

Universidade Federal de São Carlos – UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Engenharia Mecânica – DEMec

Trabalho de Conclusão de Curso

Protótipo de Veículo Autônomo
Controlado via Sinal GPS

Henrique Roncon

Orientador:
Prof. Dr. Luis Antonio Oliveira Araujo



São Carlos - SP – 2017

Universidade Federal de São Carlos – UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Engenharia Mecânica – DEMec

HENRIQUE RONCON

Projeto de Protótipo de Veículo Autônomo
Controlado via Sinal GPS

Trabalho de Conclusão de Curso
apresentado à Universidade Federal de São
Carlos como parte dos requisitos para
obtenção do título de bacharel em Engenharia
Mecânica.

Orientador:

Professor Dr. Luis Antonio Oliveira Araujo



São Carlos – SP – 2017



São Carlos, 21 de dezembro de 2017.

Ref.: Comissão avaliadora de TCC
Estudante: Henrique Roncon.

Em nome da coordenação do curso de engenharia mecânica, certifico que estudante **Henrique Roncon**, apresentou o trabalho de conclusão de curso (TCC), intitulado “**Projeto de Protótipo de Veículo Autônomo Controlado via Sinal GPS**” no dia 18 de dezembro de 2017, às 14h00, no LPI/Nuleen. Sendo este, aprovado, pela comissão avaliadora de TCC, composta pelos membros:

Prof. Dr. Luis Antonio Oliveira Araujo (presidente)

Prof. Dr. Rafael Vidal Aroca

Prof. Dr. Vitor Ramos Franco

Sem mais para o momento, subscrevo-me cordialmente.

Prof. Dr. Luis A. O. Araújo

Prof. Dr. Luis Antonio Oliveira Araujo
(Presidente)

Departamento de Engenharia Mecânica
Centro de Ciências Exatas e de Tecnologia
DEMec | CCET | UFSCar

“Dedico este trabalho aos meus pais, Manoel Ricardo Roncon e Geani Rodrigues Roncon, por todo o incentivo e auxílio que me deram sempre, principalmente neste período de graduação.”

AGRADECIMENTOS

Agradeço aos Professores Doutores Luis Antonio Oliveira Araujo e Rafael Aroca pela orientação e auxílio durante a execução deste trabalho. Sou grato também aos demais docentes e técnicos administrativos que, direta ou indiretamente, contribuíram no meu desenvolvimento que culminou na realização deste trabalho.

RESUMO

É notório o avanço da mecatrônica e das tecnologias computacional e robótica nas últimas três décadas. Dentre tantas ramificações provenientes da associação entre as engenharias mecânica, elétrica e de computação estão os estudos e desenvolvimento de veículos autônomos – tanto para uso civil quanto militar em operações de combate para se evitar a colocação de vidas em risco. Ainda que a fabricante Tesla desponte como exceção, a maior parte dos veículos fabricados atualmente ainda não apresentam total funcionalidade controlada por módulos e unidades eletrônicas, são vários os exemplos de dispositivos inseridos nos automóveis que já se encontram disponíveis no mercado: ativação do sistema de “airbag”, “cruise control”, autofrenagem e correção automática de direção são alguns destes recursos. Como o desenvolvimento da tecnologia tende ao objetivo da concepção, em um futuro próximo, de veículos capazes de se locomover de um ponto a outro sem intervenção humana, o presente trabalho tem como propósito realizar aplicação simplificada e em ambiente controