Autonomous Platoon.

629.89

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

Santo André, 07 de Julho de 2018

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA: **"CONTROLE DE DISTÂNCIA NO AUTONOMUS PLATOON"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. MURILO ZANINI DE CARVALHO_____

MEMBROS:
PROF. ELIEL WELLINGTON MARCELINO_____

PROF. PAULO TETSUO HOASHI_____

ALUNOS:

CAIQUE DE CAMPOS MENDONÇA_____

ERIC ESTEVAN PAGIATO_____

RICARDO CELSO MICHELETTO JUNIOR_____

Agradecemos a Deus, pois sem ele nós não teríamos forças para essa longa jornada. Agradecemos aos nossos professores, nossos familiares e aos nossos colegas que nos ajudaram na conclusão desta monografia.

AGRADECIMENTOS

A esta faculdade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbramos um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Ao nosso orientador Prof. Me. Murilo Zanini de Carvalho, ao nosso coorientador Prof. Me. Eliel Wellington Marcelino e ao Professor Fernando Garup Dalbo, pelo suporte no pouco tempo que lhe couberam, pelas suas correções, incentivos e pelo empenho dedicado à elaboração deste trabalho.

Ao Prof. Dr. Zdenek Hurak pelo suporte e apoio fornecidos.

Aos nossos pais, pelo amor, incentivo e apoio incondicional.

Nossos agradecimentos aos amigos, companheiros de estudos e irmãos na amizade que fizeram parte de nossa formação e que vão continuar presentes em nossas vidas com certeza.

A todos que direta ou indiretamente fizeram parte da nossa formação, o nosso muito obrigado.

RESUMO

O presente trabalho procura estabelecer um modelo para a navegação semi-autônoma de veículos que estejam se locomovendo em grupos. Este conceito é definido como *Autonomous Platoon* (Comboio Autônomo). Nele, um único motorista é necessário para guiar o grupo, com o restante apenas seguindo-o e espelhando sua direção. Para que isso seja possível, existem três preocupações principais a se levar em consideração: leitura de informações, conectividade entre os componentes e controle de entrada e saída dos automotores no pelotão. Neste projeto, a análise de dados sobre a distância entre um veículo e outro é feita por meio de sensores ultrassônicos. O sensor detecta o tempo do que o sinal ultrassônico leva para transitar entre o veículo em que ele está acoplado e o que está à frente deste e envia essa informação ao mestre do automóvel correspondente. Um Arduino do modelo nano foi utilizado como mestre para cada transporte. Nele, o programa lê a informação fornecida pelo sensor ultrassônico e interpreta os dados de modo a determinar a distância. Com ela determinada, é possível controlar a potência do motor elétrico de modo a regular a velocidade de locomoção em cada um dos automotores. Com isto, o microcontrolador regula a distância entre um veículo e outro. Com a evolução da tecnologia IoT (*Internet of Things*) e da microeletrônica, a conexão é feita por meio de módulos *wireless* em cada um dos veículos, permitindo o controle e arquivos dos dados envolvidos no processo. O veículo mestre é determinado pelo código do servidor que está alocado na Nuvem para processamentos e monitoramentos mais eficientes. Os controladores devem estar conectados entre si para troca de informações. O objetivo dessa conexão é controlar o comboio e organizá-lo a fim de que seu funcionamento seja prudente de acordo com as leis e normas que regulamentam a direção no país.

Palavras-chave: Comboio Autônomo, *Autonomous Platoon,* Arduino, ESP 8266.

ABSTRACT

This paper seeks to establish a model for the semi-autonomous navigation of vehicles that are moving around in groups. This concept is defined as Autonomous Platoon. In it, a single driver is required to guide the group, with the rest just followed him and mirroring his direction. For this to be possible, there are three main concerns to take into consideration: information reading, connectivity between components and control of entry and exit of the automotive in the platoon. In this project, the reading of information on distance is made by means of ultrasonic sensors. The sensor detects the time that the ultrasonic signal takes to travel between the vehicle in which it is attached and what is ahead of it and sends that information to the corresponding vehicle master. An Arduino of the nano model was used as master for each vehicle. In it, the program reads the information provided by the ultrasonic sensor and interprets the data in order to determine the distance. With it determined, it is possible to control the power of the electric motor in order to regulate the speed with which the vehicle is moving. With this control, the microcontroller regulates the distance between one vehicle and another. This distance is predetermined for the safety of the platoon and of those who travel near it on the road, that is, to prevent accidents. With the evolution of IoT (Internet of Things) technology and microelectronics, the connection is made through wireless modules connected in each of the vehicles, allowing the control and archives of the data involved in the process. The master is determined by the server code that is allocated in the Cloud for more efficient processing and monitoring. Vehicle masters (controllers) must be connected to each other for information exchange. The purpose of this connection is to control the platoon and organize it so that its operation is prudent according to the laws and norms that regulate the direction in the country.

Key words: Autonomous Platoon, Arduino, ESP 8266.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A indústria automobilística começou com a produção da Ford em 1903. Desde então esse ramo de produção vem sofrendo modificações, como criações e adequações de modelos que vem sendo produzidos de acordo com as necessidades de seus consumidores, substituição e automatização de maquinário de manufatura buscando obter como resultado a produção em grande escala.

Os problemas no trânsito urbano, devido ao aumento da frota de veículos e de habilidades reduzidas dos motoristas, causam transtornos em quase todos os grandes centros urbanos. Junto deles, problemas como acidentes e desgaste mental do condutor do automóvel, chamam a atenção de toda a sociedade ao redor do mundo.

Atualmente, segundo a Lei Nº 13.103, de 2 de março de 2015 no Brasil, um caminhoneiro pode trabalhar quarenta e quatro horas semanais, podendo acrescer duas horas diárias com pagamento extra. Entretanto, a cada quatro horas de viagem, ele deve descansar trinta minutos; nesse caso, excluindo o horário de almoço também, teria uma carga horária de seis horas diárias.

Um dos tópicos mais relevantes no segmento de veículos autônomos é a implementação de veículos mais inteligentes, capazes de seguir um líder. Enquanto o primeiro veículo da frota tem seu condutor humano, os demais apenas se orientam através dele. A essa fila de veículos se dá o nome de *Autonomous Platoon*.

Dado o cenário acima apresentado, ficou estabelecido como objetivo deste trabalho o desenvolvimento de um sistema capaz de manter a distancia entre veículos que acompanham um veículo principal. Toda a comunicação entre os veículos foi projetada para acontecer de forma autônoma utilizando o protocolo HTTP e um servidor embarcado em um desses veículos.

Este trabalho está divido em revisão sobre os conceitos principais dos assuntos mais relevantes relacionados ao tema do trabalho, a metodologia utilizada para elaboração da pesquisa e do protótipo, o desenvolvimento do trabalho, suas considerações finais e das referências utilizadas em sua elaboração.

2 FUNDAMENTAÇÃO TEÓRICA

Ao longo do tópico foi realizada uma abordagem teórica sobre os conceitos que estão envolvidos no projeto, tais como o comboio autônomo, as tecnologias em sensores, atuadores e processadores utilizados em projetos relacionados.

2.1 Comboio Autônomo (*Autonomous Platoon*)

Cabral & Murphy (2012) relatam que os automóveis surgiram já na primeira metade do século XIX e utilizavam a máquina a vapor, criada por Humphrey Gainsborough em 1760. A partir da segunda metade do século XIX, surgiram os primeiros motores a combustão interna, que são os mesmos modelos utilizados ainda hoje.

Fullin (2015) explica que um novo modelo fora criado por Daimler, em 1896, que utilizava um motor a combustão, se diferenciando dos modelos que utilizavam motor a vapor. As primeiras modificações vieram no ano seguinte por conta de Gottlied Daimler que utilizava um motor ciclo Otto. Esse utilizava gasolina no lugar do diesel e tinha a capacidade de carga de 1,5 toneladas, possuindo apenas 3 marchas.

O autor conclui que, após os primeiros modelos, suas evoluções foram ocorrendo de poucos em poucos anos, tendo sua potência mais que dobrada, mas ainda eram feitos com carrocerias únicas. A adaptação para o modelo conhecido como caminhão cegonha foi feita pelo escocês Alexander Winton, nos EUA, tendo em vista que sua produção estava crescendo demasiadamente e ele não teria como entregar os pedidos em locais distantes, então o modelo era composto de uma plataforma acoplada a um veículo motorizado.

Com o passar dos tempos surgiu a ideia de ter veículos que não são dirigidos por uma pessoa, mas sim por um sistema de controle computacional. Esse conjunto é chamado de veículo autônomo, porém os projetos de ter veículos de transporte autônomos não são exatamente uma novidade, já que, em 1939, os autores Pissardini, Wei, & Júnior (2013), descrevem que foi realizado A Feira Mundial de

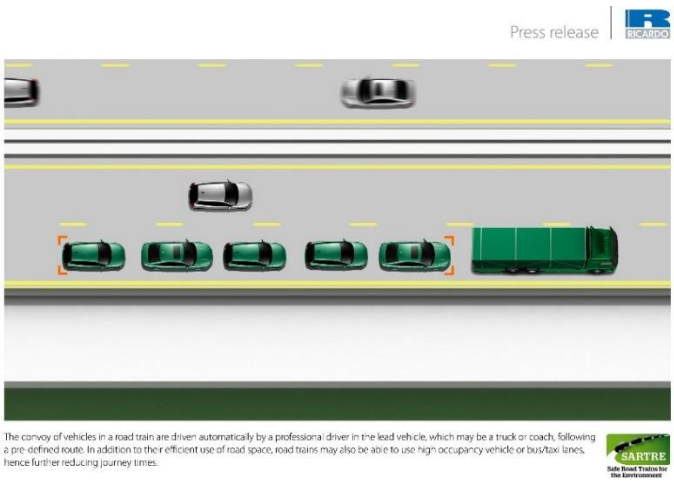
Nova Iorque, nos EUA, que apresentava um protótipo de rodovias automatizadas, onde falhas humanas eram corrigidas pelas estradas, impedindo que ações imprudentes não ocorressem.

Mas a primeira condução autônoma ocorreu apenas em 1958 onde, segundo Pissardini, Wei, & Júnior (2013), cabos elétricos enterrados no solo induziam ondas eletromagnéticas, esse campo eletromagnético exercia reação sobre as bobinas localizadas na parte frontal do veículo.

Em 1977, foi construído pelo Laboratório de Engenharia Mecânica da Universidade de Tsukuba, o que é considerado pela literatura científica como o primeiro veículo robótico inteligente. Pissardini, Wei, & Júnior (2013) se referem a um sistema de visão computacional utilizando câmeras de televisão e uma unidade de processamento acoplados em um veículo.

A Comissão Europeia financiou o projeto *Safe Road Trains for the Environment* (SARTRE), entre 2009 e 2012, onde Pissardini, Wei, & Júnior (2013) dizem que o conceito de comboios lógicos era composto de até oito veículos utilizavam comunicação sem fio e eram conectados à rede fornecida pelo líder que tem sua direção feita por um profissional representado na figura 1.

Figura 1: Modelo de composição SARTRE



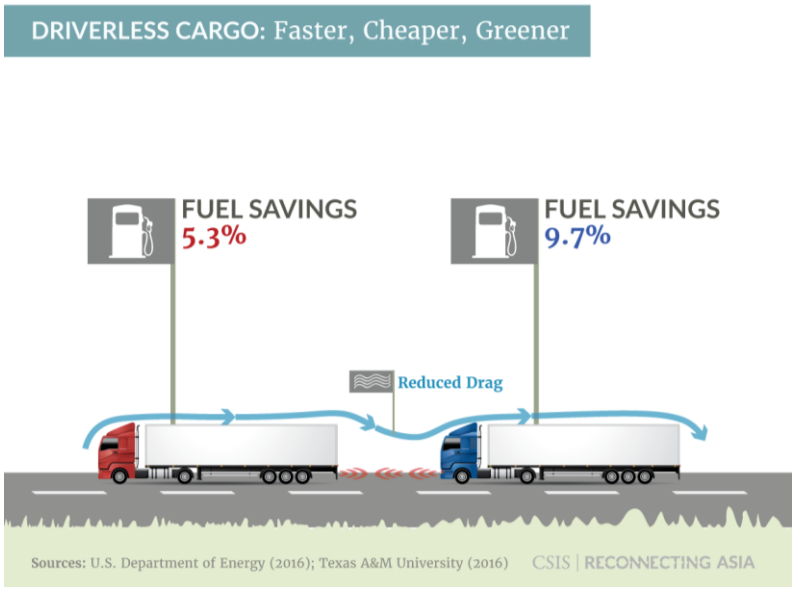
Fonte: Ricardo Plc. Disponível em: <https://ricardo.com/news-and-media/press-releases/cars-that-drive-themselves-can-become-reality-with> Acesso em 2 de nov de 2017.

Segundo Daquino (2014), o conceito do *Autonomous Platoon* consiste em fazer com que caminhões trafeguem em comboio nas estradas, onde o primeiro

veículo será dirigido por um caminhoneiro e os demais terão sua direção baseada e espelhada no líder.

A vantagem dessa formação do comboio, segundo Jaynes (2016), faz com que exista uma redução de consumo de combustível a partir do segundo integrante em diante, gerando economia e redução de emissões de CO_2 (dióxido de carbono), pois o atrito e o arrasto do vento aumentam a quantidade de energia exigida para impulsionar algum veículo. Um exemplo dessa redução pode ser visto na figura 2.

Figura 2: Economia de combustível utilizando *Autonomous Platoon*



Fonte: Reconnecting Asia. Disponível em: <https://reconnectingasia.csis.org/analysis/entries/driverless-eurasia/>. Acesso em 2 de novembro de 2017

2.1.1 Obstáculos para os motoristas

A Lei Nº 13.103/2015, de 2 de março de 2015, é conhecida como “Lei do Motorista” e se aplica aos caminhoneiros e aos condutores de ônibus, visando regulamentar sua atividade remunerada. (Brasil, Lei Nº 13.103, de 2 de março de 2015, 2015).

Com relação à jornada de trabalho de condutores de transporte de carga ou

passageiros, o Art. 235-C da Seção IV-A do Capítulo I do Título III da Consolidação das Leis do Trabalho – CLT que fora aprovada pelo Decreto-Lei Nº 5.452, de primeiro de maio de 1943, vigora que caminhoneiros brasileiros não podem trabalhar mais de oito horas por dia e fazer mais do que duas horas extras. (Brasil, Decreto-Lei N.º 5.452, de 1º de maio de 1943, 1943).

Mas Daquino (2014), afirma que a semi automatização permite que o conjunto realize o transporte por horas, apenas revezando o condutor do líder.

2.1.2 Obstáculos do *Autonomous Platoon*

Alguns testes já foram efetuados ao redor do mundo (DAQUINO, 2014; JAYNES, 2016), na Europa, a figura 3 mostra um comboio da Volvo em operação, EUA, Singapura e Suécia. Porém, o feito mais recente e que ganhou grande notoriedade aconteceu no Japão, onde um comboio de caminhões controlados por computador andou pelas estradas do país.

Figura 3: Comboio de caminhões autônomos cruzando estados europeus

Fonte: 2025 AD: *The Automated Driving Comunity*. Disponível em: <https://www.2025ad.com/latest/2016-review-driverless-cars/>. Acesso em 2 de novembro de 2017.

Um dos maiores desafios encontrados por Contet, Gruer, Koukam, & Gechter (2007) são o controle longitudinal, o controle lateral e a capacidade de se juntar ou se separar do comboio. Onde o controle longitudinal vai variar a velocidade conforme a distância do veículo à frente, e o controle lateral vai alinhar o veículo com seu predecessor.

2.1.3 Propostas de Solução

Os autores Contet, Gruer, Koukam, & Gechter (2007), ressaltam que os problemas podem ser solucionados por meio de um sistema multiagente para comboio. Para isso, os caminhões usam uma combinação de radares, sensores com lasers, câmeras, aparelhos para comunicação de dados via *wireless* e ferramentas de geolocalização. Mediados por sistemas próprios das fabricantes, esses sistemas possuem recursos de segurança que podem identificar um possível acidente com o "veículo-líder", como saída abrupta do traçado. Os veículos conseguem andar em grupos.

Outra solução para alguns dos problemas, para Choi, Kang, Lee, & Kim (2009), é aplicar o mesmo conceito de pista e carros de autorama, onde as ruas teriam traçados pré-definidos e monitorados por antenas, leitores e *tags* de *RFID*, que servem para obter leitura do trajeto e obter informações para o sistema de controle realizar a devida direção do auto, e os veículos teriam sua energização feita por indução eletromagnética.

Segundo Contet, Gruer, Koukam, & Gechter (2007), caso um veículo queira entrar no comboio, ele deve esperar o último veículo passar e entrar atrás dele, essa entrada é determinada pelo sistema. A separação é mais simples: caso um agente queira deixar o comboio, ele notifica o sistema e deixa a formação; o sistema atualiza o índice e as distâncias entre os agentes seguintes.

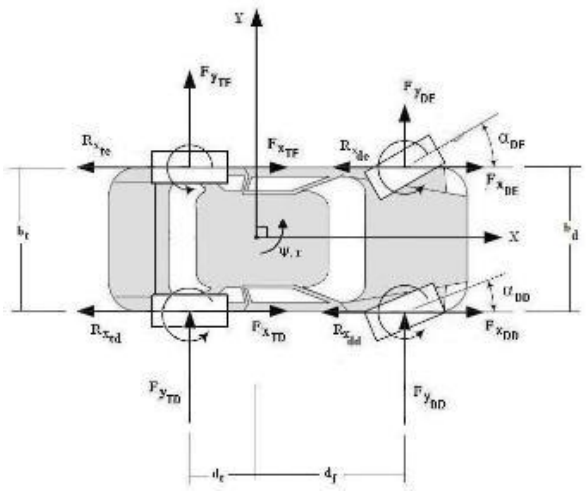
A princípio, como esse sistema ainda está em desenvolvimento, ele deve ser implementado apenas em rodovias porque elas possuem menos obstáculos imprevisíveis em comparação as vias urbanas. Segundo Daquino (2014), um dos problemas a ser enfrentado pelas montadoras será com a justiça, pois mesmo países que já possuem iniciativas de direção sem motorista estão tendo dificuldades em criar leis ou regulamentações para tal transporte.

Órgãos de segurança e justiça deverão fazer vários testes de avaliação para ser possível finalizar o sistema e criar normas regulamentadoras, possibilitando que as nações façam acordos legais.

2.2 Dinâmica Veicular

Segundo Guerra (2017), a dinâmica veicular estuda as dimensões, medidas, formatos, massas, aerodinâmica e suas influências em um comportamento geral, isto é, ela estuda e explica a interação existente entre o motorista, o veículo em movimento e o pavimento. Para tal, ela leva em conta as forças que influenciam no veículo, como é possível observar na figura 4.

Figura 4: Visão superior de distribuição de forças em um veículo



Fonte: Apostila de Modelagem da Dinâmica Veicular. Disponível em: <http://www.ebah.com.br/content/ABAAAg3igAB/modelagem-dinamica-veicular>. Acesso em 15 de novembro de 2017.

A dirigibilidade e a estabilidade de um automóvel são reguladas pela suspensão. Barbieri (1995) ressalta que é um dos componentes responsáveis por controlar as mudanças de atitude do veículo com respeito às irregularidades da via e amenizar a reação que as rodas exercem no corpo do veículo, quando existem variações na superfície da via.

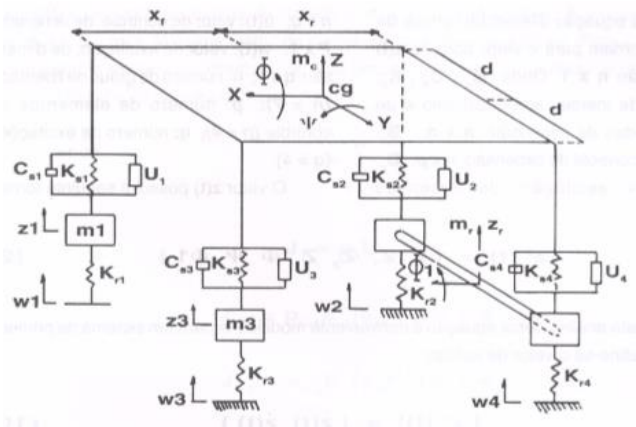
Portanto um veículo precisa ter estabilidade, sendo caracterizada por Leal, Rosa, & Nicolazzi (2012) como a propriedade de retornar ao estado primitivo de marcha depois de cessada uma perturbação transitória, continuando em sentido retilíneo.

Para que o automóvel possa transportar todos os seus componentes, equipamentos, passageiros e bagagem, Barbieri (1995) evidencia que o sistema de

suspensão tem que estar de acordo com o projeto, levando em conta o controle de altura com relação ao suporte de peso do veículo, controle de ângulo de balanço (*pitch*) e ângulo de rolamento (*roll*), que são provocados pelos distúrbios de acelerações laterais e longitudinais.

Segundo Barbieri (2002), todas as forças externas que atuam em um veículo, exceto as forças aerodinâmicas e gravitacionais, estão distribuídas nos pneus como demonstrado na figura 5.

Figura 5: Representação física de um automóvel com 4 eixos



Fonte: BARBIERI, SD, p. 23.

Leal, Rosa, & Nicolazzi (2012) afirmam que as medidas construtivas possibilitam a estabilidade de marcha e manter desvios de curso, realizando correções no volante.

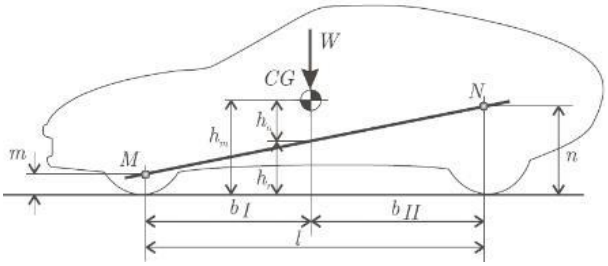
2.2.1 Centro de Gravidade (CG)

Corradi, et al., (2010) afirmam que o centro de gravidade de um corpo é o ponto específico onde uma força da gravidade pode ser aplicada, em outras palavras, é o ponto onde as forças de atração podem se equilibrar.

Os autores também ressaltam que o Centro de Gravidade (CG) pode ser definido como o centro da distribuição do peso de um objeto, sendo que a força da gravidade é atuante no sistema sem alterar o equilíbrio do objeto e é o ponto onde o peso (P) de um objeto se concentra.

Para que as cargas aplicadas sobre as rodas sejam distribuídas de modo a se obter a estabilidade, é necessário ter um ponto chamado Centro de Gravidade (CG). Leal, Rosa, & Nicolazzi (2012) descrevem que é necessário verificar a capacidade de força entre o pneu e a pista para se determinar a posição do CG, pois é neste ponto que as forças peso e inércia irão agir, como mostrado na figura 6.

Figura 6: Demonstração de posicionamento de CG

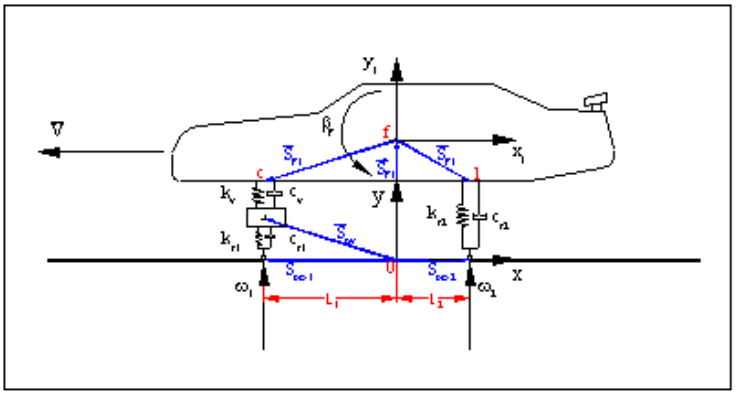


Fonte: Apostila Dinâmica Veicular. Disponível em: <http://www.ebah.com.br/content/ABAAAg3ioAG/apostila-dinamica-veicular> Acesso em 15 de novembro de 2017.

2.2.2 Centro de Massa (CM)

O movimento de um corpo rígido pode ser representado pelo movimento do centro de massa desse corpo. Corradi, et al., (2010) consideram que o centro de massa (CM) é o ponto de equilíbrio do corpo, onde todas as forças externas serão aplicadas, e que o movimento deste ponto é o mesmo que o do sistema como um todo, sendo demonstrado na figura 7.

Figura 7: Modelo de um veículo plano



Fonte: Página do VIII Congresso Nacional do Estudantes de Engenharia Mecânica. Disponível em: <http://abcm.org.br/app/webroot/anais/creem/2001/anais/a09_04.html> Acesso em 15 de novembro de 2017.

A estrutura veicular precisa ter um ponto de Centro de Massa (CM), onde, para Guerra (2017), nos veículos comerciais é buscada a melhor distribuição de peso caso o veículo esteja carregado, levando em consideração uma situação crítica em que a carga total ultrapassa o peso do próprio veículo.

Mas vale ressaltar que não existe uma configuração padrão, o acerto fica por conta da equipe de engenharia responsável pelo projeto. Portanto, “No caso da distribuição de peso transversal, quase sempre o objetivo reside na neutralidade (50:50) (esquerda:direita)”. (Guerra, 2017).

Para saber qual a atitude do chassi, Spinola (2010) descreve sendo necessário estabelecer a cinemática global, que auxilia na determinação da trajetória do Centro de Massa, realizando uma sequência de rotações, levando um referencial paralelo a referência global da estrutura.

2.3 Ultrassom

Bistafa (2011) relata que as ondas sonoras são ondas mecânicas que podem se propagar por qualquer meio material. Porém, há uma em que os seres humanos podem escutar, que está aproximadamente entre 20 e 20000 Hz, como é exemplificado no Quadro 1. O que é reproduzido acima desse limite poderia ser

considerado inútil, caso não pudesse ser utilizado.

Quadro 1: Faixas de frequências audíveis para diferentes animais

INTERVALO DE FREQUÊNCIAS AUDÍVEIS (Hz)	
HUMANOS	20 – 20.000
CÃES	15 – 50.000
MORCEGOS	1000 – 120.000
GOLFINHOS	150 – 150.000

Fonte: Site Brasil Escola. Disponível em: <http://brasilescola.uol.com.br/fisica/ondas-sonoras.htm>. Acesso em 16 de dezembro de 2017.

Um estudo mais avançado do som fez com que essas ondas, que são reproduzidas acima do limite de 20000 Hz, fossem notadas. Cavalcante (SD) diz que a essas ondas deu-se o nome de ultrassônicas. Após descobrirem-nas, era necessário um meio para utilizá-las. Logo, notou-se que certos animais que vivem em ambientes mais escuros conseguiam usar essas ondas como um mecanismo de localização.

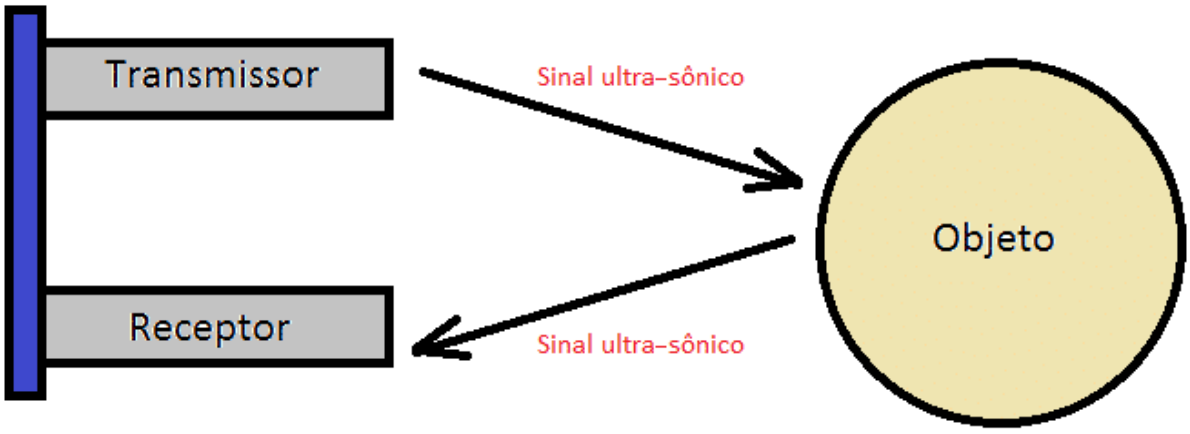
Segundo Souza (SD), a ecolocalização é basicamente a utilização de ondas sonoras para identificar objetos ou seres que podem estar “invisíveis” ou situados a longas distâncias. Assim, golfinhos podem nadar em locais com pouquíssima luminosidade ou morcegos podem sobreviver em suas viagens noturnas.

Então, o ser humano começou a desenvolver métodos com a utilização dessas ondas para realizar certas tarefas. Silva (SD) explica que os pulsos de ultrassom são gerados por cristais especiais que vibram de acordo com uma determinada corrente elétrica. Assim, é possível utilizar frequências de vários MHz (mega-hertz) com duração de apenas alguns milionésimos de segundo para a geração desses pulsos.

Silva (SD) ressalta também que, para utilizá-las, envia-se um pulso de ultrassom para dentro de um material e esse pulso será parcialmente refletido ao encontrar qualquer descontinuidade no meio, como uma mudança na densidade ou defeitos no próprio material. Medindo o tempo decorrido entre o início do pulso e o

seu retorno (eco), pode-se determinar a profundidade e dimensão da falha. Este processo está expresso na figura 8.

Figura 8: Transição do sinal ultrassônico de um transmissor para um receptor



Fonte: Site Embarcados. Disponível em: <https://www.embarcados.com.br/vazao-sensor-ultra-sonico-de-distancia/>. Acesso em 09 de Julho de 2018.

2.3.1 Sensor Ultrassônico HC-SR04

Marcatto & Coelho (2014) mostram que o sensor consegue informar a distância entre dois objetos sem estabelecer contato entre eles. Ele envia uma onda longitudinal e espera pelo seu retorno, que ocorre quando essa for refletida no objeto que se encontra à frente do sensor.

Conforme visto no *Datasheet* do componente, o sinal ultrassônico tem frequência de 40 kHz e um alcance de 0,2 até 4,0 metros com um ângulo de detecção de aproximadamente 15°.

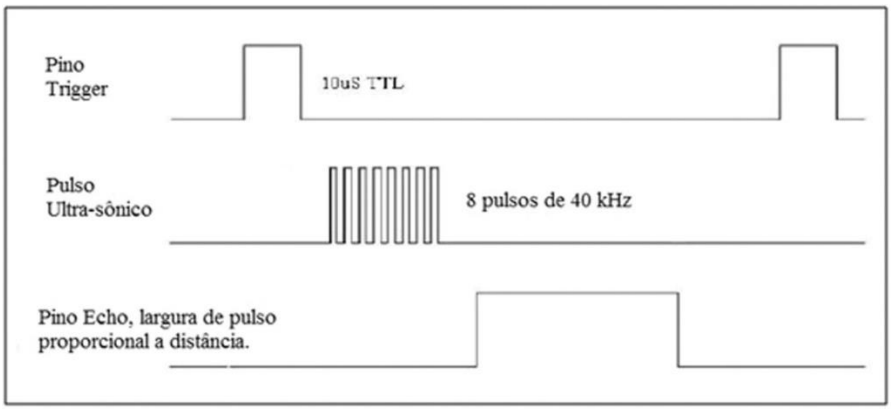
Como pode ser visto na figura 9, o sensor apresenta 4 pinos: Vcc, *Trigger*, *Echo* e GND (*Ground*).

Figura 9: Representação física do módulo HC-SR04

Fonte: *Buildbot*. Disponível em: <http://buildbot.com.br/blog/como-utilizar-o-sensor-ultrasonico-hc-sr04/>. Acesso em 21 set. 2017.

Na Figura 10, vê-se um diagrama de tempo do funcionamento do sensor. Segundo Nakatani, Guimarães & Neto (2013), o módulo, quando alimentado com 5V em seu pino Vcc e 0V no pino GND, funciona pela seguinte lógica: o pino *trigger* recebe um pulso de 10µs ou mais. Um pulso de 8 ciclos de 40 kHz é emitido pelo transdutor Tx e um retorno desse pulso é esperado no transdutor Rx. O intervalo de tempo entre a emissão do pulso e seu retorno constitui a largura, ou *delay*, do pulso gerado no pino *echo*.

Figura 10: Diagrama de temporização do módulo HC-SR04



Fonte: SciELO. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172016000300603> Acesso em 21 set. 2017.

O tempo de voo da onda sonora é utilizado para determinar a distância entre o emissor e o objeto encontrado. A equação 1 relaciona essas grandezas.

$$Dist = \frac{t_{HIGH} \times v_{SOM}}{2} \quad \text{(Eq.1)}$$

Onde:

- Dist é a distância entre o sensor e o objeto medido (em m);
- t_{HIGH} é o tempo total no qual o pino *Echo* emitiu um sinal de nível lógico alto (em s); e
- v_{SOM} é a velocidade do som. Deve-se considerar 340m/s.

2.4 Arduino

Santos (2012) caracteriza o Arduino como uma placa de componentes eletrônicos na qual é possível desenvolver projetos interativos, utilizando periféricos, com uma facilidade grande.

Evans, Noble, & Hochebaum (2012) afirmam que sua criação ocorreu em 2005 por um grupo de pesquisadores italianos que estavam em busca de um dispositivo que fosse funcional, de fácil programação e que fosse acessível monetariamente para todas as pessoas. Portanto, a placa foi feita para prototipagem de projetos. Ela também funciona para teste de circuitos projetados antes de sua montagem definitiva.

Os autores (Evans, Noble & Hochebaum, 2012) completam que a placa é composta por um microcontrolador Atmel e circuitos de entrada e saída, como mostrado na figura 11, podendo ser expandida utilizando placas especializadas chamadas de *Shields*, garantindo mais funcionalidade sem perder pontos de conexões.

Figura 11: Placa Arduino, modelo Uno

Fonte: Site Arduino. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em 15 de novembro de 2017.

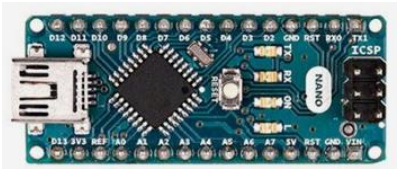
Para sua programação é utilizado uma linguagem de programação baseada em C/C++ (que são linguagens de programação de nível da máquina chamadas de baixo nível) que são desenvolvidas em uma IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado), de fácil utilização, com códigos abertos (códigos *open source*), que, segundo Gugik (2009), consistem, basicamente, em acesso ao código do programa por qualquer pessoa, com a ressalva de que seu desenvolvedor ainda pode determinar como ele será utilizado.

Para aprimorar seu funcionamento, as bibliotecas são conjuntos de recursos utilizados para habilitar funções que facilitarão a execução do projeto. As bibliotecas podem ser obtidas através de sites na internet.

2.4.1 Arduino Nano

O Arduino Nano, conforme o datasheet do componente, é uma placa eletrônica que possui um micro controlador ATMega328. Possui um conector para cabos Mini-USB, utilizados para a gravação de um algoritmo. Ele tem 14 entradas/saídas digitais, 8 entradas analógicas e um jumper de +5V AREF. É possível realizar a alimentação dele utilizando o pino Vin. A figura 12 apresenta o componente.

Figura 12: Arduino Nano



Fonte: https://store.arduino.cc/usa/arduino-nano

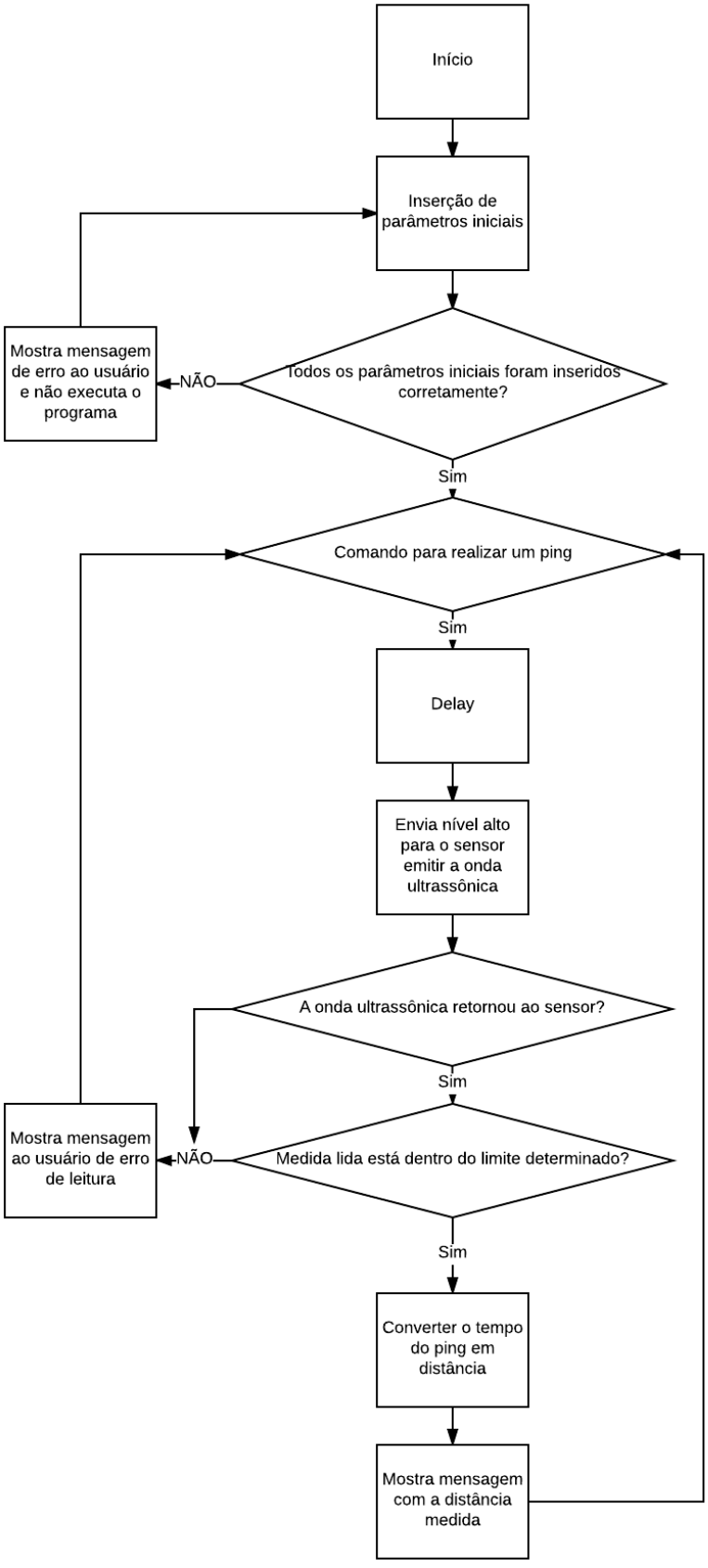
2.4.2 Biblioteca *NewPing*

Todas as informações a seguir sobre a biblioteca *NewPing* foram escritas por Eckel (2017).

É uma biblioteca que funciona com pequenos *delays*, que são tempos de espera que param as atividades do programa para não o travar totalmente. Ela aceita diferentes modelos de sensores ultrassônicos, como, por exemplo: SR04, SRF05, SRF06, DYP-ME007 e Parallax PING.

Para descrever de maneira simples um pouco do funcionamento da biblioteca, é necessário observar o fluxograma na figura 13.

Figura 13: Diagrama de fluxo do funcionamento da biblioteca NewPing



Fonte: Elaborado pelo autor.

É possível dar 30 *pings* por segundo com leituras consistentes e confiáveis, a biblioteca contém um método de filtro digital que facilita corrigir erros de leitura (ping_median()). *Ping* é definido como um pulso de som que é emitido pelo e recebido pelo sensor.

A biblioteca interrompe o temporizador para esboços conduzidos por eventos e permite usar, de maneira simplificada, mais que um único sensor no mesmo programa – tendo como máximo de 15 sensores.

Ela também realiza o cálculo de distância de maneira muito mais precisa, os resultados podem ser obtidos em microssegundos (μS) e convertidos para centímetros (cm) ou polegadas (*inch*).

Como seu mapa de pinos é definido bem no início do código, ele se torna mais organizado e de fácil manutenção ou alteração, caso seja necessário para o projeto.

A biblioteca conta com uma variável específica que determina qual a distância máxima que o programador deseja que o sensor leia. Leituras acima dessa distância não serão feitas.

2.5 *Internet of Things* (IoT)

Internet das Coisas é uma tradução literal da expressão em inglês *Internet of Things (I*oT*)*. Em português, o nome mais adequado poderia ser algo como "Internet em Todas as Coisas". (Alecrim, 2017).

Esse tema vem sendo muito abordado nos últimos anos. Onde o termo descreve, segundo Alecrim (2017), um cenário em que vários objetos estarão conectados à internet e se comunicando mutuamente. Baseia-se na Internet, usando um grande número de técnicas avançadas, como técnicas de identificação de objetos e técnicas de comunicação de dados sem fio, seguindo um determinado protocolo para permitir que diferentes itens se conectem para "conversar" uns com os outros e para trocar informações utilizando a Internet.

A ideia não é, necessariamente, ter novos meios de se conectar com a internet, mas sim buscar outros métodos para deixar objetos mais eficientes ou receber atributos complementares destaca Alecrim (2017). Foi lançado recentemente o *Google Home*, que segundo a própria empresa Google “é um alto-falante ativado por voz que usa os recursos do Google Assistente. Faça perguntas e peça para que ele realize tarefas”.

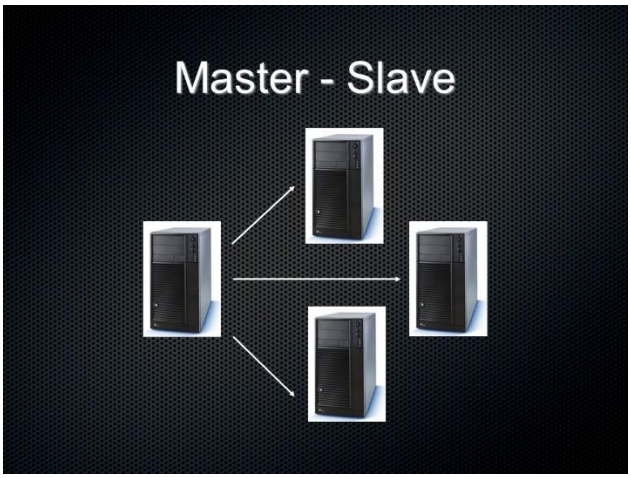
De acordo com Silveira (2017), através da Internet das Coisas pode-se realizar uma rede inteligente de identificação, localização, rastreamento, monitoramento e gestão e possibilitar uma maneira mais fina e dinâmica de viver e gerenciar a produção. Por meio da implementação da interação entre o mundo físico e o mundo virtual, novas aplicações e novos serviços serão habilitados. Um exemplo prático é o que chamamos de Quarta Revolução Industrial ou Indústria 4.0, que permite comandar, de maneira remota, desde uma máquina em um setor de uma fábrica até mesmo uma matriz fabril inteira.

Como ressaltam Karzel, Marginean, & Tran (2016), para qualquer rede de comunicação é necessário ter uma arquitetura de rede, que é um conjunto de camadas e protocolos de rede podendo ter muitas formas diferentes. Esse conjunto deve conter informações suficientes para permitir que um alguém desenvolva programas ou *hardwares* específicos para cada camada.

Para Momote (2016), há quatro modelos básicos de comunicação em IoT, sendo eles: Dispositivo-para-Dispositivo, Dispositivo-para-Nuvem, Dispositivo-para-Intermediário e Análise-de-Dados.

O autor ressalta que, quando se tem uma conexão *Device-to-Device* (Dispositivo-para-Dispositivo), o dispositivo estabelece uma conexão direta com seu subordinado, como mostrado na figura 14, ou seja, é uma relação direta conhecida como Mestre-Escravo (*Master-Slave*), na qual o Mestre manda e o Escravo obedece ao comando.

Figura 14: Esquematização de uma Rede *Master-Slave*

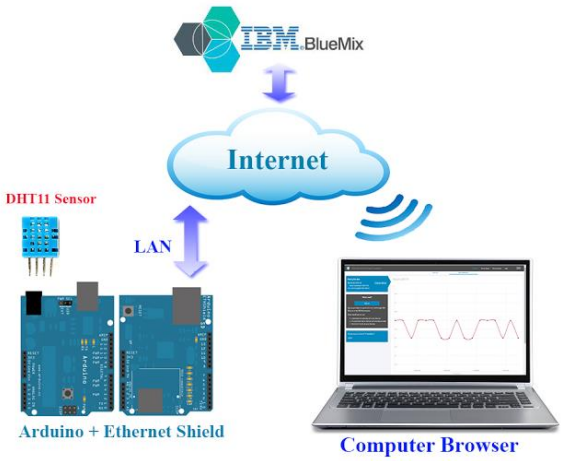


Fonte: Apresentação sobre "NoSQL: O Futuro dos Bancos de Dados para Web". Disponível em: <http://slideplayer.com.br/slide/2578207/> Acesso em 16 de novembro de 2017.

Momote (2016) acentua que conexões *Device-to-Cloud* (Dispositivo-para-Nuvem) são muito frequentes em nossos dias, onde, sem equipamentos intermediários, o aparato se comunica diretamente com a internet.

O escritor evidencia também que redes que necessitam de um intermediário para realizar a comunicação, como a rede Dispositivo-para-Intermediário (*Device-to-Gateway*), um sensor ou um periférico de monitoramento, envia informações para uma central de tratamento de dados. Essa unidade, envia-os para a Nuvem, representado na figura 15. Esse tipo de comunicação vem ganhando grande força no mercado com o esquema de casas inteligentes que possuem dispositivos conectados à rede da aplicação.

Figura 15: Redes *Device-to-Cloud* e *Device-to-Gateway*

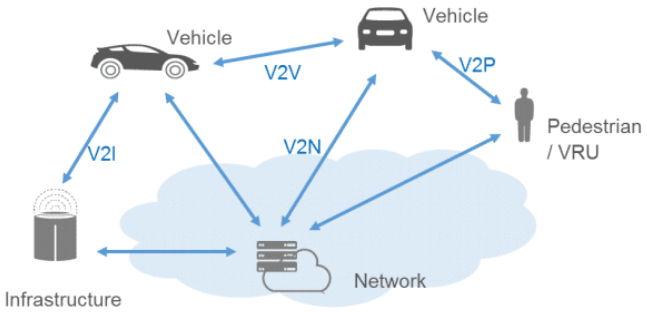


Fonte: Arduino IoT Send Sensor Data to IBM Bluemix. Disponível em: < http://wiznetmuseum.com/portfolio-items/arduino-iot-send-sensor-data-to-ibm-bluemix-2/> Acesso em 16 de novembro de 2017.

Na Análise-de-Dados (*Back-End-Data-Sharing*), o literato conclui que, um grande pacote de dados (*Big-Data*) é exportado para um servidor na Nuvem, neste servidor é possível organizar e monitorar os dados adquiridos durante o período de trabalho da aplicação como um todo, tendo seu monitoramento realizado por páginas de internet ou aplicativos para celular.

Para Soares, Galeno, & Soares (2015), as arquiteturas existentes para comunicação para redes veiculares são: *Vehicle-toVehicle* (V2V- Veículo-para-Veículo), havendo comunicação apenas de veículo para veículo; *Vehicle-to-Infrastructure* (V2I – Veículo-para-Infraestrutura), nessa arquitetura ocorre a comunicação entre veículo e infraestrutura localizada na margem da via; e um Modelo Híbrido (V2X) que é caracterizado pela existência dos dois tipos de comunicação, como mostrado na figura 16, buscando obter as melhores características das duas arquiteturas.

Figura 16: Comunicação existente em sistemas V2X



Fonte: Conhecendo o V2X - Conectando veículos para tudo. Disponível em: <https://www.embarcados.com.br/conhecendo-o-v2x/> Acesso em 16 de novembro de 2017.

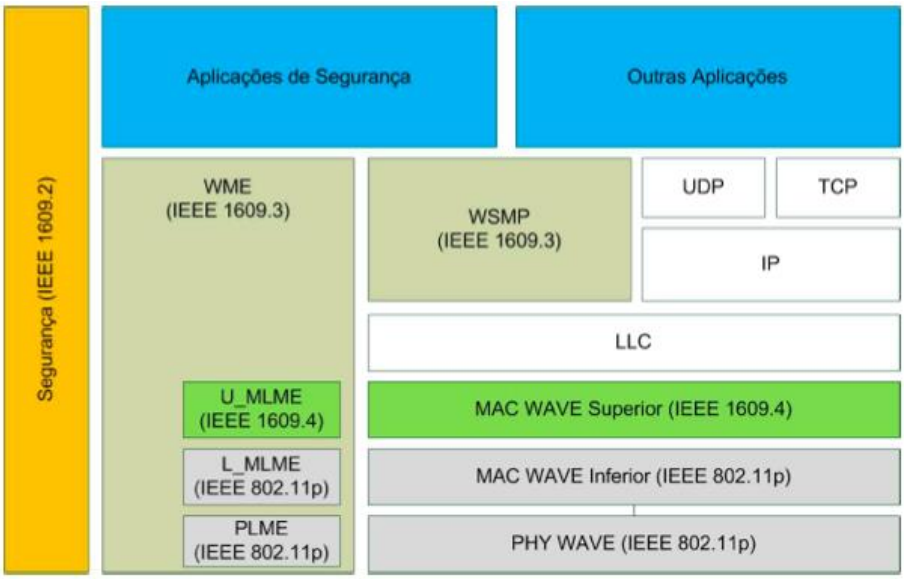
Segundo Soares, Galeno & Soares (2015), desde 2004, o grupo IEEE está desenvolvendo padrões para a rede WAVE *(Wireless Access in the Vehicular Environment),* em busca do desenvolvimento de dispositivos WAVE que auxiliam o transporte seguro realizando transações de dados utilizando baixa latência e baixa sobrecarga.

Alves, et al. (2009) ressalta que a arquitetura WAVE é definida em seis documentos. Onde:

- IEEE 802.11p
  Define as camadas físicas e de controle de acesso ao meio (MAC) para redes veiculares, baseando-se no padrão IEE 802.11ª (redes locais).
- IEEE P1609.1
  Especifica serviços e dispositivos para coordenar a comunicação entre as entidades.
- IEEE P1609.2
  Responsável pela segurança da rede, definindo formatos e processamentos seguros das mensagens.
- IEEE P1609.3
  Define a MIB (*Management Information Base*), que define os serviços das camadas de rede e de transporte.
- IEEE P1609.4
  Define as modificações no padrão IEEE 802.11, para a operação em múltiplos canais.

Para Alves et al. (SD), “A arquitetura WAVE designa uma família de padrões que não se restringe às camadas MAC e física [...]”, à camada IP é uma característica de segurança para aplicações DSRC (*Dedicated Short Range Communications*), que são aplicações em pequenos alcances e operações utilizando múltiplos canais de comunicação, sendo baseada nos padrões IEE 1609. Podendo ser representados pela figura 17.

Figura 17: A pilha de protocolos WAVE.



Fonte: Redes Veiculares:Princípios, Aplicacões e Desafios, SD, p. 4. Disponível em: <https://www.gta.ufrj.br/ftp/gta/TechReports/ACC09.pdf> Acesso em 13 de out de 2017.

2.5.1 Módulo Wireless – ESP8266

Curvello (2015) ressalta que o módulo serve como uma Ponte Serial-*Wi-Fi*, pois pode receber comandos por meio da comunicação serial (UART) e se comunicar com uma rede *Wi-Fi* por meio de conexões TCP/UPD. Thomsen (2015) complementa dizendo que o ESP pode ser ser um Ponto de Acesso (PA ou *Access Point*), enviando comandos ou dados para a internet ou para ser acessado remotamente, ou como uma Estação (*Station*), gerando sua própria rede *Wi-Fi*.

O protocolo de comunicação UART, segundo Braga (SD), é responsável por converter a comunicação paralela em comunicação serial, mudando os dados que

chegam nas entradas paralelas em uma linha de dados sequencial, e vice-versa, utilizando o sentido de transmissão conhecido por *full-duplex* que indica que o dispositivo pode transmitir e receber dados ao mesmo tempo pelo pino receptor (Rx) e o pino transmissor (Tx), onde o Tx deve ser ligado no Rx de outro dispositivo.

Comer (2015) explica que o protocolo TCP/IP é formado pelas siglas de seus dois padrões principais (onde TCP significa *Transmission Control Protocol* ou Protocolo de Controle de Transmissão e IP significa *Internet Protocol* ou Protocolo de Internet). Esses protocolos fornecem regras sintáticas e semânticas para comunicação entre maquinários.

Os autores Comer (2015) e Pereira & Machado (2008) afirmam que o protocolo UDP (*User Datagram Protocol*) manda informações a um destinatário, sem se preocupar se elas foram devidamente recebidas, diferente do TCP. Portanto, em caso de erros, simplesmente ocorre o envio do próximo pacote programado pelo sistema, e os anteriores não podem ser recuperados.

Campos (2015) afirma que o módulo serve basicamente para que um microcontrolador estabeleça conexão por meio de uma rede *Wi-Fi* e que, a partir do momento em que está conectado à rede, ele pode se comunicar com outros equipamentos.

3 METODOLOGIA

Este trabalho utilizou como metodologia a seguinte sequência de tarefas: em primeiro lugar, foi realizada uma extensa pesquisa com trabalhos relacionados ao tema proposto.

Depois, os principais tópicos apontados na pesquisa foram sumarizados na formação da fundamentação teórica deste trabalho. Essa fundamentação foi realizada utilizando artigos de congressos e conferências, livros e revistas relacionadas ao tema Comboios Autônomos. Com ela, os objetivos do trabalho foram traçados.

O foco principal do projeto é o de se controlar a distância entre os veículos que participam de um comboio que atua em um circuito fechado de navegação. Para isso, se mostrou necessário estabelecer um modo de ler essa distância, interpretar essas leituras, controlar os motores, buscar o equilíbrio dos veículos, estabelecer uma comunicação entre eles e controlar as informações pertinentes para o funcionamento do comboio.

Seguido da fundamentação, o levantamento dos sensores e das possibilidades de placa controladora foram listadas. Desse levantamento, as principais características determinadas pela pesquisa foram utilizadas como critério para a escolha dos componentes.

A escolha dos sensores e da placa controladora teve como foco atender a limitação de realizar a detecção do veículo que está à frente do veículo em que o sensor está alocado e realizar a troca de mensagens utilizando o padrão *Wi-Fi* (802.11), para construção de uma rede entre os veículos. O processamento de imagens e detecção de faixa (*lane-detection*) não foram contemplados nesse trabalho.

O início do desenvolvimento do algoritmo de funcionamento utilizou como objetivo controlar o acionamento do motor de maneira que fosse possível controlar sua velocidade de trabalho, limitando a sua corrente e sua tensão de funcionamento enviando ondas em PWM para que o acionamento não seja direto, mas controlado.

Os testes de bancada com as placas controladoras foram realizados, na primeira tentativa, a fim de controlar um motor e testar componentes adequados

para seu funcionamento. Isso ajudou a delimitar o que seria necessário para compor os veículos utilizados no projeto. Como o motor utilizado é simples e para aplicações pequenas, isso auxiliou a determinar e limitar os componentes utilizados para efetuar seu controle, não necessitando de equipamentos industriais, mas sim de micro controladores e elementos eletrônicos.

Para o cálculo e a medição de distâncias, foi necessária a criação de uma equação para converter a velocidade que o som gerado pelo sensor leva para atingir um objeto e voltar em distância entre o sensor e o objeto. Foi modificada a codificação para que o controle do atuador fosse em função da distância, limitando as distâncias máxima e mínima a serem medidas.

No segundo experimento, adicionou-se o sensor de distância para controlar o acionamento, a velocidade deste motor e medir a distância. Isso foi realizado para que fosse possível determinar se o controle de posicionamento do veículo estava funcionando e se precisaria de ajustes. Determinaram-se limites de trabalho para o motor, descobrindo que o mesmo mantém uma velocidade constante de acordo com a medida que o sensor retorna da leitura para evitar a colisão com o objeto localizado a frente.

Um servidor foi desenvolvido para o controle de dados do projeto. Como um dos objetivos do trabalho é a troca de dados entre os veículos, um servidor e um banco de dados se fizeram necessários para que fosse possível um controle sobre o comboio de uma forma geral.

O controle do veículo líder também é feito pelo servidor neste projeto para simular as ações de um motorista na realidade. O servidor foi desenvolvido de forma local.

O banco de dados foi feito partindo-se do princípio de que seria melhor alocar cada informação armazenada nele em uma linha diferente, para que a visualização e atualização de informações fossem mais simples.

O módulo de comunicação faz a ponte que realiza a comunicação entre os veículos e entre os mesmos com o sistema supervisório. Então, fez-se necessário criar um arquivo de código para habilitar o acesso do módulo à internet, tendo como primeiras limitações a aquisição de dados discretos e envio de comandos simples.

Isso foi realizado para saber como o módulo poderia criar um meio de acesso ao servidor desenvolvido, mesmo que este seja local.

Para o terceiro ensaio, foi necessário criar um circuito de teste para habilitação e utilização dos módulos de comunicação, conectando-os em uma rede de internet, respondendo comandos de funcionamento simples, onde realizou a troca de dados entre dispositivo e página da internet. Limitando parâmetros necessários que o módulo necessitou para acessar a rede de comunicação; e parâmetros que o usuário necessita para monitorar e modificar parâmetros de controle.

Com o módulo de comunicação já habilitado e funcionando para a aquisição de dados obtidos pelo sensor e para habilitar a modificação de parâmetros de funcionamentos, tornou-se necessário realizar a união e ligação dos componentes. Mas ainda tendo como limite não implementar os comandos da trajetória a ser feita.

O próximo exame realizado uniu o sensor de distância, com a placa controladora e ao módulo de comunicação, para que fossem realizados comandos simples para medição de distância, acionamento e controle do motor. Isso foi realizado para determinar se era possível controlar as plataformas de maneira remota.

Os testes de bancada de cada sensor com suas placas controladoras foram realizados com a finalidade de verificar seu funcionamento e comparar a validade das informações obtidas. Assim, tornou-se possível validar a utilização dos sensores.

A proposta do protótipo do veículo foi realizada e validada junto à fundamentação teórica e com profissionais da indústria automobilística com foco no cálculo para distribuição de peso e componentes, de modo que o protótipo veicular fique com uma estabilidade e conduções semelhantes com a de um automóvel real, e na busca do melhor modelo a ser utilizado. Esses cálculos garantem que o veículo possa manter uma estabilidade necessária para se controlar o comboio.

A premissa do projeto envolveu desenvolver um modelo para controle de distância que possa ser utilizado em um comboio autônomo. Esse controle tem como meta aprimorar a segurança durante a locomoção deste comboio e reduzir

gastos, pois, além de garantir um espaço seguro entre os componentes do mesmo, ele auxilia o controle de velocidade e minimiza os gastos de combustível do veículo uma vez que a diminuição do atrito com o ar proveniente da redução do espaço entre os veículos também diminui os esforços.

A integração dos sensores e do controlador ao protótipo foi realizada, e os testes preliminares do controlador e dos sensores foram refeitos para garantir sua calibração na plataforma.

Mais duas plataformas de teste foram construídas nos mesmos moldes que a primeira para simular os demais veículos do comboio.

O algoritmo para seguir o líder em linha reta e evitar colisões foi simulado, buscando manter uma distância com poucas variações entre veículos e objetos localizados a frente de cada plataforma. Seus resultados foram comparados com os obtidos com os da sua implantação nos protótipos.

Ao longo do próximo capítulo, foram relatadas as etapas necessárias para o desenvolvimento do protótipo utilizado no projeto.

4 DESENVOLVIMENTO

A ideia proposta inicialmente, tida com base nas pesquisas realizadas em *autonomous platoon*, era a de se fazer o projeto baseado em um sistema fechado com base no seguimento de um trilho, ou seja, um autorama. Como os veículos se movimentariam de forma a não se preocupar com o percurso (no caso tendo como foco só a movimentação dos motores para frente e para trás), o uso dos sensores ultrassônicos se mostrou mais viável para a determinação das distâncias, pois possuía um melhor custo benefício em relação às outras opções que eram passíveis de implementação para essa tarefa (no caso a utilização de câmeras ou o uso do infravermelho). O sensor ultrassônico utilizado foi o HC-SR04.

Foi realizada uma tentativa de implementação desse sistema no autorama, porém ele se provou ineficaz, pois o trilho não conseguia fornecer a energia necessária para a alimentação dos componentes presentes nos veículos. Houveram problemas também neste momento com respeito à dinâmica dos veículos. As contas com respeito a centro de massa não haviam sido realizadas e isso resultou em um desequilíbrio dos veículos.

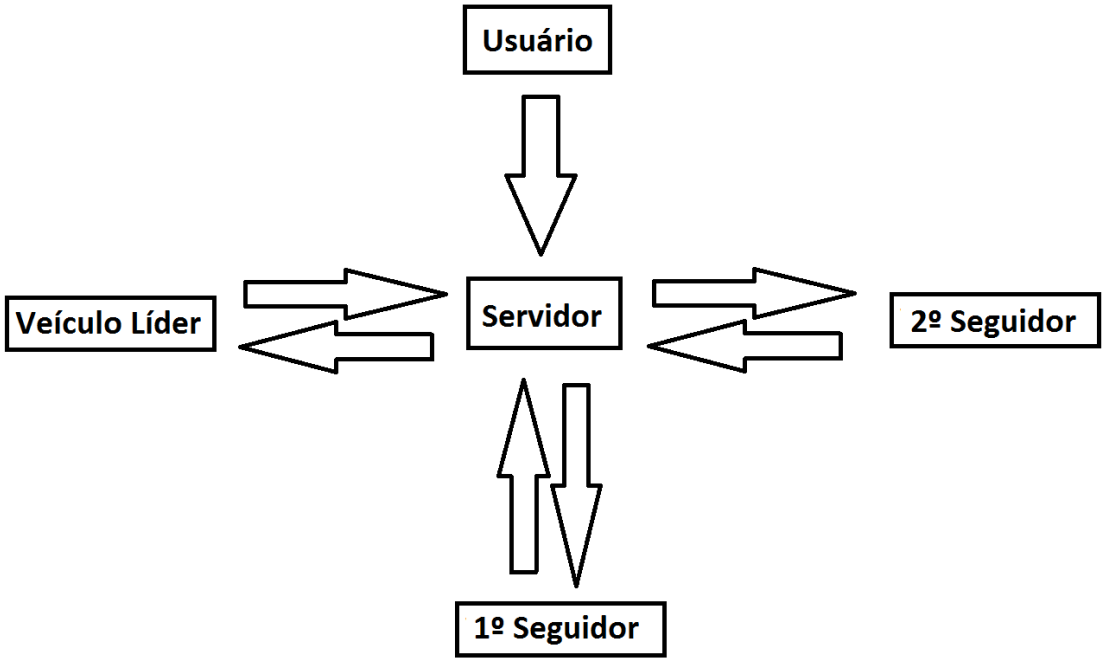
Utilizando como referência a tentativa de implementação no autorama, o modelo do projeto foi idealizado. O principal objetivo da proposta é a de se controlar a distância entre os veículos, sendo que estes ainda se movimentariam em um circuito fechado em suposição, embora não se fosse utilizar mais o mesmo. Com isso, o tipo de movimentação ainda seria limitada com o movimento para frente e para trás e a utilização de um sensor ultrassônico foi mantida.

Foi necessário realizar os dimensionamentos com relação à dinâmica veicular para que os veículos utilizados não sofressem com o problema de equilíbrio enfrentado na primeira tentativa. Definiu-se também que seria necessária a criação de um servidor para um controle de informações do comboio e, como neste projeto em específico não existe um motorista, para o controle do veículo líder.

O controle de informações seria feito por meio de um banco de dados. Portanto, a comunicação que deveria ser estabelecida no servidor está expressa na figura 18. Nela, nota-se que os veículos só tomam alguma decisão com base no que é informado pelo usuário ao servidor.

Os veículos solicitam constantemente ao servidor qual é a sua posição na cadeia de comando do comboio, utilizando a arquitetura V2N. Este retorna o último dado informado pelo usuário que está salvo no banco de dados. O líder também solicita constantemente ao banco em qual velocidade ele deve se locomover.

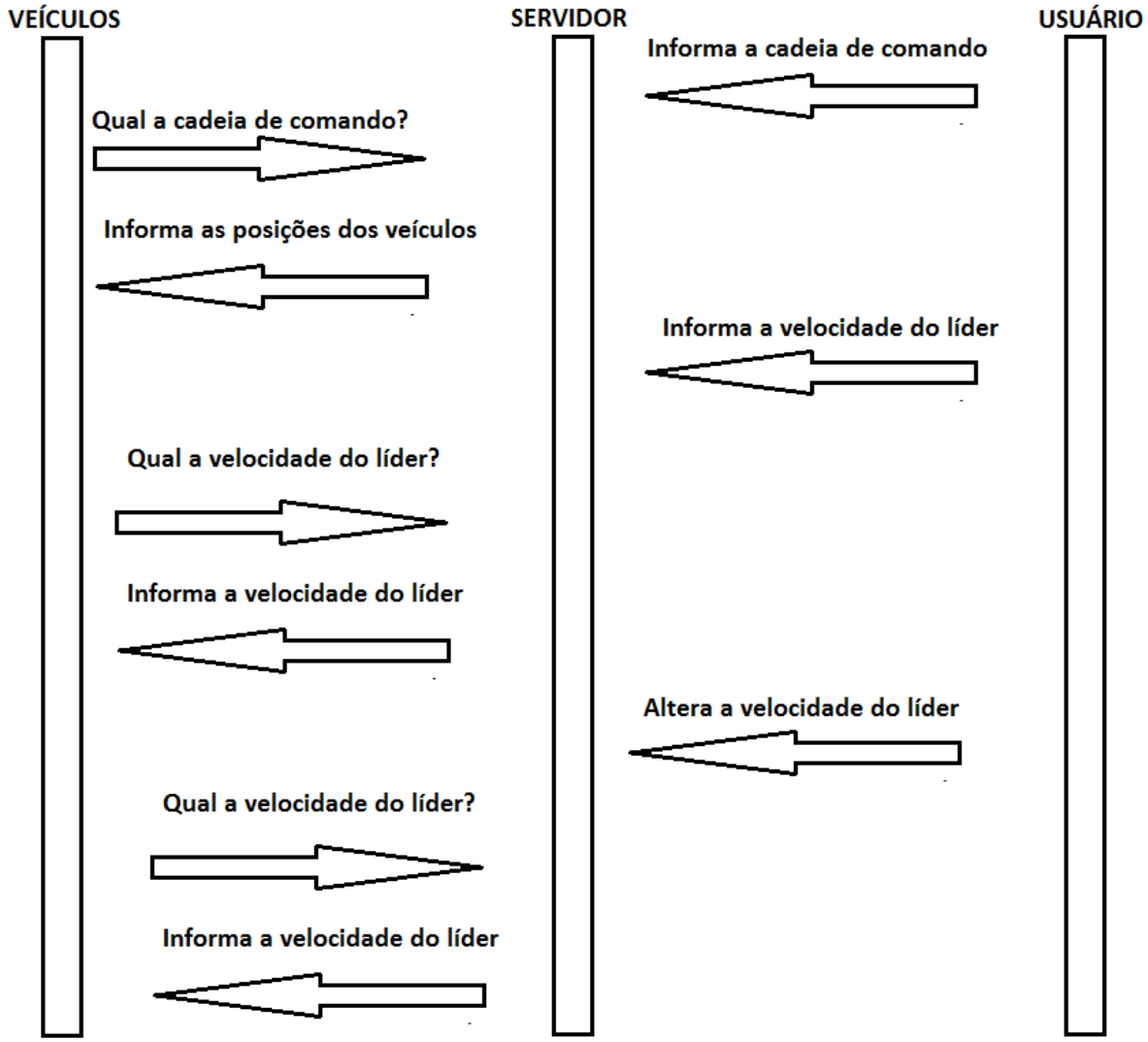
Figura 18: Diagrama de funcionamento do projeto



Fonte: Elaborado pelo autor

A comunicação feita entre o usuário, o servidor e os veículos está expressa na figura 19 por meio de uma carta de tempo.

Figura 19: Carta de tempo da comunicação servidor x usuário x veículos



Fonte: Elaborado pelo autor

Agora, são apresentadas as informações sobre o que foi feito no projeto e como isso influencia no seu funcionamento.

4.1 Teste de funcionamento da Ponte H

O seguinte roteiro de testes tem como objetivo testar os componentes para

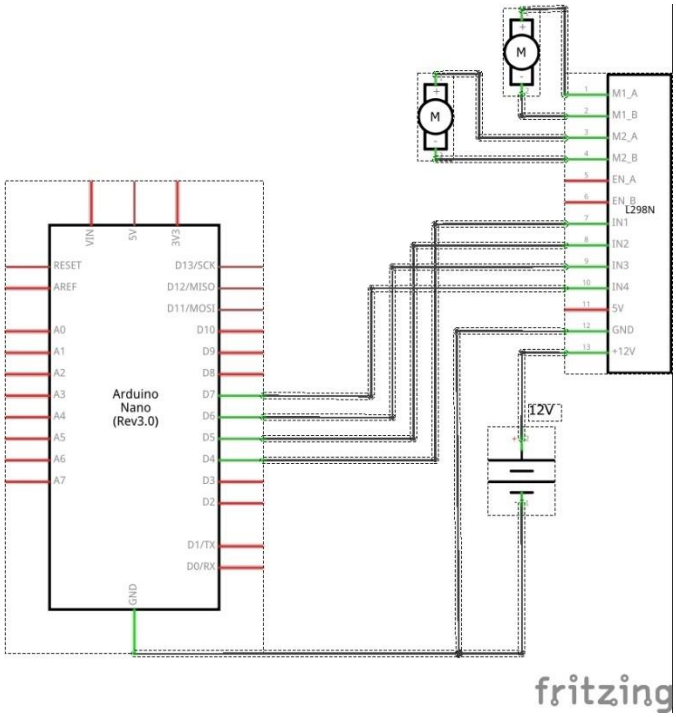
acionamento do motor (*Shield* Ponte H L298N) e códigos básicos para seu funcionamento.

4.1.1 Configurações iniciais

Para a alimentação do motor, foi utilizado o circuito de acionamento L298N. Segundo o *datasheet* do componente, o mesmo possui a opção de receber alimentação com 5V ou 12V. Foi escolhida a alimentação com 5V, pois assim ela pode ser conectada direto ao Arduino e o motor utilizado não requer tanta potência. O *shield* possui entradas para dois motores DC ou para um servo motor. Para o controle desses motores, é preciso modificar os pinos de entrada.

A figura 20 representa o circuito de ligação do circuito L298N ao Arduino.

Figura 20: Circuito de ligação para o teste



Fonte: Elaborado pelo autor

O circuito de ligação apresentado na figura 20 funciona com alimentações externas entre 6 e 35 Volts, se manter o *jumper* que ativa o regulador de tensão integrado a placa, que disponibiliza uma saída de 5V no pino (5V) para usos

externos. O quadro 2 representa como ocorre a ativação do motor A. Para o motor B, são utilizados os pinos IN3 e IN4.

Quadro 2: Esquema para ativação do motor A ou motor B

Motor	IN1	IN2
Horário	5V	GND
Anti-Horário	GND	5V
Ponto Morto	GND	GND
Freio	5V	5V

Adaptado de: https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/

Foram então realizadas todas as ligações, testando uma entrada de motor por vez. O código para utilizado para o teste está localizado no Anexo A.

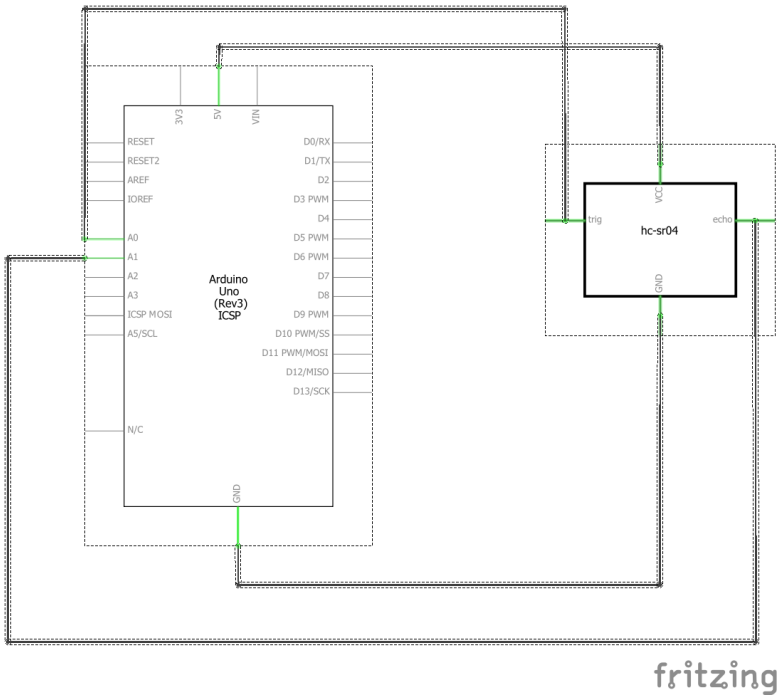
4.2 Teste de funcionamento do Sensor Ultrassônico (HC–SR04)

O seguinte roteiro de testes tem como objetivo validação de medições do sensor ultrassônico HC-SR04, para medição de distâncias e desenvolvimento de códigos básicos para seu funcionamento.

4.2.1 Configurações iniciais do teste com o sensor de ultrassom

Para os primeiros testes dos sensores ultrassônicos, além deles próprios, também foram utilizados um Arduino, uma *protoboard* e cabos para as conexões elétricas que estão na figura 21.

Figura 21: Esquema eletrônico entre Sensor Ultrassônico e Arduino



Fonte: Elaborado pelo autor

Primeiramente, foi necessário testar se todos os sensores HC-SR04 estavam recebendo pulsos elétricos e enviando uma resposta. Com a garantia de que todos os sensores funcionavam, desenvolveu-se um código para o Arduino para que ele realizasse 30 medidas seguidas, porém com intervalos de alguns segundos entre elas, e simultaneamente enviasse esses valores para o monitor serial da sua IDE de programação para que os valores pudessem ser lidos. Esse código encontra-se no Apêndice A.

4.2.2 Aquisição de medições do sensor de ultrassom

Por questão de conveniência, foi escolhida uma parede como ponto a ser medido, uma vez que essa é plana, que é um dos requisitos do sensor ultrassônico escolhido.

Em seguida, foi construída uma espécie de graduação, com o auxílio de uma trena, que continha as seguintes distâncias, todas em metros, do ponto a ser medido pelo sensor: 0,05; 0,50; 1,00; 1,50; 2,00; 2,50; 3,00; 3,50; 4,00; 4,50 e 5,00. É

possível observar essa régua na figura 22.

Figura 22: Distâncias e local utilizados para testes do sensor

Fonte: Elaborado pelo autor

Os valores, exceto 4,50 e 5,00 m, foram escolhidos por contemplarem os limites garantidos pelo *datasheet* do componente. Os valores de 4,50 e 5,00 m foram escolhidos para uma melhor visualização do futuro comportamento do projeto fora dos limites estabelecidos pelos fabricantes.

4.2.3 Validação das medições

Para cada distância estabelecida anteriormente para os três sensores, foram retiradas trinta amostras, pois, segundo o estudo conhecido como “distribuição de ‘*Student’* t”, trinta é o valor que divide pequenas amostras e grandes amostras, sendo assim um valor adequado para não correr riscos de erros estatísticos por ter uma amostra muito pequena.

Os valores encontrados nas medições de cada sensor para cada distância foram documentados e se encontram nos quadros 4 a 6 no item 5, assim como suas respectivas médias aritméticas e seus respectivos desvios padrões (indicados nas

tabelas por média e D. P.).

A média aritmética é uma maneira de conhecer o valor médio de uma amostra, ou seja, considerando todos os valores dessa amostra, demonstrar qual o número que todas elas juntas se aproximam.

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \ (Eq.2)$$

Onde:

- $\bar{X}$ é a média;
- $\sum_{i=1}^{n} X_i$ é a somatória de todos os valores desde X_1 até X_n;
- X_i é um elemento da amostra;
- n é o número de amostras.

Já o desvio padrão é uma maneira de expressar qual a distância entre os valores da amostra e a média levando em consideração todos os valores da amostra, ou seja, o desvio padrão indica a uniformidade dos dados de uma amostra.

$$DP = \sqrt{\frac{\sum_{i=1}^{n}(X_1 - \bar{X})^2}{n-1}} \ (Eq.3)$$

Onde:

- DP é o desvio padrão;
- $\bar{X}$ é a média;
- $\sum_{i=1}^{n} X_i$ é a somatória de todos os valores desde X_1 até X_n;
- X_i é um elemento da amostra;
- n é o número de amostras.

4.3 Teste de funcionamento do Módulo *Wi-Fi* (ESP8266)

O seguinte roteiro de testes tem como objetivo habilitar a utilização do módulo *Wi-Fi* ESP8266, permitindo receber comandos através de uma página na internet ou

em um aplicativo móvel.

4.3.1 Detecção de redes de internet

A primeira tarefa é fazer com que o módulo pesquise quais as redes de WiFi que existem dentro do raio de atuação para o seu funcionamento.

O código utilizado para os primeiros testes do módulo está localizado no anexo B e opera da seguinte maneira: a comunicação serial entre o computador e o módulo é habilitado em 9600 de *baudrate*, para permitir a supervisão do funcionamento do código. O modo de operação do módulo é alterado para estação e desconecta o mesmo caso já esteja conectado em alguma rede.

Após a configuração do dispositivo para a aplicação, o programa realiza a verificação de quantidade de redes dentro do raio de alcance do módulo, se não houver redes, é emitida no monitor serial que não houveram redes encontradas.

Caso uma ou mais redes sejam captadas, é feito uma lista com os nomes das respectivas redes descobertas, que são exibidas no monitor serial da IDE do Arduino utilizada.

4.3.2 Adquirindo dados

A segunda tarefa é fazer com que o módulo consiga acessar uma página na internet para pesquisar e retornar endereços completos partindo apenas do fornecimento do CEP desejado.

Junto da programação do Arduino são utilizados conhecimentos em linguagem em HTML para realizar a leitura da estrutura de uma página da internet de maneira simples. O código utilizado para este teste está documentado no anexo C.

O código realiza as seguintes operações: habilita a conexão serial entre o módulo e o computador, com uma velocidade de 9600 *baudrate* para que seja

permitido a supervisão do funcionamento do código. Estabelece uma conexão com uma rede de internet definida quando o programa é carregado no componente e determina que o modo de operação seja como uma estação.

Sempre é feita a verificação da conexão entre o módulo é o site requisitado, caso a conexão não seja concluída com êxito, um código de erro é gerado pelo servidor responsável pelo site; se a conexão for sucedida, o código 200 é retornado do servidor. Após a conexão ser estabelecida corretamente, o programa aguarda que o usuário digite o número do CEP desejado, acompanhado de um “;” (ponto e vírgula) ao final, para identificar que o comando chegou ao fim.

Esse código inserido é armazenado em uma variável, chamada temp; e essa variável é concatenada junto ao link utilizado para pesquisar estes dados. O módulo acessa essa URL, e utiliza o recurso *GET* para captar o código *HTML* da página acessada.

Tendo acesso ao código *HTML* da página, os dados como: nome da rua, CEP e UF (Unidade Federal) são separados e exibidos ao usuário, para indicar que a pesquisa foi bem-sucedida.

4.3.3 Aquisição e armazenamento de dados

A terceira tarefa é fazer com que o módulo permaneça fornecendo dados dos sensores para a formação de um banco de dados. O algoritmo utilizado para este teste está documentado no anexo D.

O código realiza as seguintes operações: habilita a conexão serial entre o módulo é o computador, com uma velocidade de 9600 *baudrate* para que seja permitido a supervisão do funcionamento do código. Estabelece uma conexão com uma rede de internet definida quando o programa é carregado no componente e determina que o modo de operação seja como uma estação.

Onde no primeiro momento existem duas instâncias que o módulo deve obedecer, onde ele inicia realizando a aquisição do valor da posição dele no comboio e depois realiza o *upload* dos dados de maneira geral para o banco de

dados. Sempre é feita a verificação da conexão entre o módulo é o site requisitado, caso a conexão não seja concluída com êxito, um código de erro é gerado pelo servidor responsável pelo site; se a conexão for sucedida, o código 200 é retornado do servidor.

Após a conexão ser estabelecida corretamente, o programa recebe o valor que foi inserido no banco de dados, como mostrado no programa do anexo D, para a variável pos3, que armazena a posição do veículo de número três.

O valor contido nessa variável é mostrado para o usuário pelo monitor serial da IDE. O módulo acessa essa URL, e utiliza o recurso *GET* para captar o código *HTML* da página acessada.

Na segunda instância, o módulo realiza a submissão dos dados para o servidor, de modo que os valores das variáveis de velocidade do primeiro ao terceiro integrante do comboio e após as variáveis de distâncias, respectivamente: vel, vel2, vel3, dis, dis2 e dis3. Para indicar que a ação foi bem-sucedida, só quando aparecer o código de conexão concluída com êxito, que seria o número 200.

No segundo momento o código sofreu modificações de modo que, com o valor que se encontra na variável pos3, ou na pos2 ou na pos1 do banco de dados, vai determinar em quais parâmetros submeter uma sequência de dados. Como por exemplo: se for determinado que o terceiro veículo deve ser o segundo veículo no comboio, o programa vai parar de submeter informações para a posição anterior em que o mesmo estava.

4.4 Dimensionamento e montagem da placa controladora

As placas controladoras foram dimensionadas de modo que os componentes necessários pudessem ser alocados e soldados de modo que a mesma fosse compacta.

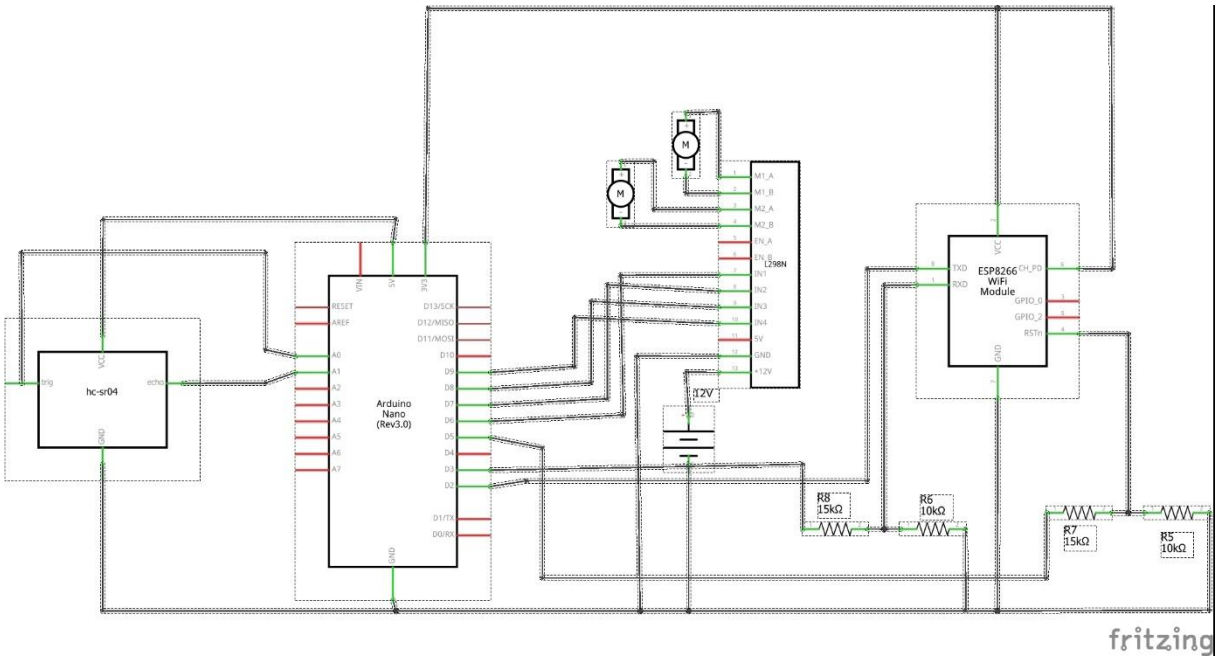
Para a construção das placas controladoras foram utilizados:

- 3 Arduinos Nano

- 3 Módulos de *Wi-Fi* – ESP8266-E01
- 3 soquetes de 8 pinos
- 3 Soquetes de 30 pinos
- 3 Soquetes de 40 pinos
- 6 resistores de 15k Ohm
- 6 resistores de 10k Ohm

Os componentes estão dispostos conforme representado abaixo a vista superior do esquema elétrico, na figura 23.

Figura 23: Esquema elétrico utilizado nas placas controladoras



Fonte: Elaborado pelo autor

O soquete com 30 pinos é para acomodação do Arduino ao circuito e interligar seus contatos com o soquete de 40 pinos. Permitindo a troca de pinos utilizados nos programas do Arduino para a utilização dos componentes, sem a necessidade de soldar diretamente nos contatos do sistema embarcado, e permite a conexão de mais componentes caso necessário.

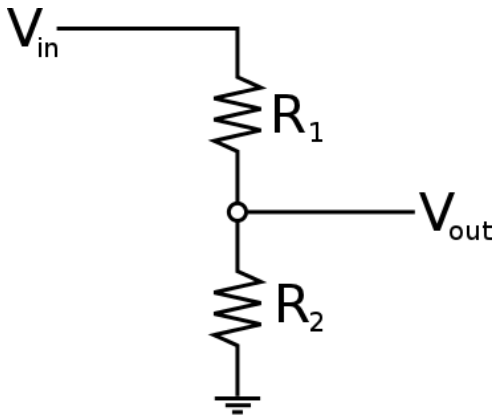
O soquete de 8 pinos é para acomodar e interligar o Módulo de *Wi-Fi* aos pinos das extremidades do soquete de 40 pinos, permitindo que o acesso aos contatos e suas conexões sejam realizadas de maneira mais eficientes.

Para realizar a troca de informações entre o Arduino e o Módulo de *Wi-Fi*, é

necessária a utilização de um divisor de tensão. Pois o Arduino transmite com 5 Volts em seus pinos de I/O, e o ESP8266 opera com tensões de 3,3 Volts.

O divisor de tensão é uma técnica que cria uma tensão elétrica (Vout) que seja proporcional à outra tensão (Vin) e segue o esquema de exemplo, representado na figura 24 a seguir.

Figura 24: Exemplo de ligação no divisor de tensão



Fonte: https://static.efetividade.net/img/20120307151629voltagedividerschematic-59536.png Acesso em 21 de jun de 2018.

Para dimensionar os resistores do circuito, foi utilizada a seguinte equação:

$$v_{out} = \frac{R_2}{R_1 + R_2} \cdot v_{in} \ (Eq.4)$$

Onde:

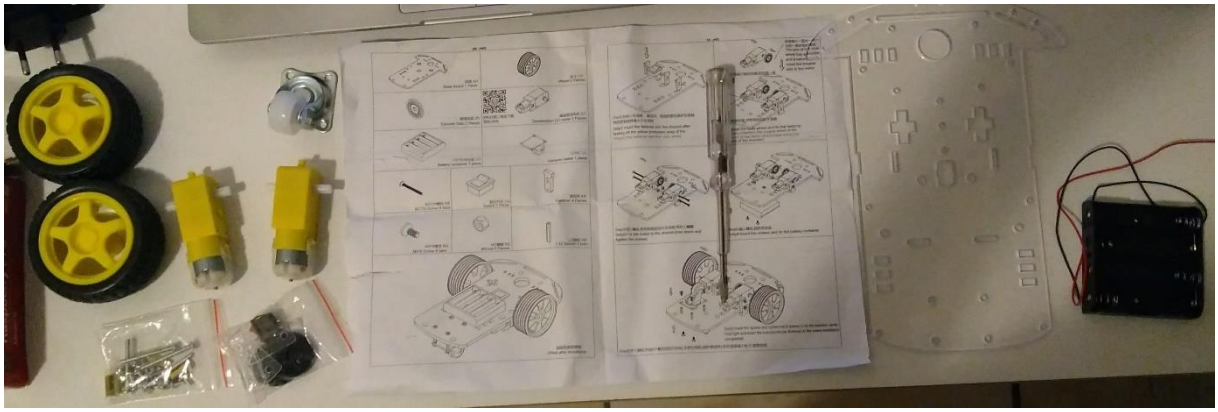
- $Vout =$ Tensão de saída desejada
- $Vin =$ *Tensão de entrada*
- $R1 =$ *Valor de resistência em Ohms*
- $R2 =$ *Valor de resistência em Ohms*

4.5 Montagem da plataforma

Os primeiros passos para montagem dos veículos foi seguir o manual de instruções de montagem que acompanhou o produto adquirido, para fixar os suportes dos motores, os motores e o rodízio giratório no chassi, passos registrados

nas figuras 25, 26, 27 e 28 a seguir.

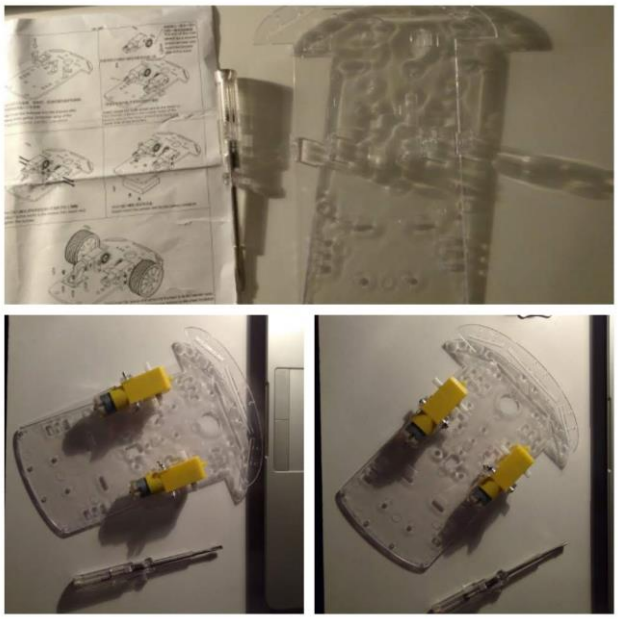
Figura 25: Peças necessárias para a montagem do chassi



Fonte: Elaborado pelo autor

Na figura 26, são apresentadas o posicionamento e a montagem dos motores na base do robô.

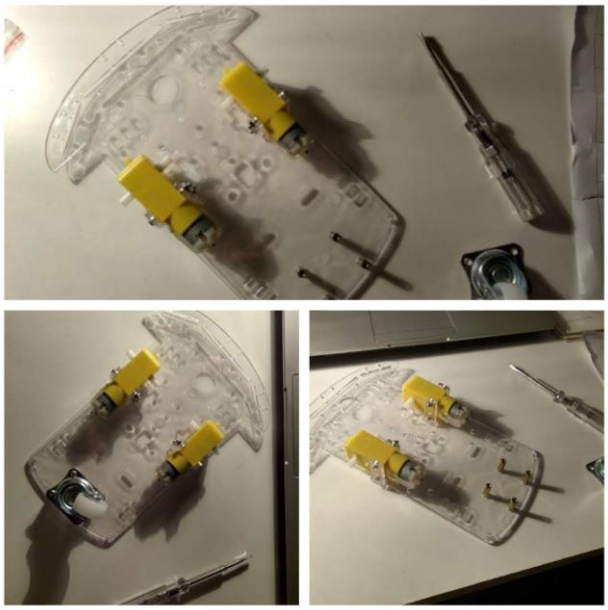
Figura 26: Montagem do motor no chassi



Fonte: Elaborado pelo autor

Na figura 27, o posicionamento do rodizio guia para o robô foi feita.

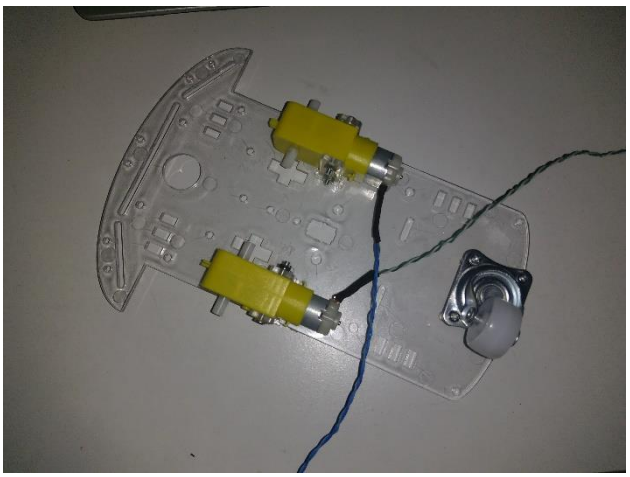
Figura 27: Montagem do rodízio giratório no chassi



Fonte: Elaborado pelo autor

A figura 28 ilustra o posicionamento dos motores já com sua ligação elétrica.

Figura 28: Fios para conexão do motor



Fonte: Elaborado pelo autor

As cotas dimensionais do chassi utilizado, estão disponíveis no Apêndice C. Para a inserção dos componentes fez-se necessário realizar o cálculo da distribuição de peso sobre o chassi. As dimensões e os pesos dos componentes utilizados estão no quadro 3 na próxima página.

Quadro 3: Dimensões e peso dos componentes utilizados

Componente	Largura (mm)	Comprimento (mm)	Altura (mm)	Peso (g)
Ponte H	43	43	27	30
HC-SR04	20	45	15	9
Placa Controladora	50	100	-	20
			Peso Total	59

Fonte: Elaborado pelo autor

De início, apenas o sensor tem posição pré-estabelecida, pois para realizar a leitura, o mesmo se encontra na frente do veículo.

Para determinar as posições dos demais componentes utilizamos o cálculo de Momentos no chassi dos veículos, de modo que sua resultante sempre seja zero.

A somatória de forças deve ser igual a zero, significando que as forças estão equilibradas e que a superfície de apoio suporta a força peso sem rupturas.

$$\sum F = 0 \ (Eq.5)$$

$$F_1 + F_2 + F_3 - Fc = 0 \ (Eq.6)$$

$$30 + 20 + 9 - Fc = 0 \ (Eq.7)$$

$$Fc = 59 \times 10^{-3} N \ (Eq.8)$$

Onde:

- F1 = Força peso da Ponte H [g]
- F2 = Força peso da Placa Controladora [g]
- F3 = Força peso do Sensor Ultrassônico [g]
- Fc = Força normal resultante do chassi [g]

Para realizar a distribuição dos componentes sobre o chassi dos veículos, realizamos uma análise estrutural de momentos fletores.

O momento fletor é a somatória dos momentos de força em relação igual a um ponto igual a zero, gerando tração e compressão.

Sempre calculando de dois em dois componentes. Utilizando as seguintes fórmulas para o eixo y:

$$\sum M_{y_1} = 0 \ (Eq.9)$$

$$F_1 \cdot y_1 = F_3 \cdot y_3 \ (Eq.10)$$

$$y_1 = \frac{F_3 \cdot y_3}{F_1} = \frac{9 \cdot 197{,}5}{30} = 59{,}25mm \ (Eq.11)$$

Onde:

- $F1$ = Força peso da Ponte H [g]
- $y1$ = Posição da Ponte H em relação à traseira do chassi [mm]
- $F3$ = Força peso do Sensor Ultrassônico [g]
- $y3$ = Posição do Sensor Ultrassônico em relação a traseira do chassi [mm]

$$\Sigma M y_2 = 0 \ (Eq.12)$$

$$F_3 \cdot y_3 = F_2 \cdot y_2 \ (Eq.13)$$

$$y_2 = \frac{F_3 \cdot y_3}{F_2} = \frac{9 \cdot 197{,}5}{20} = 88{,}875mm \ (Eq.14)$$

Onde:

- $F2$ = Força peso da Placa Controladora [g]
- $y2$ = Posição da Placa Controladora em relação à traseira do chassi [mm]
- $F3$ = Força peso do Sensor Ultrassônico [g]
- $y3$ = Posição do Sensor Ultrassônico em relação à traseira do chassi [mm]

Utilizando as seguintes fórmulas para o eixo x:

$$\sum M x_1 = 0 \ (Eq.15)$$

$$F_1 \cdot x_1 = F_3 \cdot x_3 \ (Eq.16)$$

$$x_1 = \frac{F_3 \cdot x_3}{F_1} = \frac{9 \cdot 50}{30} = 15mm \ (Eq.17)$$

Onde:

- $F1$ = Força peso da Ponte H [g]
- $x1$ = Posição da Ponte H em relação a lateral esquerda do chassi [mm]
- $F3$ = Força peso do Sensor Ultrassônico [g]
- $x3$ = Posição do Sensor Ultrassônico em relação a lateral esquerda do chassi [mm]

$$\sum M x_2 = 0 \qquad (Eq.18)$$

$$F_3 \cdot x_3 = F_2 \cdot x_2 \qquad (Eq.19)$$

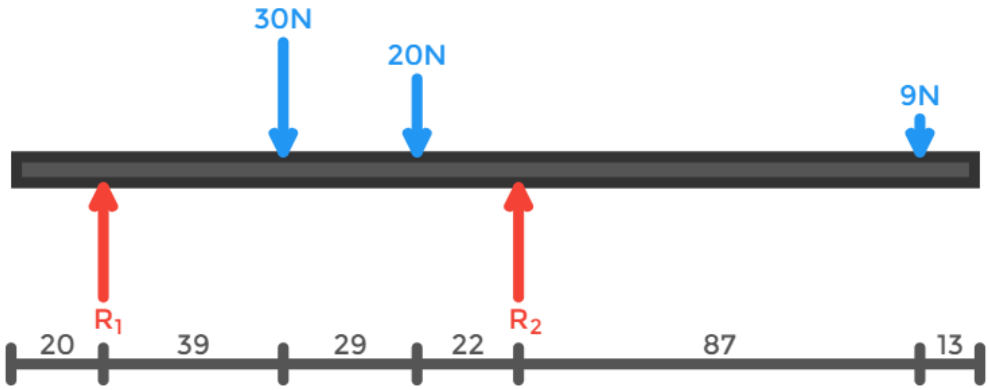
$$x_2 = \frac{F_3 \cdot x_3}{F_2} = \frac{9 \cdot 50}{20} = 22{,}5mm \qquad (Eq.20)$$

Onde:

- $F2$ = Força peso da Placa Controladora [g]
- $x2$ = Posição da Placa Controladora em relação a lateral esquerda do chassi [mm]
- $F3$ = Força peso do Sensor Ultrassônico [g]
- $x3$ = Posição do Sensor Ultrassônico em relação a lateral esquerda do chassi [mm]

Com o dimensionamento feito, é possível montar o diagrama de forças no chassi, ilustrados na figura 29 a seguir, onde as dimensões estão em milímetros.

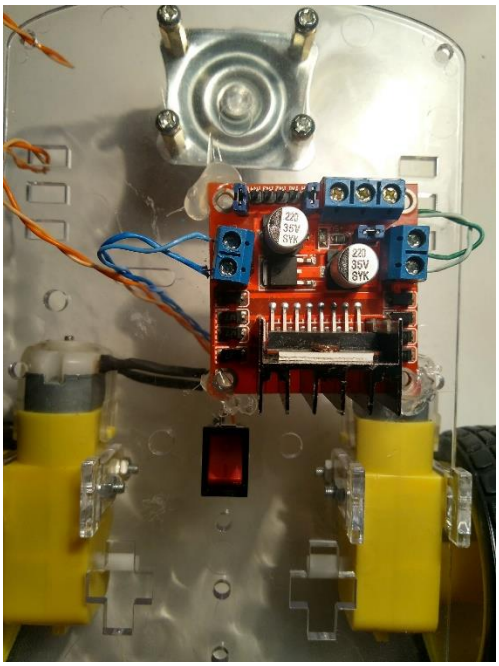
Figura 29: Diagrama de forças na estrutura dos veículos



Fonte: Elaborado pelo autor

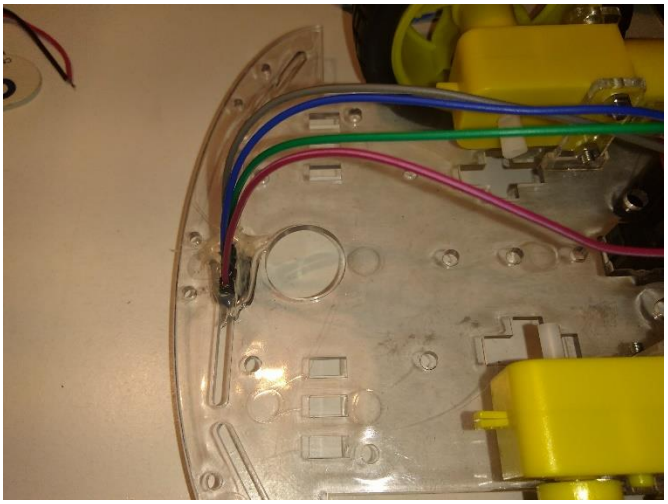
A fixação dos componentes nos veículos, ilustrados nas figuras 30 e 31 na próxima página, seguindo o dimensionamento para que a dinâmica veicular seja a mais adequada possível. Para fixar os componentes elétricos utilizamos cola quente de modo que caso seja necessário realizar modificações, os componentes sofrem fácil remodelamento.

Figura 30: Ponte H fixada na posição calculada



Fonte: Elaborado pelo autor

Figura 31: Fios de conexão do Sensor



Fonte: Elaborado pelo autor

4.6 Teste de plataforma controlando acionamento dos motores de acordo com a distância medida

Com os testes de componentes já realizados separadamente, se torna possível a integração dos mesmos para construir o sistema de controle do veículo.

Neste primeiro, o sensor ultrassônico, Arduino e *Shield* Ponte H foram integrados para realização dos testes simples da plataforma. Tendo limitações definidas como seguir apenas para frente e evitar colisão com objetos colocados, de maneira proposital, na frente dos veículos.

O código utilizado neste teste está no Apêndice B.

4.7 Construção do Servidor local

O papel do servidor para o Autonomous Platoon é muito importante, pois ele faz o controle de dados de todas as informações relevantes para que o mesmo possa estar sempre funcionando da melhor maneira. Neste projeto em especial, foi definido que o servidor deveria servir para definir a ordem de liderança dos veículos

que compões o comboio, para controlar o veículo líder e para salvar algumas informações dos demais veículos e apresentá-las ao usuário.

Foi decidido que o servidor seria feito de forma local nesse primeiro momento, tendo como proposta futura a implementação do mesmo de forma online. Conforme apontado pela redação do CANALTECH (SD), uma pesquisa feita em 2007 apontou o servidor web Apache como sendo responsável por 47,2% dos servidores ativos no mundo. Baseado nisso, foi decidido que o servidor local seria feito utilizando o Apache.

Após isso, uma pesquisa foi realizada sobre como poderia ser feito de forma a colaborar com um banco de dados. Nela, foi encontrado um pacote de código aberto chamado XAMPP. Conforme o próprio site do XAMPP explica, é um ambiente de desenvolvimento que visa principalmente possibilitar a utilização de: um servidor Apache, um banco de dados chamado MySQL, um servidor de FTP FileZilla, um servidor de e-mail Mercury e um servidor web Java chamado Tomcat.

O XAMPP tem suporte para as linguagens PHP, que foi utilizada no projeto tendo como versão a 5.2.0, e Perl e possui diversas outras aplicações de código aberto. Como esse pacote possibilita a utilização do servidor local no Windows e se mostrava funcional para o trabalho, foi decidido que o servidor seria montado com base nele (utilizando o servidor Apache e o banco de dados MySQL) e o desenvolvimento do servidor foi iniciado.

O MySQL, de acordo com Niederauer (2008), é um sistema gerenciador de bancos de dados com código-fonte aberto e que utiliza a linguagem padrão SQL. Ele possui diversas aplicações para a internet. O banco de dados é criado na forma de uma tabela.

A sua utilização no projeto é válida, pois se encaixa na estratégia adotada para a montagem do banco de dados, que no caso é a de concentrar cada uma das informações em uma determinada linha para que assim seja mais fácil selecionar uma delas para leitura ou para escrita.

Para criar o banco de dados foi necessário habilitar a função MySQL do pacote XAMPP. Com essa habilitação então foi possível acessar o phpMyAdmin, que é uma interface criada para administração de MySQL, e criar a estrutura do

banco de dados do projeto, a qual se deu o nome de “platoon”. Com a estrutura criada, foi possível configurar uma tabela para conter as informações pertinentes do projeto, que foi chamada de “positions”.

Essas informações, que consistem nas posições, distâncias relatadas pelos sensores e velocidades de cada veículo do comboio, são alocadas de forma a cada uma estar em uma linha diferente, conforme definido pela estratégia. A tabela possui três colunas: id (para definir o endereço de cada informação), nome e valor (que possui a informação que cada nome deve possuir).

Com o banco de dados devidamente configurado, o foco passou a ser o desenvolvimento das páginas do servidor e dos arquivos complementares do mesmo. Esse desenvolvimento se deu por meio da utilização de códigos em HTML, Javascript e PHP. Segundo Costa (2007), “o HTML é a linguagem padrão utilizada para o acesso e exibição de páginas web”. No projeto, foi ela que montou a parte estrutural e visual das páginas do servidor. Costa (2007) também explica que o “Javascript é uma linguagem de script ... que permite a criação de páginas interativas”.

Para o sevidor, ela foi usada para implementar o uso de funções que eram chamadas a partir do apertar de um botão. Conforme Niederauer (2011), a diferença da PHP para as outras linguagens é que ela pode interagir com o mundo web. Para o presente trabalho, ela foi utilizada para se estabelecer uma comunicação com o banco de dados e para manipular os dados contidos no mesmo.

Foi determinado que seriam necessárias quatro páginas de interação com o usuário:

- **Tela inicial:** É a tela base do servidor. Nela o usuário é direcionado para a tela de definição ou para a tela de controle do veículo líder;
- **Tela de definição:** Esta tela deve receber do usuário quais devem ser as posições de cada veículo;
- **Lista de posições:** Uma tela que exibe as informações que foram gravadas no banco de dados com respeito à ordem de liderança do comboio. Ela existe para que o usuário tenha certeza de que definiu de forma correta essa ordem;

- **Tela de controle:** A tela que visa controlar a velocidade do veículo líder. Ela apresenta ias informações que o controlador informa, ou seja, as distâncias relatadas pelos sensores dos veículos e as suas velocidades também.

Essa interação do usuário com o servidor e do servidor com os veículos é feita por HTML conforme estabelecido pela lógica da carta de tempo da figura 19.

O desenvolvimento iniciou-se pela tela inicial, apresentada no Apêndice D. Esta tela tem como título “SERVIDOR TCC” e apresenta o objetivo do projeto. Nela, há dois *links* para acessar as telas de definição e de controle.

Nesta tela também existe um código em PHP que visa garantir que os veículos não possam se mover caso o usuário não esteja na tela de controle. Isso garante segurança, pois se parte do pressuposto de que se o usuário saiu da tela de controle, é porque ele deseja redefinir a ordem de liderança. Para isso, os veículos devem estar parados. A tela inicial está representada pela figura 32.

Figura 32: Tela Inicial

Fonte: Elaborado pelo autor

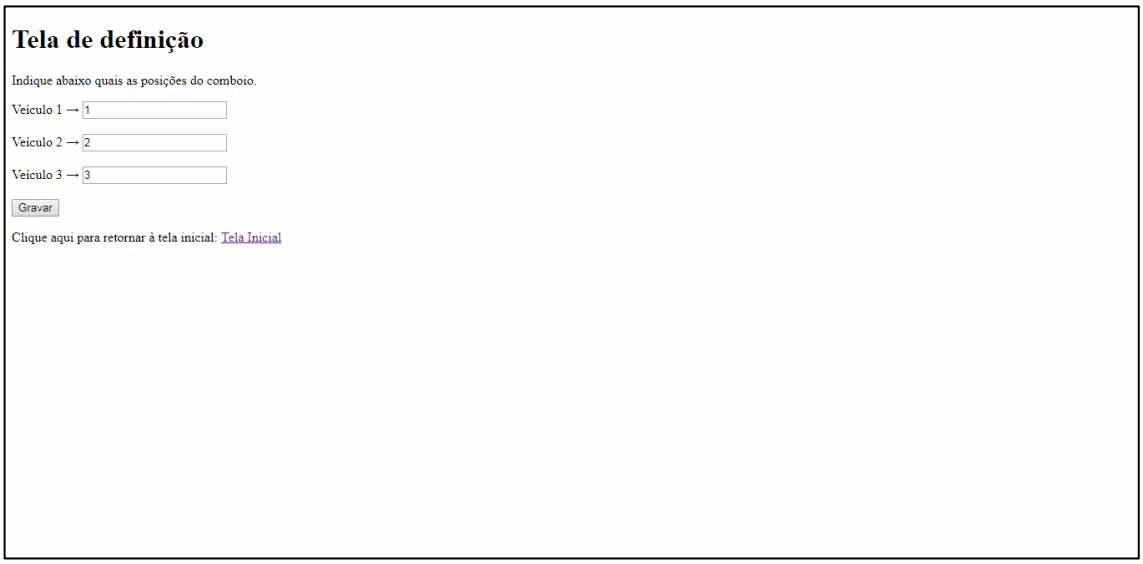
Após isso, a tela de definição foi idealizada por meio do código apresentado no Apêndice F. Aqui é possível definir as posições de cada veículo por informar cada uma em sua respectiva caixa de texto.

Cada caixa está devidamente identificada. Abaixo das caixas existe um botão chamado “Gravar”. Esse botão envia os dados das caixas de texto para outro

arquivo chamado “grava.php”, apresentado no Apêndice G. Este arquivo serve para verificar se as posições informadas estão de acordo com o que é necessário para se estabelecer a ordem de liderança.

Caso tudo esteja nos conformes, os dados são salvos no banco de dados e o servidor segue para a tela que exibe as posições ao usuário. Caso contrário, um alerta é exibido ao usuário informando o que precisa ser corrigido e abre novamente a tela de definição para que este o faça. A figura 33 demonstra um exemplo de como colocar as informações devidas na tela de definição.

Figura 33: Tela de Definição



Fonte: Elaborado pelo autor

A seguir, foi feito o código que corresponde à tela que apresenta a lista de posições, apresentado no Apêndice H. O objetivo principal dessa tela é permitir ao usuário ter certeza de que a ordem de liderança está correta e que o comboio já pode começar a rodar.

Ela apresenta as informações por meio de uma tabela que possui duas colunas: nome e posição. Se cada veículo está configurado corretamente, o usuário deve clicar no *link* da tela inicial para poder acessar a tela de controle do veículo líder. Se não, ele deve voltar à tela de definição e reconfigurar as posições. Na figura 34, é possível ver um exemplo desta tela baseado no que foi informado na figura 33.

Figura 34: Tela com a lista de posições

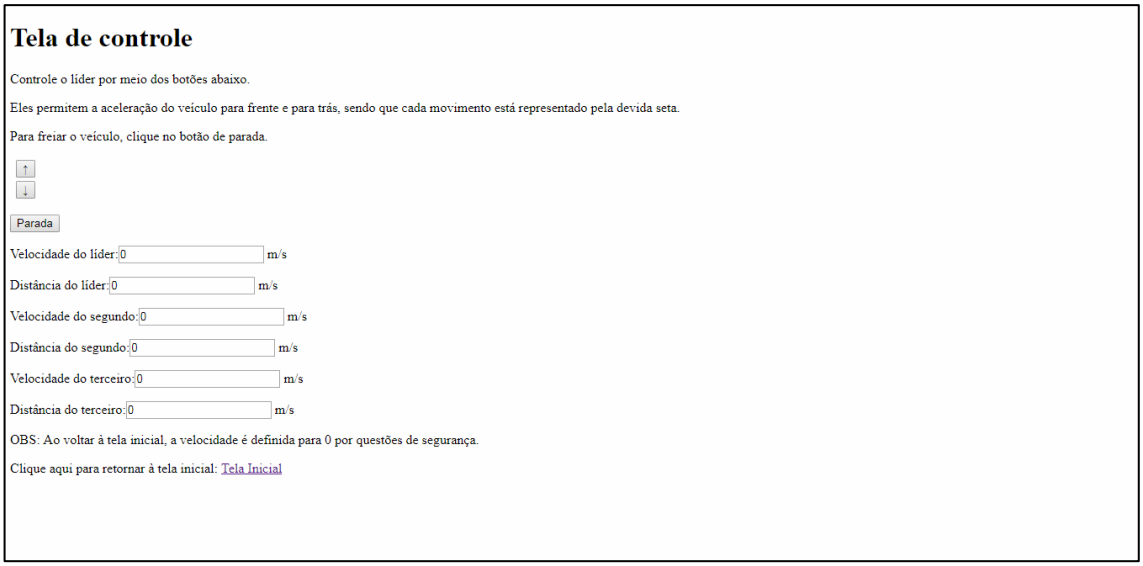
Fonte: Elaborado pelo autor

Por fim, a tela de controle foi feita por meio do código mostrado no Apêndice I. Nela, há as instruções básicas para se controlar veículo conduzido. Nela, estão os botões de controle, nos quais é possível aumentar a velocidade do veículo líder, diminuí-la ou pará-lo.

Cada botão está conectado uma determinada função, que abre um arquivo para cumprir com a solicitação do usuário. Por exemplo, ao clicar no botão que aumenta a velocidade, a função “frente” chama o arquivo “frente.php”, que está no Apêndice J. Esse arquivo irá incrementar a velocidade em 10 m/s, salvar o novo valor no banco de dados e retornar à página de controle.

O botão “re” chama o arquivo “re.php”, que está no apêndice K, e este reduz a velocidade em 10 m/s. O botão “parada”, por sua vez, abre o arquivo “parada.php”, que está no apêndice L, e este define a velocidade para 0 m/s. Essa tela também apresenta para o usuário, por meio de caixas de texto, as velocidades de todos os veículos e as distâncias medidas pelos sensores. Cada caixa está devidamente identificada com suas informações. A figura 35 apresenta a tela de controle, sendo que todas as informações no banco de dados estão zeradas.

Figura 35: Tela de Controle



Fonte: Elaborado pelo autor

Além dos arquivos já explicados, foram necessários alguns outros para que o projeto possa cumprir com os requisitos estabelecidos. Um código muito importante está no arquivo “conecta.php”, apresentado no Apêndice E. Ele serve para estabelecer a conexão com o banco de dados e é chamado por quase todos os outros arquivos.

O arquivo “salva.php”, apresentado no Apêndice Q, serve para salvar um dado no banco. Para isso, é necessário chama-lo pelo *link* “http://localhost/Servidor/salva.php” e informar uma variável para modificar. Por exemplo, caso se queira modificar a distância do veículo líder para 1 m (1000 mm), é necessário usar um *GET* no *link* “http://localhost/Servidor/salva.php?dis=1000” para que o valor seja modificado no banco.

No código, estão especificados os nomes de cada variável que deve ser acessada de acordo com o parâmetro a ser modificado. Caso o valor precise ser definido para 0, é necessário informar ao servidor da seguinte maneira: “0.”.

Os arquivos restantes são os de consulta, ou seja, fornecem valores que estão no banco de dados quando solicitado o acesso por meio de um *GET*. A seguir, estão apresentados os arquivos para cada informação:

- **“pos1.php” (Apêndice M):** Deve ser acessado para se saber a posição programada para o veículo 1;

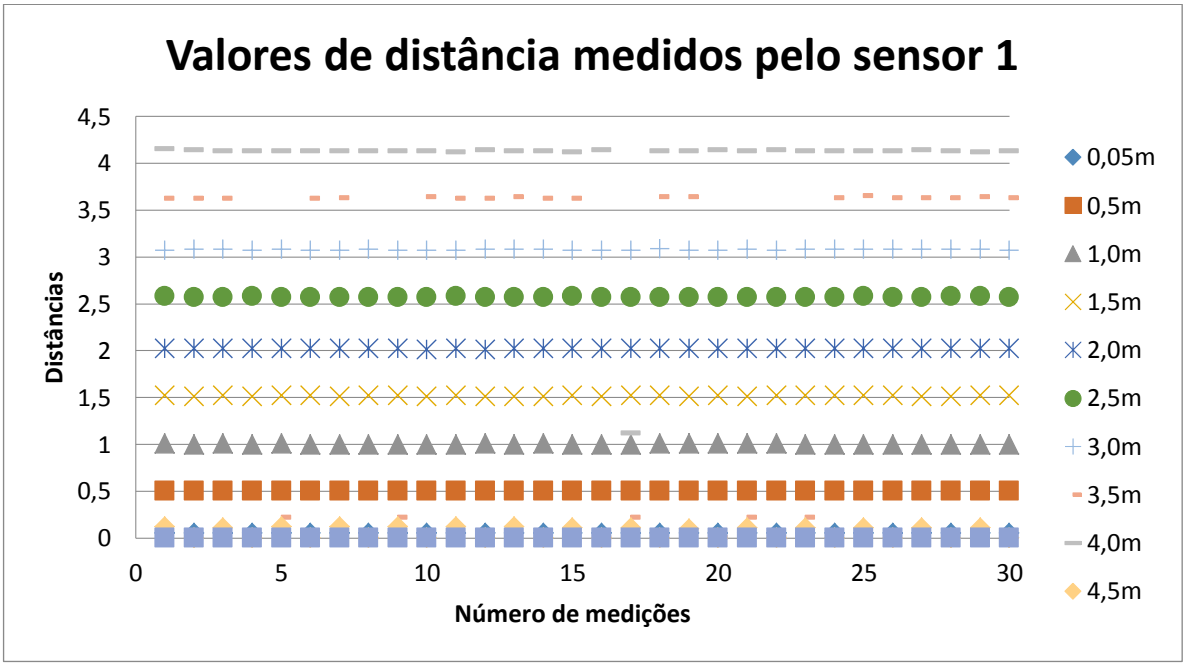
- **“pos2.php” (Apêndice N):** É acessado para se saber a posição programada para o veículo 2;
- **“pos3.php” (Apêndice O):** Informa a posição programada para o veículo 3;
- **“vel.php” (Apêndice P):** Fornece a velocidade do líder informada no banco.

5 RESULTADOS E DISCUSSÕES

Fazendo uma primeira observação dos dados nos três quadros, é notável que fora acima de 4 metros de distância o sensor realmente não se comporta de maneira eficiente. Também é possível concluir que o sensor número 2 tem problema para fazer medidas a partir dos 3 metros de distância.

A figura 36 apresenta um gráfico de tendências das medidas lidas e catalogadas no quadro 4.

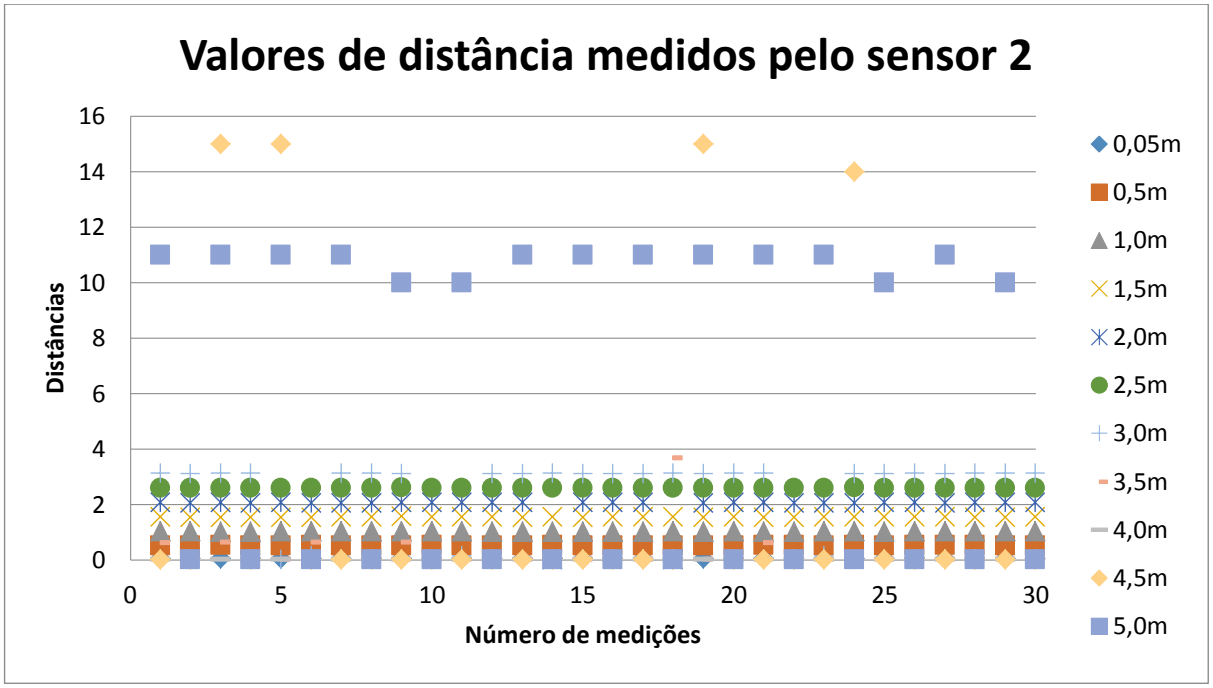
Figura 36: Gráfico das distâncias medidas pelo sensor 1



Fonte: Elaborado pelo autor

A figura 37 apresenta um gráfico de tendências das medidas lidas e catalogadas no quadro 5.

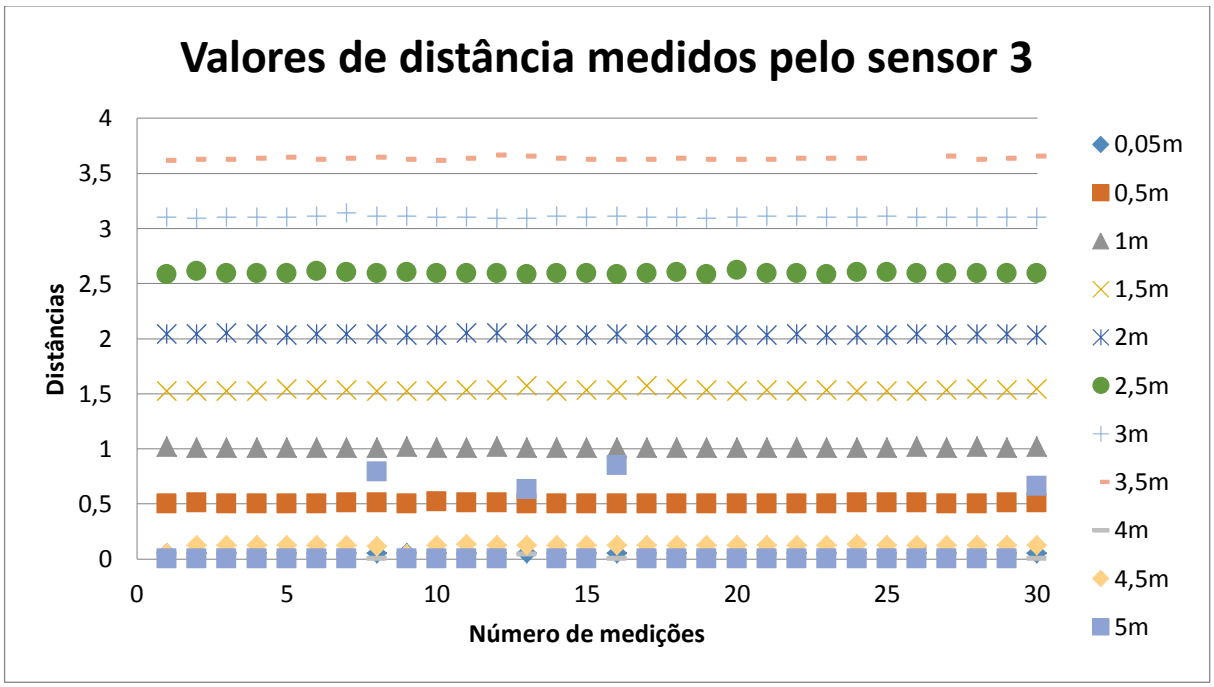
Figura 37: Gráfico das distâncias medidas pelo sensor 2



Fonte: Elaborado pelo autor

A figura 38 apresenta um gráfico de tendências das medidas lidas e catalogadas no quadro 6.

Figura 38: Gráfico das distâncias medidas pelo sensor 2



Fonte: Elaborado pelo autor

Quadro 4: Valores de distâncias medidas pelo sensor 1

Sensor 1	Distâncias										
Leituras	0,05m	0,5m	1,0m	1,5m	2,0m	2,5m	3,0m	3,5m	4,0m	4,5m	5,0m
1	0,05	0,5	1,01	1,52	2,02	2,58	3,07	3,62	4,15	0,12	0
2	0,05	0,5	1	1,51	2,02	2,57	3,08	3,62	4,14	0	0
3	0,05	0,5	1,01	1,52	2,02	2,57	3,08	3,62	4,13	0,11	0
4	0,05	0,5	1	1,51	2,02	2,58	3,07	0	4,13	0	0
5	0,05	0,5	1,01	1,52	2,02	2,57	3,08	0,22	4,13	0,12	0
6	0,05	0,5	1	1,52	2,02	2,57	3,07	3,62	4,13	0	0
7	0,05	0,5	1	1,51	2,02	2,57	3,07	3,63	4,13	0,12	0
8	0,05	0,5	1	1,52	2,02	2,57	3,08	0	4,13	0	0
9	0,05	0,5	1	1,52	2,02	2,57	3,07	0,22	4,13	0,12	0
10	0,05	0,5	1	1,51	2,01	2,57	3,07	3,64	4,13	0	0
11	0,05	0,5	1	1,52	2,02	2,58	3,07	3,62	4,12	0,12	0
12	0,05	0,5	1,01	1,51	2,01	2,57	3,08	3,62	4,14	0	0
13	0,05	0,5	1	1,51	2,02	2,57	3,08	3,64	4,13	0,12	0
14	0,05	0,5	1,01	1,51	2,02	2,57	3,08	3,62	4,13	0	0
15	0,05	0,5	1	1,52	2,02	2,58	3,07	3,62	4,12	0,11	0
16	0,05	0,5	1	1,51	2,02	2,57	3,07	0	4,14	0	0
17	0,05	0,5	1	1,52	2,02	2,57	3,07	0,22	1,12	0,11	0
18	0,05	0,5	1,01	1,52	2,02	2,57	3,09	3,64	4,13	0	0
19	0,05	0,5	1,01	1,51	2,02	2,57	3,07	3,64	4,13	0,1	0
20	0,05	0,5	1,01	1,52	2,02	2,57	3,07	0	4,14	0	0
21	0,05	0,5	1,01	1,51	2,02	2,57	3,08	0,22	4,13	0,11	0
22	0,05	0,5	1,01	1,52	2,02	2,57	3,07	0	4,14	0	0
23	0,05	0,5	1	1,52	2,02	2,57	3,08	0,22	4,13	0,11	0
24	0,05	0,5	1	1,52	2,02	2,57	3,08	3,63	4,13	0	0
25	0,05	0,5	1	1,52	2,02	2,58	3,08	3,65	4,13	0,11	0
26	0,05	0,5	1	1,52	2,02	2,57	3,08	3,63	4,13	0	0
27	0,05	0,5	1	1,51	2,02	2,57	3,08	3,63	4,14	0,11	0
28	0,05	0,5	1	1,51	2,02	2,58	3,08	3,63	4,13	0	0
29	0,05	0,5	1	1,52	2,02	2,58	3,08	3,64	4,12	0,11	0
30	0,05	0,5	1	1,52	2,02	2,57	3,07	3,63	4,13	0	0
Média	0,05	0,5	1,003333	1,516	2,019333	2,572333	3,075667	2,456333	4,031333	0,056667	0
D. P.	0	0	0,004877	0,005068	0,00258	0,004375	0,00578	1,717593	0,559302	0,058783	0

Fonte: Elaborado pelo autor

Quadro 5: Valores de distâncias medidas pelo sensor 2

Sensor 2	Distâncias										
Leituras	0,05m	0,5m	1,0m	1,5m	2,0m	2,5m	3,0m	3,5m	4,0m	4,5m	5,0m
1	0,05	0,51	1,02	1,54	2,06	2,58	3,12	0,61	0	0	11
2	0,05	0,51	1,02	1,53	2,05	2,58	3,11	0	0	0	0
3	0,05	0,52	1,01	1,53	2,07	2,59	3,12	0,62	0	15	11
4	0,05	0,51	1,01	1,53	2,05	2,59	3,12	0	0	0	0
5	0,05	0,51	1,03	1,53	2,06	2,59	0	0	0	15	11
6	0,05	0,52	1,02	1,53	2,05	2,59	0,14	0,63	0	0	0
7	0,05	0,51	1,02	1,54	2,05	2,58	3,12	0	0	0	11
8	0,05	0,51	1,02	1,54	2,05	2,58	3,13	0	0	0	0
9	0,05	0,51	1,02	1,56	2,06	2,61	3,1	0,63	0	0	10
10	0,05	0,52	1,03	1,55	2,06	2,58	0	0	0	0	0
11	0,05	0,51	1,02	1,54	2,07	2,58	0,14	0	0	0	10
12	0,05	0,51	1,01	1,54	2,07	2,58	3,11	0	0	0	0
13	0,05	0,51	1,01	1,53	2,07	2,58	3,11	0	0	0	11
14	0,05	0,52	1,01	1,54	0	2,58	3,12	0	0	0	0
15	0,05	0,51	1,01	1,53	2,06	2,58	3,11	0	0	0	11
16	0,05	0,51	1,01	1,55	2,06	2,58	3,11	0	0	0	0
17	0,05	0,51	1,01	1,54	2,06	2,58	3,11	0	0	0	11
18	0,05	0,51	1,02	1,54	0	2,58	3,12	3,66	0	0	0
19	0,05	0,51	1,01	1,53	2,05	2,58	3,1	0	0	15	11
20	0,05	0,51	1,02	1,54	2,06	2,59	3,12	0	0	0	0
21	0,05	0,52	1,01	1,53	2,07	2,58	3,12	0,6	0	0	11
22	0,05	0,52	1,01	1,53	2,05	2,59	0	0	0	0	0
23	0,05	0,51	1,02	1,55	2,05	2,58	0,14	0	0	0	11
24	0,05	0,52	1,02	1,54	2,05	2,61	3,1	0	0	14	0
25	0,05	0,51	1,01	1,54	2,07	2,59	3,11	0	0	0	10
26	0,05	0,52	1,02	1,54	2,06	2,58	3,12	0	0	0	0
27	0,05	0,52	1,02	1,54	2,05	2,58	3,11	0	0	0	11
28	0,05	0,52	1,02	1,53	2,07	2,59	3,12	0	0	0	0
29	0,05	0,51	1,02	1,53	2,06	2,58	3,13	0	0	0	10
30	0,05	0,51	1,02	1,54	2,06	2,59	3,12	0	0	0	0
Média	0,05	0,51	1,02	1,54	1,92	2,59	2,51	0,23	0	0,05	5,37
D. P.	0	0,004877	0,006168	0,007871	0,531353	0,00834	1,260439	0,701294	0	5,189512	5,56114

Fonte: Elaborado pelo autor

Quadro 6: Valores de distâncias medidas pelo sensor 3

Sensor 3	Distâncias (m)										
Leituras	0,05	0,5	1,0	1,5	2,0	2,5	3,0	3,5	4,0	4,5	5,0
1	0,05	0,5	1,02	1,52	2,04	2,58	3,1	3,61	0,04	0,05	0
2	0,05	0,51	1,01	1,52	2,04	2,61	3,09	3,62	0	0,12	0
3	0,05	0,5	1,01	1,52	2,05	2,59	3,1	3,62	0,04	0,12	0
4	0,05	0,5	1,01	1,52	2,04	2,59	3,1	3,63	0	0,12	0
5	0,05	0,5	1,01	1,54	2,03	2,59	3,1	3,64	0,04	0,12	0
6	0,05	0,5	1,01	1,53	2,04	2,61	3,11	3,62	0	0,12	0
7	0,05	0,51	1,01	1,53	2,04	2,6	3,14	3,63	0,04	0,12	0
8	0,05	0,51	1,01	1,52	2,04	2,59	3,11	3,64	0	0,11	0,79
9	0,05	0,5	1,02	1,52	2,03	2,6	3,11	3,62	0,04	0,04	0
10	0,05	0,52	1,01	1,52	2,03	2,59	3,1	3,61	0	0,12	0
11	0,05	0,51	1,01	1,53	2,05	2,59	3,1	3,63	0,04	0,13	0
12	0,05	0,51	1,02	1,53	2,05	2,59	3,09	3,66	0	0,12	0
13	0,05	0,5	1,01	1,57	2,04	2,58	3,09	3,65	0,04	0,12	0,63
14	0,05	0,5	1,01	1,52	2,03	2,59	3,11	3,63	0	0,12	0
15	0,05	0,5	1,01	1,53	2,03	2,59	3,1	3,62	0,04	0,12	0
16	0,05	0,5	1,01	1,53	2,04	2,58	3,11	3,62	0	0,12	0,85
17	0,05	0,5	1,01	1,57	2,03	2,59	3,1	3,62	0,04	0,12	0
18	0,05	0,5	1,01	1,54	2,03	2,6	3,1	3,63	0	0,12	0
19	0,05	0,5	1,01	1,53	2,03	2,58	3,09	3,62	0,04	0,12	0
20	0,05	0,5	1,01	1,52	2,03	2,62	3,1	3,62	0	0,12	0
21	0,05	0,5	1,01	1,53	2,03	2,59	3,11	3,62	0,04	0,12	0
22	0,05	0,5	1,01	1,52	2,04	2,59	3,11	3,63	0	0,12	0
23	0,05	0,5	1,01	1,53	2,03	2,58	3,1	3,63	0,04	0,12	0
24	0,05	0,51	1,01	1,52	2,03	2,6	3,1	3,63	0	0,13	0
25	0,05	0,51	1,01	1,52	2,03	2,6	3,11	0	0,04	0,12	0
26	0,05	0,51	1,02	1,52	2,04	2,59	3,1	0,1	0	0,12	0
27	0,05	0,5	1,01	1,53	2,03	2,59	3,1	3,65	0,04	0,12	0
28	0,05	0,5	1,02	1,54	2,04	2,59	3,1	3,62	0	0,12	0
29	0,05	0,51	1,01	1,53	2,04	2,59	3,1	3,63	0,04	0,12	0
30	0,05	0,51	1,02	1,54	2,03	2,59	3,1	3,65	0	0,12	0,66
Média	0,05	0,50	1,01	1,53	2,04	2,59	3,10	3,39	0,02	0,05	0,10
D. P.	0	0,005729	0,004138	0,013216	0,006862	0,009512	0,009606	0,92361	0,02069	0,019761	0,259855

Fonte: Elaborado pelo autor

Outra conclusão obtida é que muito provavelmente houve erros de medida na construção da régua base, pois todos os sensores estiveram um pouco fora do valor ideal, mas ainda as respostas obtidas foram satisfatórias levando em consideração as limitações de recursos para poder executar os testes.

Levando em consideração apenas as medidas entre 5 cm e 3 m, pôde-se concluir que as variações entre as medidas são pouco relevantes para a aplicação proposta.

O servidor se mostrou adequado para a aplicação. O banco de dados armazena de forma correta e retorna os valores corretos, quando solicitados. *Quando* esta é feita pelo ESP, por solicitar acesso ao arquivo previamente definido para a determinada informação que este está buscando, recebe um dado no formato *Json* com essa informação.

Os testes iniciais do servidor foram realizados utilizando as funções *GET* e *POST* de um aplicativo chamado *Postman*. Com o *GET* ocorria uma solicitação de página e com o *POST*, uma solicitação para gravar uma informação. Os resultados obtidos destes testes foram satisfatórios, já que o banco de dados fornecia as informações ou tinha o valor de alguma delas alterado de forma correta.

As comunicações dos veículos foram iniciadas com o módulo de *Wi-Fi* ESP8266-E01, onde nos primeiros momentos foram enfrentados problemas para habilitação e utilização do mesmo. Com isso, o manual do fabricante do módulo foi consultado para verificação de códigos base, com o objetivo de averiguar as condições do mesmo. Isso ajudou a identificar as redes de internet que estavam ao alcance do módulo e a executar a conexão com redes específicas para adquirir os códigos-fonte *HTML* de determinadas páginas *web*. Com isso, surgiu um novo problema, pois o módulo não fornecia a resposta esperada e só reportava erros de aquisição.

A solução encontrada foi a de trocar o módulo para uma versão alternativa que possui uma interface de programação de melhor compreensão. Os testes de aquisição de dados de endereços partindo da inserção do valor do CEP foram efetuados com êxito. Até o momento em que essa monografia foi redigida, os testes para troca de dados entre o módulo de internet e o Arduino não foram realizados

com êxito.

As respostas obtidas com o ESP8266 12E estão disponibilizadas a seguir.

Resposta do programa documentado no anexo B:

```
scan start
scan done
3 networks found
1: Templo do Nigga (-30)*
2: VIvo Fibra (-86)*
3: Zezo_Nira_2 (-88)*
```

É possível observar a quantidade e quais são os nomes das redes encontradas nessa busca realizada pelo módulo.

Resposta do programa documentado no anexo C:

```
{
  "cep": "03136-040",
  "logradouro": "Rua Falchi Gianini",
  "complemento": "",
  "bairro": "Vila Prudente",
  "localidade": "SÃ£o Paulo",
  "uf": "SP",
  "unidade": "",
  "ibge": "3550308",
  "gia": "1004"
}
```

Com a pesquisa realizada, é possível ver os dados pertinentes ao número de CEP digitado pelo usuário.

Resposta do programa documentado no anexo D:

```
[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200

[HTTP] begin...
[HTTP] GET...
[HTTP] GET... code: 200
"1"
```

O primeiro “[HTTP] GET ... code: 200” indica que os dados foram enviados para o servidor com êxito.

Após o código troca de instância, após o segundo “[HTTP] GET ... code: 200”, no monitor serial é exibido o valor da posição determinada pelo controle, através do servidor. O valor “1” com as aspas, é o valor recebido do servidor de maneira “bruta”.

Resposta do programa documentado no apêndice R:

```
[HTTP] GET...
[HTTP] GET... code: 200
1


"1"
[HTTP] GET...
[HTTP] GET... code: 200
<br />
<b>Notice</b>:  Undefined index: dis2 in
<b>C:\xampp\htdocs\Servidor\salva.php</b> on line <b>14</b><br />
<br />
<b>Notice</b>:  Undefined index: dis3 in
<b>C:\xampp\htdocs\Servidor\salva.php</b> on line <b>17</b><br />
<br />
<b>Notice</b>:  Undefined index: vel2 in
<b>C:\xampp\htdocs\Servidor\salva.php</b> on line <b>20</b><br />
<br />
<b>Notice</b>:  Undefined index: vel3 in
<b>C:\xampp\htdocs\Servidor\salva.php</b> on line <b>23</b><br />
```

O primeiro número que aparece após o “[HTTP] GET ... code: 200” é o valor da posição determinada pelo controle, através do servidor. A seguir o valor “1” com as aspas, é o valor recebido do servidor de maneira “bruta”.

Após o código troca de instância, após os dados serem enviados para o servidor, no monitor serial são exibidas essas notificações, alegando que os valores das outras variáveis não foram submetidos, mas isso não representa um problema pois os valores restantes serão submetidos por outros módulos inseridos nos outros veículos.

Ao final de tudo o que foi possível desenvolver, o protótipo acabou não sendo finalizado graças ao problema de envio de dados do ESP para o servidor, onde ao tentar enviar os valores obtidos da leitura de um potenciômetro, da maneira que esse valor simula a variação da velocidade de um dos veículos, um erro do protocolo de comunicação foi exibido no monitor serial.

6 CONCLUSÃO

O desenvolvimento do presente trabalho possibilitou uma análise de como um software pode ser desenvolvido para garantir a segurança no controle de um comboio automotivo e de se os componentes pesquisados e, posteriormente, escolhidos se adequaram aos requisitos da proposta inicial.

Com respeito à mecânica do veículo, a disposição dos equipamentos necessários para o funcionamento atende aos requisitos para que a dinâmica do veículo esteja em equilíbrio. É possível afirmar isso pois o centro de massa do veículo está correto e os momentos fletores não são diferentes de 0 (zero).

O microcontrolador escolhido (Arduino Nano) atendeu parcialmente ao que se era esperado que ele fizesse. Ele recebe a leitura do meio escolhido para se medir a distância entre os veículos e a interpreta. A sua segunda função, que seria enviar esses dados lidos e receber comandos do servidor, não pôde ser testada porque houve problemas de comunicação que estão relatados abaixo na justificativa do módulo *Wi-Fi* contida neste capítulo. Por isso, não é possível afirmar que o Arduino Nano pode cumprir plenamente as suas funções estimadas para o projeto.

Segundo os testes gerais realizados com respeito a ponte H (L298), esta cumpre com os requisitos básicos estabelecidos para a mesma. Ela aciona os motores de forma eficiente.

O sensor ultrassônico atende às necessidades estabelecidas para a prototipagem. Ele realiza a leitura de forma eficaz para as distâncias trabalhadas no projeto. Contudo, o ângulo de abertura do sinal emitido limitou a movimentação dos veículos, pois não é possível delimitar os trajetos de conversão, ou seja, as curvas. Para a implementação em veículos de porte apropriado para transporte de cargas, ele não é a melhor opção, pois o seu alcance de leitura é limitado assim como a sua abertura de sinal.

Para realizar trabalhos com o ESP8266-E01 são necessários que circuitos de alimentação consigam garantir um fornecimento estável de energia para que seu funcionamento e acionamento sejam feitos. A aquisição de códigos por comandos AT cumpre com a sua função enquanto o módulo funciona como um servidor.

Funcionando como cliente a situação muda, pois a aquisição de dados é um pouco mais dificultosa por falta de comandos.

O ESP8266-12E possui uma interface de programação de mais fácil entendimento, já que a placa do componente tem quase a mesma quantidade de pinos que um Arduino Nano e o chip já é integrado a placa. Isso é de ajuda porque o módulo de comunicação já possui as ligações elétricas necessárias para seu funcionamento. Assim, a utilização do módulo tem sua eficiência aumentada.

Com essas informações em mente, é seguro concluir que o ESP8266-12E atende às exigências do projeto por possibilitar a troca do Arduino Nano pela placa do próprio componente. O ESP8266-E01 também atende às necessidades, mas seria necessário dedicar um tempo maior para que a aplicação pudesse ser desenvolvida em sua plenitude.

O meio escolhido para a construção do servidor, um banco de dados em MySQL e um servidor Apache, atendeu aos requisitos necessários do presente projeto. Ele cria uma interface intuitiva e o banco de dados troca informações conforme a proposta inicial estabelecida para ele. Porém, caso sejam necessários arquivos mais compactos em questão de número de linhas de código ou em questão do próprio espaço de armazenamento, é mais interessante buscar outro meio de guardar e acessar os dados que possa atender a esses novos requisitos.

Apesar de os meios apresentados serem adequados para o funcionamento do projeto, o mesmo não havia sido finalizado ainda no momento em que esta monografia foi redigida. Isso ocorreu, pois seria necessária uma maior quantidade de tempo para ser dedicada ao estudo da comunicação e, além disso, para a implementação desta rede ao projeto.

Ao final do projeto, foram constatadas algumas possíveis melhorias do protótipo. Estas foram definidas como propostas futuras e são:

- **Finalização do projeto:** terminar o estudo a respeito da comunicação e aplicá-lo para que o projeto possa ser testado de forma completa;
- **Aquisição de dados dos motores:** verificar os dados de funcionamento dos motores e catalogar esses dados;
- **Melhorias do servidor:** melhorar a parte visual por meio da implementação de um arquivo *CSS* e pode ser feita uma alteração na forma de armazenamento de dados;

- **Atualização do servidor:** fornecer ao servidor a possibilidade de hospedagem *online*;
- **Melhorias de protótipo:** adaptar o projeto para que ele possa realizar curvas e para que ele se adeque a novas estruturas de veículos;
- **Implementação do sistema em veículos reais:** realizar um estudo das leis que envolvem o comboio autônomo no momento em que esta proposta for considerada, já que no momento atual não existem tais leis. Também será necessário adaptar o projeto para que este possa atender plenamente as necessidades de um veículo de proporções normais.

Referências

Alecrim, E. (17 de Janeiro de 2017). *O que é Internet das Coisas (Internet of Things)?* Acesso em 4 de Outubro de 2017, disponível em Info Wester: https://www.infowester.com/iot.php

Alves, R. d., Campbell, I. d., Couto, R. d., Campista, M. E., Moraes, I. M., Rubinstein, M. G., . . . Abdalla, M. (2009). Redes Veiculares: Princípios, Aplicações e Desafios. Em *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2009* (pp. 199 - 254). Pernambuco. Acesso em 13 de Outubro de 2017, disponível em https://www.gta.ufrj.br/ftp/gta/TechReports/ACC09.pdf

Arduino. (SD). *ARDUINO NANO*. Acesso em 16 de Setembro de 2017, disponível em Arduino: https://store.arduino.cc/usa/arduino-nano

Barbieri, N. (1995). Análise Dinâmica de Veículos Automotivos com Suspensões Passivas e Ativas. *Tecnologia e Humanismo*, 20-28. Acesso em 1 de Novembro de 2017

Bistafa, S. R. (2011). *Acústica Aplicada ao Controle do Ruído.* São Paulo: Edgar Blucher Ltda. Acesso em 25 de Novembro de 2017

Braga, N. C. (SD). *Como funcionam as UARTs (TEL006)*. Acesso em 20 de Novembro de 2017, disponível em Instituto NCB: http://newtoncbraga.com.br/index.php/telecom-artigos/1709-tel006.html

Brasil. (1 de Maio de 1943). Decreto-Lei N.º 5.452, de 1º de maio de 1943. *Aprova a Consolidação das Leis do Trabalho.* Acesso em 15 de Novembro de 2017, disponível em http://www.planalto.gov.br/ccivil_03/Decreto-Lei/Del5452.htm#tituloiiicapituloisecaoiva

Brasil. (2 de Março de 2015). Lei Nº 13.103, de 2 de março de 2015. *Dispõe sobre o exercício da profissão de motorista*. Acesso em 15 de Novembro de 2017, disponível em http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13103.htm

Cabral, J. B., & Murphy, C. C. (2012). *Fuel Cell: O Meio-Ambiente e o Carro.* Cassará. Acesso em 04 de Novembro de 2017, disponível em http://www.h2brasil.com/parte-1.php

Campos, A. (2 de Novembro de 2015). *ESP8266: Comandos AT*. Acesso em 18 de Dezembro de 2017, disponível em BR-Arduino.org: https://br-arduino.org/2015/11/esp8266-comandos-at.html

Cavalcante, K. G. (SD). *Som, Infrassom e Ultrassom*. Acesso em 21 de Setembro de 2017, disponível em Mundo Educação: http://mundoeducacao.bol.uol.com.br/fisica/som-infrassom-ultrassom.htm

Choi, Y., Kang, D., Lee, S., & Kim, Y. (2009). The Autonomous Platoon Driving System of the On Line Electric Vehicle. *Fukuoka International Congress Center* (pp. 3423-3426). Japão: ICROS-SICE International Joint Conference. Acesso em 13 de Setembro de 2017

Comer, D. E. (2015). *Interligação de Redes com TCP/IP: Protocolos, protótipos e arquitetura.* (Vol. 1). São Paulo: Elsevier Editora Ltda. Acesso em 20 de Novembro de 2017

Contet, J. M., Gruer, P., Koukam, A., & Gechter, F. (2007). Application of reactive multiagent system to linear vehicle platoon. *19th IEEE International Conference on Tools with Artificial Intelligence* (pp. 67-70). Belfort, France: University of Technology of Belfort-Montbeliard (UTBM). Acesso em 4 de Outubro de 2017

Corradi, W., Társia, R. D., Oliveira, W. S., Vieira, S. L., Nemes, M. C., & Balzuweit, K. (2010). *FUNDAMENTOS DE FÍSICA I.* Belo Horizonte: Editora UFMG. Acesso em 17 de Dezembro de 2017

Costa, C. J. (2007). *Desenvolvimento para Web.* Lisboa, Portugal: ITML press / Lusocredito. Acesso em 08 de junho de 2018, disponível em https://books.google.com.br/books?id=Jn6dTDF-wcsC&printsec=frontcover&hl=pt-BR&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Curvello, A. (29 de Abril de 2015). *Apresentando o módulo ESP8266*. Acesso em 21 de Setembro de 2017, disponível em Embarcados: https://www.embarcados.com.br/modulo-esp8266/

Daquino, F. (26 de Junho de 2014). *Na boleia: caminhões autônomos circularão por estradas antes do imaginado.* Acesso em 25 de Maio de 2017, disponível em TECMUNDO: https://www.tecmundo.com.br/caminhao/58318-boleia- caminhoes-autonomos-circularao-estradas- imaginado.htm

Eckel, T. (24 de Fevereiro de 2017). *NewPing Library for Arduino*. Acesso em 25 de Maio de 2017, disponível em Arduino: http://playground.arduino.cc/Code/NewPing

Evans, M., Noble, J., & Hochebaum., J. (2012). *Arduino em Ação.* São Paulo: Novatec Editora Ltda. Acesso em 15 de Novembro de 2017

Fullin, E. (26 de outubro de 2015). *Por falar em caminhão antigo*. Acesso em 16 de Dezembro de 2017, disponível em Planeta Caminhão - Pra quem gosta: https://planetacaminhao.com.br/por-falar-em-caminhao-antigo/

GitHub. (23 de Novembro de 2016). *ITEADLIB_Arduino_WeeESP8266*. Acesso em 16 de Novembro de 2017, disponível em GitHub: https://github.com/itead/ITEADLIB_Arduino_WeeESP8266

Guerra, P. H. (30 de Março de 2017). *Distribuição de peso: aspecto importante da dinâmica veícular*. Acesso em 14 de Outubro de 2017, disponível em Educação Automotiva: https://educacaoautomotiva.com/2017/03/30/distribuicao-de-peso/

Gugik, G. (13 de Março de 2009). *Código Aberto e Software Livre não significam a mesma coisa!* Acesso em 4 de Outubro de 2017, disponível em TECMUNDO: https://www.tecmundo.com.br/linux/1739-codigo-aberto-e-software-livre-nao-significam-a-mesma-coisa-.htm

Inc., E. S. (18 de Novembro de 2013). *Microsoft Word - ESP8266_Specifications_v4.0.* Acesso em 16 de Setembro de 2017, disponível em Espressif Systems: https://www.mikrocontroller.net/attachment/231858/0A-ESP8266_Specifications_v4.pdf

Inc., G. (SD). *Meet your Google Assistant.* Acesso em 17 de Setembro de 2017, disponível em Google Assistant: https://assistant.google.com/

Jaynes, N. (5 de Abril de 2016). *Three autonomous Mercedes-Benz trucks just platooned across Europe*. Acesso em 13 de Setembro de 2017, disponível em Mashable: http://mashable.com/2016/04/05/mercedes-actros-platoon-europe/#01uxyIM7viqI

Karzel, D., Marginean, H., & Tran, T.-S. (7 de Dezembro de 2016). *Uma arquitetura de referência para a Internet das Coisas - Parte 1*. (T. Lima, Editor) Acesso em 16 de Novembro de 2017, disponível em InfoQ: https://www.infoq.com/br/articles/internet-of-things-reference-

architecture

Leal, L. d., Rosa, E. d., & Nicolazzi, L. (2012). *Uma introdução à modelagem quase-estática de automóveis.* Florianópolis: Publicação interna do GRANTE Departamento de Engenharia Mecânica da UFSC.

Momote, V. (19 de Maio de 2016). *Modelos de comunicação para IoT*. Acesso em 16 de Novembro de 2017, disponível em Embarcados: https://www.embarcados.com.br/modelos-de-comunicacao-para-iot/

Nakatani, A. M., Guimarães, A. V., & Neto, V. M. (2013). MEDIÇÃO COM SENSOR ULTRASSÔNICO HC-SR04. *3º Congresso Internacional de Metrologia Mecânica.* Gramado. Acesso em 16 de Dezembro de 2017

Niederauer, J. (2008). *Integrando PHP 5 com MySQL.* São Paulo, Brasil: Novatec Editora Ltda. Acesso em 08 de junho de 2018, disponível em https://s3.amazonaws.com/academia.edu.documents/34869982/php.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1531151005&Signature=1Vzm2DaqYg3MZJSmtxm4XP4kpM4%3D&response-content-disposition=inline%3B%20filename%3DGuia_de_Consulta_Rapida.pdf

Niederauer, J. (2011). *Desenvolvendo Websites com PHP* (2ª ed.). São Paulo, Brasil: Novatec Editora Ltda. Acesso em 08 de junho de 2018, disponível em http://www.martinsfontespaulista.com.br/anexos/produtos/capitulos/650595.pdf

Pereira, T. A., & Machado, L. d. (2008). Um Módulo de Rede para a Construção de Aplicações Médicas Colaborativas. *X Symposium on Virtual and Augmented Reality (SVR2008)*, pp. 99-102. Acesso em 20 de Novembro de 2017, disponível em https://s3.amazonaws.com/academia.edu.documents/31913040/2008_svr1.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1511180868&Signature=YUsrTliRRL%2FEZsQPoiSL54Kd0YY%3D&response-content-disposition=inline%3B%20filename%3DUm_Modulo_de_Rede_para_a_Construca

Pissardini, R. d., Wei, D. C., & Júnior, E. S. (2013). Veículos Autônomos: Conceitos, Histórico e Estado-da-Arte. *XXVII Congresso de Pesquisa e Ensino em Transportes.* Belém do Pará. Acesso em 1 de Novembro de 2017, disponível em https://www.researchgate.net/publication/318897282_VEICULOS_AUTONOMOS_CONCEITOS_HISTORICO_E_ESTADO-DA-ARTE

ROBOCORE. (SD). *Módulo WiFi - ESP8266*. Acesso em 16 de Setembro de 2017, disponível em ROBOCORE Tecnologia LTDA.: https://www.robocore.net/loja/produtos/modulo-wifi-esp8266.html#descricao

Santos, D. d. (2012). Coisas básicas para se fazer com um Arduino (ou Arduíno). *Conferência Latino-Americana de Software Livre, IX.* Paraná: Portal EBC. Acesso em 15 de Novembro de 2017

Seguras, P. V. (18 de Fevereiro de 2017). *Estatísticas de acidentes de trânsito*. Fonte: Por Vias Seguras: http://www.vias-seguras.com/os_acidentes/estatisticas

Silva, D. C. (SD). *Ultrassom*. Acesso em 14 de Setembro de 2017, disponível em Brasil Escola: http://brasilescola.uol.com.br/fisica/ultrassom.htm

Silveira, C. B. (8 de Outubro de 2017). *O Que é Indústria 4.0 e Como Ela Vai Impactar o Mundo*.

Acesso em 12 de Outubro de 2017, disponível em Citisystems: https://www.citisystems.com.br/industria-4-0/

Soares, R., Galeno, S., & Soares, A. (2015). *Simulação de Redes Veiculares.* (R. S. Moura, H. S. Araújo, & J. V. Júnior, Eds.) Parnaíba. Acesso em 13 de Outubro de 2017

Souza, J. S. (SD). *Ecolocalização*. Acesso em 21 de Setembro de 2017, disponível em InfoEscola: https://www.infoescola.com/biologia/ecolocalizacao/

Spinola, A. D. (2010). *Modelagem da Dinâmica Veicular.* Rio de Janeiro: PUC-Rio – Certificação Digital Nº0410305/CA.

Teixeira, F. C., Oliveira, M. C., & Helleno, A. L. (28 de Fevereiro de 2014). Telemetria Automotiva via Internet Móvel. *Revista Ciência e Tecnologia, 16*. Acesso em 20 de Novembro de 2017, disponível em http://www.revista.unisal.br/sj/index.php/123/article/view/264

Anexo A – Código utilizado para acionamento básico dos motores

```
//Autor : FILIPEFLOP

//Definicoes pinos Arduino ligados a entrada da Ponte H
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;

void setup()
{
  //Define os pinos como saida
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}

void loop()
{
 //Gira o Motor A no sentido horario
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
 delay(2000);
 //Para o motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);
 //Gira o Motor B no sentido horario
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 delay(2000);
 //Para o motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);

 //Gira o Motor A no sentido anti-horario
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
 delay(2000);
 //Para o motor A
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, HIGH);
 delay(500);
 //Gira o Motor B no sentido anti-horario
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, HIGH);
 delay(2000);
 //Para o motor B
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, HIGH);
 delay(500);
}
```

Anexo B – Código utilizado para detecção de redes de WiFi

```
/*
 *  This sketch demonstrates how to scan WiFi networks.
 *  The API is almost the same as with the WiFi Shield library,
 *  the most obvious difference being the different file you need to
include:
 */
#include "ESP8266WiFi.h"

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

  // Set WiFi to station mode and disconnect from an AP if it was
previously connected
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);

  Serial.println("Setup done");
}

void loop() {
  Serial.println("scan start");

  // WiFi.scanNetworks will return the number of networks found
  int n = WiFi.scanNetworks();
  Serial.println("scan done");
  if (n == 0)
    Serial.println("no networks found");
  else
  {
    Serial.print(n);
    Serial.println(" networks found");
    for (int i = 0; i < n; ++i)
    {
      // Print SSID and RSSI for each network found
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (");
      Serial.print(WiFi.RSSI(i));
      Serial.print(")");
      Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE)?" ":"*");
      delay(10);
    }
  }
  Serial.println("");

  // Wait a bit before scanning again
  delay(5000);
}
```

Anexo C – Código utilizado para aquisição de endereços a partir do número do CEP

```
//Autor: Murilo Zanini de Carvalho

#include <Arduino.h>

#include <ArduinoJson.h>

#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>

#include <ESP8266HTTPClient.h>

#define USE_SERIAL Serial

ESP8266WiFiMulti WiFiMulti;
int i;
char temp;

void TrataMsg();

//Cria um buffer para trabalhar com as msg em JSON
StaticJsonBuffer<200> jsonBuffer;

void setup() {

    USE_SERIAL.begin(9600);
    USE_SERIAL.println("Inicializando comunicacao, aguarde:");
    //Delay para garantir que a comunicacao foi estabelecida

    USE_SERIAL.println("Sistema pronto para uso:");
    //Seleciona o mode de funcionamento do ESP como uma estacao que se
conecta a um AcessPoint
    WiFi.mode(WIFI_STA);
    //Inicializa a comunicacao com o AcessPoint
    //ID da rede e senha
    WiFiMulti.addAP("Moto G Play 8438", "ricardo2205");

}
//Variavel para guardar a mensagem recebida pela requisicao do GET
String msg = "";
void loop() {
    //Espera o usuario realizar o envio do CEP que ele quer consultar
    if(Serial.available()>0){
      temp = (char) Serial.read();
      if(temp == ';'){
        //msg += '\0';
        TrataMsg();
      }
      else
        msg += temp;

    }
}

void TrataMsg(){
  //Funcao que realiza a requisicao HTTP
  // wait for WiFi connection
    if((WiFiMulti.run() == WL_CONNECTED)) {
```

```
        //Cria um cliente Http para realizar a requisicao
        HTTPClient http;
        //Estabelece a conexao com o servidor
        http.begin("http://viacep.com.br/ws/" + msg+ "/json/"); //HTTP

        // Comeca a conexão e inicia o leitor de http
        int httpCode = http.GET();

        // Se httpCode for maior que zero
        if(httpCode > 0) {
            //Resposta quando a requisicao deu certo
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if(httpCode == HTTP_CODE_OK) {

                String payload = http.getString();

                //Toda a informacao esta nessa string
                USE_SERIAL.println(payload);

                //Recebe o pacote JSON
                //Cria um buffer para trabalhar com JSON
                JsonObject& recebe = jsonBuffer.parseObject(payload);
                //Verifica se o JSON e valido
                if(recebe.success()){
                  USE_SERIAL.print("CEP:");
                  const char* txt = recebe["cep"];
                  USE_SERIAL.println(txt);
                  USE_SERIAL.print("UF:");
                  const char* txt2 = recebe["uf"];
                  USE_SERIAL.println(txt2);
                }
                else{
                  USE_SERIAL.println("ERRO AO CONVERTER O JSON");
                }
            }
        } else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
        }

        //Fecha a conexao HTTP
        http.end();


    } else {
        USE_SERIAL.println("ERRO AO CONECTAR NA REDE!");
    }
    //Limpa a string
        msg = "";
}
```

Anexo D - Código utilizado para testes iniciais com o ESP8266 para obter e mandar dados para o servidor

```
//Autor: Murilo Zanini de Carvalho
#include <Arduino.h>

#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>

#include <ESP8266HTTPClient.h>



#define USE_SERIAL Serial

ESP8266WiFiMulti WiFiMulti;

Int estado=1;

void setup() {

  USE_SERIAL.begin(9600);

  USE_SERIAL.println();
  USE_SERIAL.println();
  USE_SERIAL.println();

  for (uint8_t t = 4; t > 0; t--) {
    USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
    USE_SERIAL.flush();
    delay(1000);
  }

  WiFi.mode(WIFI_STA);
  // WiFiMulti.addAP("FSA-SALA20", "sl20@igv24*");
  // WiFiMulti.addAP("FSA-TERREO", "fatec@1234+");
  // WiFiMulti.addAP("FSA-PV1", "fatec@1234+");
  // WiFiMulti.addAP("FSA-PV2", "fatec@1234+");
  WiFiMulti.addAP("Moto G Play 8438", "ricardo2205");

}

void loop() {
  while (1) {
    switch (estado) {
      case 0: //ENVIA DADOS PARA O SERVIDOR
        // Aguarda pela conexao com o WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          USE_SERIAL.print("[HTTP] begin...\n");
          // URL que sera acessada no Servidor
http.begin("http://192.168.1.34/Servidor/salva.php?vel=10&vel2=11&vel3=12&d
is=40&dis2=14&dis3=15");
          USE_SERIAL.print("[HTTP] GET...\n");
          // Comeca a conexao e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero
```

```
          if (httpCode > 0) {
            // Leitor HTTP e enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informacao esta nessa string
              USE_SERIAL.println(payload);
            }
          } else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
        estado++;
        delay(10000);
        break;

      case 1: //RECEBE DADOS DO SERVIDOR
        // Aguarda pela conexão WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          USE_SERIAL.print("[HTTP] begin...\n");
          // URL que sera acessada no Servidor
          http.begin("http://192.168.1.34/TCC/Servidor/pos3.php"); //HTTP
          USE_SERIAL.print("[HTTP] GET...\n");
          // Comeca a conexao e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero
          if (httpCode > 0) {
            // Leitor HTTP e enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informacao esta nessa string
              USE_SERIAL.println(payload);
            }
          } else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
        estado--;
        delay(10000);
        break;
    }
  }
}
```

Apêndice A – Código utilizado para realizar medições com os Sensores Ultrassônicos

```
#include <NewPing.h> //Biblioteca que utiliza recursos para utilizaÃ§Ã£o do
Sensor HC-SR04
#include <string.h> //Biblioteca que permite a interpretaÃ§Ã£o de frases

//Variáveis Globais
char msgIn [20];
byte i;

#define TRIGGER_PIN A0 // Pino Trigger
#define ECHO_PIN A1 // Pino Echo
#define MAX_DISTANCE 1000 // Máxima distância em cm. (Máximo que ele mede
entre 400-500cm).
#define MIN_DISTANCE 0.01 //Mínima distância em cm. (Mínimo que ele mede a
partir de 2 cm)

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // Setup da biblioteca
int distance = 0; //Variável que armazena a distância do objeto a ser lido

void setup()
{
  Serial.begin(115200); // Abre o monitor serial em 115200 bits por segundo
(baud) para ver o resultado da leitura
  pinMode (13, OUTPUT); //Habilita o pino 13 como saída para comandar um
led
}

void loop()
{

  if (Serial.available()) //Verifica se tem mensagem na entrada serial
  {

    msgIn[i] = char (Serial.read());//Recebe dado da porta serial

    //Compara para checar fim da mensagem
    if (msgIn[i] == ';') //Se o último caractere for igual ao ";" ele
habilita a comparação da mensagem recebida
    { msgIn[i + 1] = '\0'; //Adiciona um elemento vazio ao final da
mensagem para reconhecimento interno do programa
      i = 0; //Reinicia o índice de leitura
      if (strcmp(msgIn, "CHECK;") == 0) //Se o comando recebido for igual
"CHECK;" o programa vai executar um ping e verificar a distância do objeto
      {
        for (int i=0; i<30; i++)
        {
        unsigned int uS = sonar.ping();
        // Manda um ping, recebe o tempo em microssegundos (uS).
        distance = uS / US_ROUNDTRIP_CM; //Converte o tempo em distância

        Serial.print("Distancia:");
        Serial.print(distance);
        Serial.println("cm");

        delay (2000);

      }
```

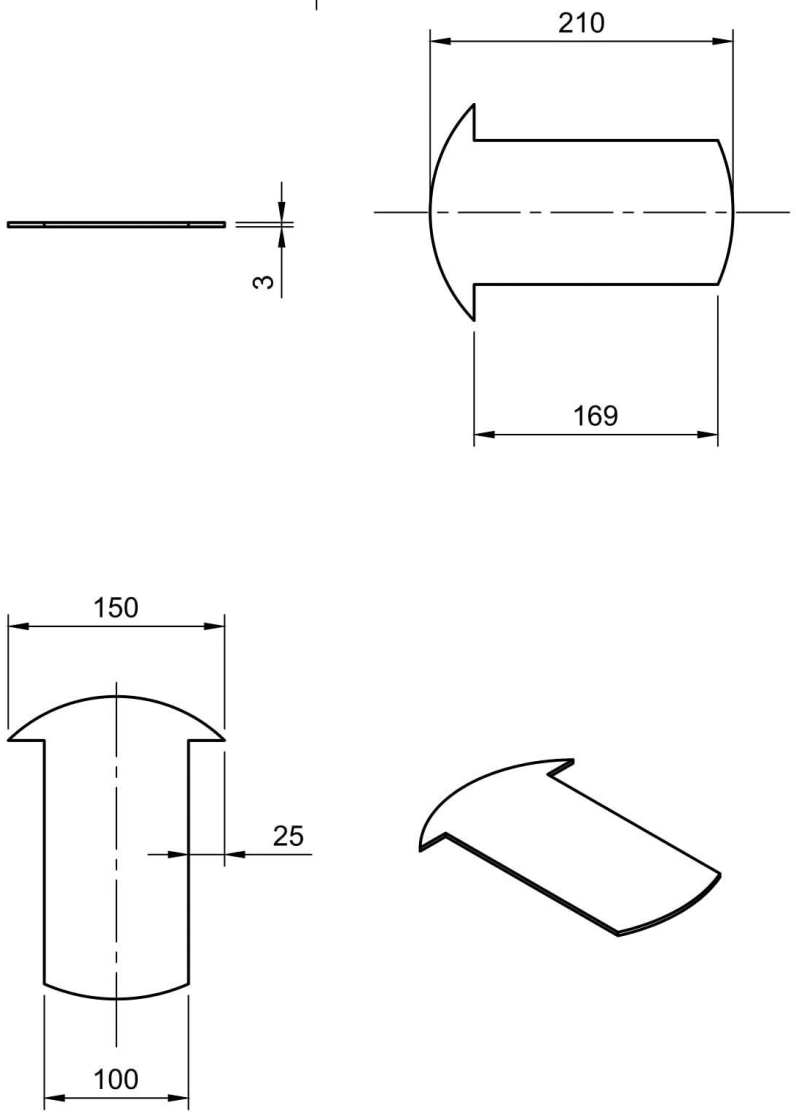
```
        }


      }
      else {
          //Prepara para receber próximo char
          i++;
        }
    }
}
```

Apêndice B – Programação utilizada para realizar testes simples nas plataformas

```
#include <NewPing.h> //Biblioteca que utiliza recursos para utilização do
Sensor HC-SR04
#include <SoftwareSerial.h>

//Definicoes pinos Arduino ligados ao Sensor
#define TRIGGER_PIN A0 // Pino Trigger
#define ECHO_PIN A1 // Pino Echo
#define MAX_DISTANCE 200 // Máxima distancia em cm.(Máximo que ele mede é
entre 400-500cm).
#define MIN_DISTANCE 10 //Mínima distancia que o carro deve manter do
carrinho ou obstáculo à frente

//Definicoes pinos Arduino ligados ao ESP8266
#define RST 5
SoftwareSerial esp8266(2, 3); //RX pino 2 e TX pino 3

#define DEBUG true

//Definicoes pinos Arduino ligados a entrada da Ponte H
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;

//Definicoes parametros para utilizacao do sensor
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // Setup da biblioteca
int distance = 0; //Variável que armazena a distância do objeto a ser lido
int detect = 0; //Variável que indica se algo foi detectado


void setup()
{
  pinMode(13, OUTPUT); //são para o led embutido piscar junto e indicar que
o mesmo está em funcionamento
  Serial.begin(9600); // Abre o monitor serial em 115200 bits por segundo
(baud) para ver o resultado da leitura
  //Define os pinos como saida
  pinMode(IN1, OUTPUT); //Pino 1 do motor A
  pinMode(IN2, OUTPUT); //Pino 2 do motor A
  pinMode(IN3, OUTPUT); //Pino 1 do motor B
  pinMode(IN4, OUTPUT); //Pino 2 do motor B
  pinMode(RST, OUTPUT); //Pino de Reset do ESP8266
  digitalWrite(RST, HIGH); //Pino de Reset do ESP8266
  esp8266.begin(9600); //Configura a velocidade de comunicação da software
serial para comunicar com o ESP8266

  //Rotina de checkagem da comunicacao do ESP8266
  String x = "";
  esp8266.print("ATrn");
  delay(1);
  while (esp8266.available())
    x += (char)esp8266.read();

  Serial.println(x);
```

```
  Serial.println("Enviar os dados:");
}

void loop() {

  delay(30); // Espera 50ms entre pings (cerca de 20 pings/s). 29ms é o
mínimo de tempo entre pings para uma leitura adequada.
  unsigned int uS = sonar.ping();
  // Manda um ping, recebe o tempo em microssegundos (uS).
  distance = uS / US_ROUNDTRIP_CM;

  if (distance >= MIN_DISTANCE && distance <= MAX_DISTANCE)
  {
    // Algo detectado
    detect = 0;
    // Aciona
    digitalWrite(13, HIGH); //led
    //Gira o Motor A no sentido horario
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    //Gira o Motor B no sentido horario
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
  }

  else if (distance > MAX_DISTANCE) {
    // Algo foi detectado
    detect = 1;
    // Aciona o motor
    digitalWrite(13, HIGH); //led
    //Gira o Motor A no sentido horario
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    //Gira o Motor B no sentido horario
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
  }

  else {
    digitalWrite(13, LOW); //led
    //Para o motor A
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, HIGH);
    //Para o motor B
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, HIGH);
    delay(1000);
    //Para o motor A
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    //Gira o Motor B no sentido anti-horario
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(2000);

  }

  Serial.print(detect); //Mostra se detectou algo
  Serial.print(", ");
  Serial.println(distance); //Mostra a distância do objeto
}
```

Apêndice C – Cotas do chassi dos veículos



Dept.	Technical reference	Created by Ricardo Celso 02/06/2018	Approved by	
		Document type	Document status	
		Title Chassi_TCC	DWG No.	
			Rev. / Date of issue	Sheet 1/1

Apêndice D – Código em PHP da página “Tela Inicial”

index.php

```
<!DOCTYPE html>
<! Codigo que cria a pagina inicial do servidor.>
<! Comando na linha 1 informa ao navegador que este vem a ser um codigo
html.>
<html>
<! Comando que visa providenciar os metadados da pagina html. Inicia-se na
linha 6 e termina na linha 11.>
<head>
        <! Elemento que representa diversos tipos de metadados.>
    <meta charset="UTF-8"/>
    <! Comando que define o titulo que aparece na aba do navegador.>
    <title>Autonomous Platoon</title>
</head>
<! Comando que visa delimitar o que compoe o corpo da pagina. Inicia-se na
linha 12 e termina na linha 33>
<body>
        <! Comando que define o texto contido nele como sendo de uma
categoria de titulo. Neste caso, "Servidor TCC".>
        <h1>Servidor TCC</h1>
        <! Comando que estabelece que o que esta escrito na linha sera
colocado em uma nova linha na pagina vista no navegador.>
        <p>Este projeto visa replicar de forma eficaz o autonomous platoon
visando o controle de velocidade e distÃ¢ncia dos veÃculos.</p>
        <! Comando que coloca uma imagem previamente alocada na pasta
"images" na tela.>
        <p><img src="images/comboio.jpg" alt="Caminhao"
title="Representacao Autonomous Platoon" /></p>
        <! Link que leva a pagina de definicao da ordem do comboio.>
        <p>Para definir a ordem dos integrantes do comboio clique aqui: <a
href="definicao.php">Tela de definiÃ§Ã£o</a></p>
        <! Link que leva a pagina de controle do veiculo lider.>
        <p>Para controlar o veÃculo lÃder clique aqui: <a
href="controle.php">Tela de controle</a></p>
        <! Codigo em php que visa zerar a variavel da velocidade contida no
banco de dados. Isso serve para garantir que os veiculos nao devem se mover
a menos que o usuario esteja controlando o lider na tela de controle.
Inicia-se na linha 25 e termina na linha 32.>
        <?php
                //Comando que inclui a variavel "$conexao" que esta em
"conecta.php".
                include("conecta.php");
                //Comando que define uma variavel "$vel" como tendo o valor
0.
                $vel=0;
                //Comando que visa atualizar o valor da variavel de
velocidade do lider contida no banco de dados "platoon". O comando
estabelece uma conexao com o banco por meio da variÃ¡vel "$conexao" e
atualiza uma tabela chamada "positions". Ele atualiza o "valor" (uma das
colunas da tabela) do item de id 4, que vem a ser o item da velocidade.
                mysqli_query($conexao,"UPDATE positions SET valor = $vel
WHERE id = 4");
        ?>
</body>
</html>
```

Apêndice E – Código em PHP da função que conecta o servidor ao banco de dados

conecta.php

```
<?php
        //Codigo em php que visa estabelecer uma conexao com o banco de
dados "platoon".
        $conexao=mysqli_connect("localhost", "root", "", "platoon");
?>
```

Apêndice F – Código em PHP da página “Tela de definição”

definicao.php

```
<!DOCTYPE html>
<! Codigo que cria a pagina de definicao das posicoes de cada veiculo no
comboio.>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>Ordem dos veÃculos</title>
</head>
<body>
        <h1>Tela de definiÃ§Ã£o</h1>
        <p>Indique abaixo quais as posiÃ§Ãµes do comboio</p>
        <! Comando que visa a criacao de um formulario de dados para serem
enviados. Possui tres caixas de texto devidamente identificadas para conter
as devidas posicoes de cada veiculo. Ao final, possui um botao que envia
essas posicoes para o arquivo "grava.php".>
        <form method="get" action="grava.php" class="formulario">
                VeÃculo 1 &rarr; <input type="text" name="1veiculo"
placeholder="NÃºmero no comboio" required class="input"><br><br>
                VeÃculo 2 &rarr; <input type="text" name="2veiculo"
placeholder="NÃºmero no comboio" required class="input"><br><br>
                VeÃculo 3 &rarr; <input type="text" name="3veiculo"
placeholder="NÃºmero no comboio" required class="input"><br><br>
                <input type="submit" value="Gravar" class="botform">
        </form>
        <! Link que leva a pagina inicial.>
        <p>Clique aqui para retornar a tela inicial: <a
href="index.php">Tela Inicial</a> </p>
</body>
```

Apêndice G – Código em PHP da função que grava as posições no banco de dados

grava.php

```
<?php
	//Codigo que visa gravar as informacoes fornecidas na pagina
"definicao.php" no banco de dados "platoon".
	include("conecta.php");
	//Os comandos a seguir definem os valores de posicoes informados na
outra pÃ¡gina em sua respectiva variavel.
	$vei1=$_GET["1veiculo"];
	$vei2=$_GET["2veiculo"];
	$vei3=$_GET["3veiculo"];
	//Caso a posicao do veiculo 1 esteja dentro do que seria possivel,
neste caso de um a tres, o programa segue para as outras condicoes.
	if ($vei1<=3 && $vei1>0){
		//Caso a posicao do veiculo 2 esteja dentro do que seria
possivel, neste caso de um a tres, o programa segue para as outras
condicoes.
		if($vei2<=3 && $vei2.value>0){
			//Caso a posicao do veiculo 3 esteja dentro do que
seria possivel, neste caso de um a tres, o programa segue para as outras
condicoes.
			if ($vei3<=3 && $vei3>0){
				//Caso as posicoes dos veiculos 1 e 2 nao
sejam iguais, ele segue para as outras condicoes.
				if($vei1!=$vei2){
					//Caso as posicoes dos veiculos 1 e 3
nao sejam iguais, ele segue para as outras condicoes.
					if($vei1!=$vei3){
						//Caso as posicoes dos
veiculos 2 e 3 nao sejam iguais, ele segue para as novas definicoes do
banco de dados.
						if($vei2!=$vei3){

	mysqli_query($conexao,"UPDATE positions SET valor = $vei1 WHERE id
= 1");

	mysqli_query($conexao,"UPDATE positions SET valor = $vei2 WHERE id
= 2");

	mysqli_query($conexao,"UPDATE positions SET valor = $vei3 WHERE id
= 3");
							//Comando que abre a
tela de exibicao das posicoes.

	header("location:lista.php");
						}
						//Caso as posicoes do veÃculo
2 e 3 sejam iguais, ele emite um alerta e se direciona de volta a pagina
"definicao.php".
						else{
							//Comando que,
convertido para javascript, emite um alerta na parte superior da tela.
							echo
"<script>alert('Mesma posiÃ§Ã£o definida para 2 e 3');

	window.location='definicao.php'</script>";
						}
```

```
                                        }
                                        //Caso as posicoes do veÃculo 1 e 3
sejam iguais, ele emite um alerta e se direciona de volta a pagina
"definicao.php".
                                        else{
                                                echo "<script>alert('Mesma
posiÃ§Ã£o definida para 1 e 3');

        window.location='definicao.php'</script>";
                                        }
                                }
                                //Caso as posicoes do veÃculo 1 e 2 sejam
iguais, ele emite um alerta e se direciona de volta a pagina
"definicao.php".
                                else{
                                        echo "<script>alert('Mesma posiÃ§Ã£o
definida para 1 e 2');

        window.location='definicao.php'</script>";
                                }
                        }
                        //Caso a posiaoo do veiculo 3 nÃ£o esteja dentro do
que seria possivel, ele emite um alerta e se direciona de volta a pagina
"definicao.php".
                        else{
                                echo "<script>alert('Valor do veÃculo 3 nÃ£o
corresponde a uma posiÃ§Ã£o');
                                window.location='definicao.php'</script>";
                        }
                }
                //Caso a posiaoo do veiculo 2 nÃ£o esteja dentro do que
seria possivel, ele emite um alerta e se direciona de volta a pagina
"definicao.php".
                else{
                        echo "<script>alert('Valor do veÃculo 2 nÃ£o
corresponde a uma posiÃ§Ã£o');
                        window.location='definicao.php'</script>";
                }
        }
        //Caso a posiaoo do veiculo 1 nÃ£o esteja dentro do que seria
possivel, ele emite um alerta e se direciona de volta a pagina
"definicao.php".
        else{
                echo "<script>alert('Valor do veÃculo 1 nÃ£o corresponde a
uma posiÃ§Ã£o');
                window.location='definicao.php'</script>";
        }
?>
```

Apêndice H– Código em PHP da página “Lista de posições”

lista.php

```
<!DOCTYPE html>
<! COdigo que cria a pagina que exibe as posicoes salvas no banco de dados
"platoon".>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>PosiÃ§Ãµes</title>
</head>
<body>
        <h1>Lista com as posiÃ§Ãµes definidas aos veÃculos</h1>
        <! Cria uma tabela que ira exibir as informacoes conforme as
colunas "nome" e "posicao">
        <table>
                <tr>
                        <td><strong>Nome</strong></td>
                        <td><strong>PosiÃ§Ã£o</strong></td>
                </tr>
                <?php
                        include("conecta.php");
                        $seleciona=mysqli_query($conexao,"select * from
positions WHERE id<4");
                        //While que visa alocar cada dado selecionado do
banco de dados na tabela.
                        while($campo=mysqli_fetch_array($seleciona)){?>
                                <tr>
                                        <td><?=$campo["nome"]?></td>
                                        <td><?=$campo["valor"]?></td>
                                </tr>
                        <?php } ?>
        </table>
        <p>Para redefinir a ordem dos integrantes do comboio clique aqui:
<a href="definicao.php">Tela de definiÃ§Ã£o</a></p>
        <p>Clique aqui para retornar Ã  tela inicial: <a
href="index.php">Tela Inicial</a> </p>
</body>
</html>
```

Apêndice I – Código em PHP da página “Tela de controle”

controle.php

```
<!DOCTYPE html>
<! Codigo que cria a pagina de controle do veiculo lider.>
<html>
<head>
    <meta charset="UTF-8"/>
    <title>Controle</title>
</head>
<body>
        <h1>Tela de controle</h1>
        <p>Controle o lÃder por meio dos botÃµes abaixo.</p>
        <p>Eles permitem a aceleraÃ§Ã£o do veÃculo para frente e para
trÃ¡s, sendo que cada movimento estÃ¡ representado pela devida seta.</p>
        <p>Para freiar o veÃculo, clique no botÃ£o de parada.</p>
        <! Comando que organiza os botoes de comando (frente e re
sinalizados pelas devidas setas) de modo a ficarem exatamente um acima do
outro.>
        <table>
                <tr>
                        <td></td>
                        <! Botao que aumenta a velocidade do veÃculo e
aciona a locomoÃ§Ã£o do mesmo para frente.>
                        <td><button onclick="frente();">&uarr;</button></td>
                        <td></td>
                </tr>
                <tr>
                        <td></td>
                        <! Botao que diminui a velocidade do veiculo e
aciona a locomocao do mesmo para re.>
                        <td><button onclick="re();">&darr;</button></td>
                        <td></td>
                </tr>
        </table>
        <! Botao que leva a velocidade do veiculo para 0.>
        <p><button onclick="parada();">Parada</button></p>
        <! Caixas de texto que visam informar ao usuario as velocidades
atuais de cada veiculo e as distancias informadas pelos sensores.>
        <p>Velocidade do lÃder:<input type="text" id="vel"> m/s</p>
        <p>Velocidade do segundo:<input type="text" id="vel2"> m/s</p>
        <p>DistÃ¢ncia do segundo:<input type="text" id="dis2"> m/s</p>
        <p>Velocidade do terceiro:<input type="text" id="vel3"> m/s</p>
        <p>DistÃ¢ncia do terceiro:<input type="text" id="dis3"> m/s</p>
        <p>OBS: Ao voltar Ã  tela inicial, a velocidade Ã© definida para 0
por questÃµes de seguranÃ§a.</p>
        <! Link que leva a pagina inicial.>
        <p>Clique aqui para retornar Ã  tela inicial: <a
href="index.php">Tela Inicial</a> </p>
        <! Codigo em php que visa determinar a velocidade atual que consta
no banco de dados. Esse valor vem a ser salvo na variavel $vel.>
        <?php
                include("conecta.php");
                //Comando que visa gravar todas as informacoes do item de id
4 na variavel "$seleciona".
                $seleciona=mysqli_query($conexao,"select * from positions
WHERE id=4");
                //Comando que organiza as informacoes que "$seleciona"
recebeu em um array, que vem a ser a variavel "$temp".
                $temp=mysqli_fetch_array($seleciona);
```

```
                //Comando que pega o que esta dentro da posicao "valor" de
"$temp" e grava isso na variavel "$vel"
                $vel=$temp["valor"];
                $seleciona=mysqli_query($conexao,"select * from positions
WHERE id=5");
                $temp=mysqli_fetch_array($seleciona);
                $dis2=$temp["valor"];
                $seleciona=mysqli_query($conexao,"select * from positions
WHERE id=6");
                $temp=mysqli_fetch_array($seleciona);
                $dis3=$temp["valor"];
                $seleciona=mysqli_query($conexao,"select * from positions
WHERE id=7");
                $temp=mysqli_fetch_array($seleciona);
                $vel2=$temp["valor"];
                $seleciona=mysqli_query($conexao,"select * from positions
WHERE id=8");
                $temp=mysqli_fetch_array($seleciona);
                $vel3=$temp["valor"];
        ?>
        <! Codigo em javascript que aciona determinadas funcoes de acordo
com o botao clicado. Cada botao esta diretamente associado a uma das
funcoes contidas nesse codigo. Tambem tem como funcao definir os valores
das caixas de testo das linhas 31 a 35 e atualizar a pagina a cada 5
segundos. Inicia-se na linha 62 e termina na linha 96.>
        <script type="text/javascript">
                //Conversao da variavel de php "vel" para a variavel de
javascript "velocidade".
                var velocidade=<?=$vel?>;
                var velocidade2=<?=$vel2?>;
                var velocidade3=<?=$vel3?>;
                var distancia2=<?=$dis2?>;
                var distancia3=<?=$dis3?>;
                //Comando que coloca o valor de "velocidade" dentro da caixa
de texto "vel".
                document.getElementById("vel").value=velocidade
                document.getElementById("vel2").value=velocidade2
                document.getElementById("vel3").value=velocidade3
                document.getElementById("dis2").value=distancia2
                document.getElementById("dis3").value=distancia3
                //Funcao que foi acionada pelo botao de aumentar a
velocidade.
                function frente(){
                        //Comando que abre a pagina "frente.php", que ira
aumentar a velocidade contida no banco de dados.
                        window.location.href = "frente.php"
                }
                //Funcao que foi acionada pelo botao de diminuir a
velocidade.
                function re(){
                        //Comando que abre a pagina "re.php", que ira
diminuir a velocidade contida no banco de dados.
                        window.location.href = "re.php"
                }
                //Funcao que foi acionada pelo botao de parada.
                function parada(){
                        //Comando que abre a pagina "parada.php", que ira
zerar a velocidade contida no banco de dados.
                        window.location.href = "parada.php"
                }
                //Funcao que atualiza a pagina
```

```
                function refresh(){
                        window.location.reload();
                }
                //Comando que conta 5 segundos para chamar a funcao
"refresh"
                setTimeout("refresh()",5000);
        </script>
</body>
<html>
```

Apêndice J – Código em PHP da função que aumenta a velocidade do líder no banco de dados

frente.php

```
<?php
        //Codigo que visa realizar o acrescimo da velocidade contida no
banco de dados.
        include("conecta.php");
        $seleciona=mysqli_query($conexao,"select * from positions WHERE
id=4");
        $temp=mysqli_fetch_array($seleciona);
        $vel=$temp["valor"];
        //Comando que visa acrescer a variavel "$vel" do valor dela mesma
mais 10.
        $vel=$vel+10;
        mysqli_query($conexao,"UPDATE positions SET valor = $vel WHERE id =
4");
        //Comando que visa retornar a pÃ¡gina de controle do lider do
comboio.
        header("location:controle.php");
?>
```

Apêndice K – Código em PHP da função que diminui a velocidade do líder no banco de dados

re.php

```
<?php
        //Codigo que visa diminuir a velocidade contida no banco de dados.
        include("conecta.php");
        $seleciona=mysqli_query($conexao,"select * from positions WHERE
id=4");
        $temp=mysqli_fetch_array($seleciona);
        $vel=$temp["valor"];
        //Comando que visa subtrair a variavel "$vel" do valor dela mesma
menos 10.
        $vel=$vel-10;
        mysqli_query($conexao,"UPDATE positions SET valor = $vel WHERE id =
4");
        header("location:controle.php");
?>
```

Apêndice L – Código em PHP da função que zera a velocidade do líder no banco de dados

parada.php

```
<?php
        //Codigo que visa zerar a velocidade contida no banco de dados.
        include("conecta.php");
        $vel=0;
        mysqli_query($conexao,"UPDATE positions SET valor = $vel WHERE id =
4");
        header("location:controle.php");
?>
```

Apêndice M – Código em PHP da função que fornece a posição do veículo 1 contida no banco de dados

pos1.php

```
<?php
        //Codigo em php que vem a ser acessado pelo controlador para se
saber qual a posicao do veiculo 1.
        include("conecta.php");
        $seleciona=mysqli_query($conexao,"select * from positions WHERE
id=1");
        $temp=mysqli_fetch_array($seleciona);
        $pos1=$temp["valor"];
        //Converte o numero de string para int.
        $pos1=intval($pos1);
        //Comando que emite a posicao do veiculo 1 para quem a selecionar.
        echo json_encode($pos1);
?>
```

Apêndice N – Código em PHP da função que fornece a posição do veículo 2 contida no banco de dados

pos2.php

```
<?php
    //Codigo em php que vem a ser acessado pelo controlador para se saber
qual a posicao do veiculo 2.
        include("conecta.php");
        $seleciona=mysqli_query($conexao,"select * from positions WHERE
id=2");
        $temp=mysqli_fetch_array($seleciona);
        $pos2=$temp["valor"];
        $pos2=intval($pos2);
        //Comando que emite a posicao do veiculo 2 para quem a selecionar.
        echo json_encode($pos2);
?>
```

Apêndice O – Código em PHP da função que fornece a posição do veículo 3 contida no banco de dados

pos3.php

```
<<?php
        //Codigo em php que vem a ser acessado pelo controlador para se
saber qual a posicao do veiculo 2.
        include("conecta.php");
        $seleciona=mysqli_query($conexao,"select * from positions WHERE
id=3");
        $temp=mysqli_fetch_array($seleciona);
        $pos3=$temp["valor"];
        $pos3=intval($pos3);
        //Comando que emite a posicao do veiculo 3 para quem a selecionar.
        echo json_encode($pos3);
?>
```

Apêndice P – Código em PHP da função que fornece a velocidade do líder contida no banco de dados

vel.php

```
<?php
        //<Codigo em php que vem a ser acessado pelo controlador para se
saber qual a velocidade do veiculo lider.
        include("conecta.php");
        $seleciona=mysqli_query($conexao,"select * from positions WHERE
id=4");
        $temp=mysqli_fetch_array($seleciona);
        $vel=$temp["valor"];
        $vel=intval($vel);
        //Comando que emite a velocidade do veiculo lider.
        echo json_encode($vel);
?>
```

Apêndice Q – Código em PHP da função que salva as informações enviadas pelo controlador no banco de dados

salva.php

```
<?php
	//Codigo em php que salva dados enviados pelo controlador no banco de dados.
	include("conecta.php");
	//Para modificar os dados do banco, se faz necessario informar a variavel que deve ser alterada dessa pagina. Abaixo estao os nomes para cada parametro.
	//(Velocidade do veiculo lider: vel) ; (Velocidade do segundo veiculo: vel2) ; (Velocidade do terciro veiculo: vel3)
	//(Distancia informada pelo veiculo lider: dis) ; (Distancia informada pelo segundo veiculo: dis2); (Distancia informada pelo terceiro veiculo: dis3)
	//Se a velocidade do veiculo lÃder foi atualizada, ela vem a ser salva no banco de dados.
	if($vel=$_GET['vel'])
		mysqli_query($conexao,"UPDATE positions SET valor = $vel WHERE id = 4");
	//Se a distancia do veiculo lÃder foi atualizada, ela vem a ser salva no banco de dados.
	if($dis=$_GET['dis'])
		mysqli_query($conexao,"UPDATE positions SET valor = $dis WHERE id = 5");
	//Se a distancia do veiculo 2 foi atualizada, ela vem a ser salva no banco de dados.
	if($dis2=$_GET['dis2'])
		mysqli_query($conexao,"UPDATE positions SET valor = $dis2 WHERE id = 6");
	//Se a distancia do veiculo 3 foi atualizada, ela vem a ser salva no banco de dados.
	if($dis3=$_GET['dis3'])
		mysqli_query($conexao,"UPDATE positions SET valor = $dis3 WHERE id = 7");
	//Se a velocidade do veiculo 2 foi atualizada, ela vem a ser salva no banco de dados.
	if($vel2=$_GET['vel2'])
		mysqli_query($conexao,"UPDATE positions SET valor = $vel2 WHERE id = 8");
	//Se a velocidade do veiculo 3 foi atualizada, ela vem a ser salva no banco de dados.
	if($vel3=$_GET['vel3'])
		mysqli_query($conexao,"UPDATE positions SET valor = $vel3 WHERE id = 9");
?>
```

Apêndice R – Código utilizado para teste com ESP8266 para aquisições de informações do servidor

```
#include <Arduino.h>
#include <String.h>

#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>

#include <ESP8266HTTPClient.h>



#define USE_SERIAL Serial

ESP8266WiFiMulti WiFiMulti;

//Variaveis usadas
int i = 0;
int estado = 1;

void setup() {

  USE_SERIAL.begin(9600);

  USE_SERIAL.println();
  USE_SERIAL.println();
  USE_SERIAL.println();

  for (uint8_t t = 4; t > 0; t--) {
    USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
    USE_SERIAL.flush();
    delay(1000);
  }

  WiFi.mode(WIFI_STA);
  // WiFiMulti.addAP("FSA-SALA20", "sl20@igv24*");
  // WiFiMulti.addAP("FSA-TERREO", "fatec@1234+");
  // WiFiMulti.addAP("FSA-PV1", "fatec@1234+");
  // WiFiMulti.addAP("FSA-PV2", "fatec@1234+");
  WiFiMulti.addAP("Moto G Play 8438", "ricardo2205");

}

void loop() {
  while (1) {
    switch (estado) {

      case 1: //RECEBE DADOS DO SERVIDOR
        // Aguarda pela conexao WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor
          http.begin("http://192.168.43.217/TCC/Servidor/pos2.php");
          USE_SERIAL.print("[HTTP] GET...\n");

          // Comeca a conexao e inicia o leitor de http
          int httpCode = http.GET();
```

```
          // Se httpCode for maior que zero
          if (httpCode > 0) {
            // Leitor HTTP e enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informaÃ§Ã£o estÃ¡ nessa string
              int payld = 0;
              for (int i = 0; i < payload.length(); i++) {
                if (char(payload.charAt(i)) >= '0' &&
char(payload.charAt(i)) <= '9')
                  payld = payld * 10 + (char(payload.charAt(i)) - 48);
              }
              USE_SERIAL.println(payld);
              if (payld == 1)
              {
                //Veiculo lider
                estado = 2;
              }

              else if (payld == 2)
              {
                //Segundo veiculo do comboio
                estado = 3;
              }

              else if (payld == 3)
              {
                //Ultimo veiculo do comboio
                estado = 4;;
              }
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }

        delay(10000);
        break;

      case 2: //ENVIA DADOS CARRO LIDER
        // Aguarda pela conexao WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor

http.begin("http://192.168.43.217/Servidor/salva.php?vel=80&dis=30");
          USE_SERIAL.print("[HTTP] GET...\n");

          // Comeca a conexao e inicia o leitor de http
```

```
          int httpCode = http.GET();

          // Se httpCode for maior que zero
          if (httpCode > 0) {
            // Leitor HTTP e enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informacao esta nessa string
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
        estado = 1;

        delay(10000);
        break;

      case 3: //ENVIA DADOS CARRO NA POSICAO 2
        // Aguarda pela conexao WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor

http.begin("http://192.168.43.217/Servidor/salva.php?vel2=80&dis2=30");
          USE_SERIAL.print("[HTTP] GET...\n");

          // Comeca a conexao e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero
          if (httpCode > 0) {
            // Leitor HTTP e enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informacao esta nessa string
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
```

```
        estado = 1;

        delay(10000);
        break;

      case 4: //ENVIA DADOS CARRO NA POSICAO 3
        // Aguarda pela conexão WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor

http.begin("http://192.168.43.217/Servidor/salva.php?vel3=80&dis3=30");
          USE_SERIAL.print("[HTTP] GET...\n");

          // Comeca a conexao e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero
          if (httpCode > 0) {
            // Leitor HTTP e enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor
            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informacao esta nessa string
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
        estado = 1;

        delay(10000);
        break;
    }
  }
}
```

Apêndice S – Código utilizado para teste com ESP8266 para aquisições e envio de informações para o servidor

```
#include <Arduino.h>

#include <String.h>

#include <ESP8266WiFi.h>

#include <ESP8266WiFiMulti.h>

#include <ESP8266HTTPClient.h>

#define USE_SERIAL Serial
ESP8266WiFiMulti WiFiMulti;

//VariÃ¡veis usadas

int i = 0;
int estado = 1;
int Pot=0; //Valor do potenciometro

void setup() {
  USE_SERIAL.begin(9600);

  USE_SERIAL.println();
  USE_SERIAL.println();
  USE_SERIAL.println();

  for (uint8_t t = 4; t > 0; t--) {
    USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
    USE_SERIAL.flush();
    delay(1000);
  }

  WiFi.mode(WIFI_STA);
  // WiFiMulti.addAP("FSA-SALA20", "sl20@igv24*");
  // WiFiMulti.addAP("FSA-TERREO", "fatec@1234+");
  // WiFiMulti.addAP("FSA-PV1", "fatec@1234+");
  // WiFiMulti.addAP("FSA-PV2", "fatec@1234+");
  WiFiMulti.addAP("Moto G Play 8438", "ricardo2205");

}

void loop() {
  while (1) {
    switch (estado) {

      case 1: //RECEBE DADOS DO SERVIDOR
        // Aguarda pela conexao WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor
          http.begin("http://192.168.43.217/TCC/Servidor/pos2.php");
          USE_SERIAL.print("[HTTP] GET...\n");
```

```
          // ComeÃ§a a conexao e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero

          if (httpCode > 0) {
            // Leitor HTTP Ã© enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor

            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informaÃ§Ã£o estÃ¡ nessa string
              int payld = 0;
              for (int i = 0; i < payload.length(); i++) {
                if (char(payload.charAt(i)) >= '0' &&
char(payload.charAt(i)) <= '9')
                  payld = payld * 10 + (char(payload.charAt(i)) - 48);
              }
              USE_SERIAL.println(payld);
              if (payld == 1)
              {
                //Veiculo lider
                estado = 2;
              }

              else if (payld == 2)
              {
                //Segundo veiculo do comboio
                estado = 3;
              }

              else if (payld == 3)
              {
                //Ultimo veiculo do comboio
                estado = 4;;
              }
              USE_SERIAL.println(payload);
            }

          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }

        delay(10000);
        break;

      case 2: //ENVIA DADOS CARRO LIDER
        // Aguarda pela coneÃ§Ã£o WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor
```

```
http.begin("http://192.168.43.217/Servidor/salva.php?vel=20&dis=50");
          USE_SERIAL.print("[HTTP] GET...\n");

          // ComeÃ§a a coneÃ§Ã£o e inicia o leitor de http

          int httpCode = http.GET();

          // Se httpCode for maior que zero

          if (httpCode > 0) {
            // Leitor HTTP Ã© enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor

            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informaÃ§Ã£o estÃ¡ nessa string
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }

        delay(2000);
        estado = 1;
        delay(10000);
        break;

      case 3: //ENVIA DADOS CARRO NA POSICAO 2
        // Aguarda pela coneÃ§Ã£o WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          Pot=analogRead(A0); //Recebe a leitura do potenciometro

          vel=map(Pot, 0, 1023, 0, 100); //Funcao que realiza uma regra de
3 para simular a velocidade do veiculo
          USE_SERIAL.println(vel); //Exibe qual o valor simulado da
velocidade

          // URL que sera acessada no Servidor
          URL="http://192.168.43.217/Servidor/salva.php?vel2="
          URL+=vel;
          URL+="&dis2=80"
          http.begin(URL);
          USE_SERIAL.print("[HTTP] GET...\n");

          // ComeÃ§a a coneÃ§Ã£o e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero
```

```
          if (httpCode > 0) {
            // Leitor HTTP Ã© enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor

            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informaÃ§Ã£o estÃ¡ nessa string
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
        estado = 1;

        delay(10000);
        break;

      case 4: //ENVIA DADOS CARRO NA POSICAO 3
        // Aguarda pela coneÃ§Ã£o WiFi
        if ((WiFiMulti.run() == WL_CONNECTED)) {

          HTTPClient http;

          // URL que sera acessada no Servidor

http.begin("http://192.168.43.217/Servidor/salva.php?vel3=40&dis3=50");
          USE_SERIAL.print("[HTTP] GET...\n");

          // ComeÃ§a a coneÃ§Ã£o e inicia o leitor de http
          int httpCode = http.GET();

          // Se httpCode for maior que zero

          if (httpCode > 0) {
            // Leitor HTTP Ã© enviado e o Servidor retorna um comando
            USE_SERIAL.printf("[HTTP] GET... code: %d\n", httpCode);

            // Arquivo encontrado no Servidor

            if (httpCode == HTTP_CODE_OK) {
              String payload = http.getString();
              //Toda a informaÃ§Ã£o estÃ¡ nessa string
              USE_SERIAL.println(payload);
            }
          }

          else {
            USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
          }

          http.end();
        }
```

```
            estado = 1;

            delay(10000);
            break;
        }
    }
}
```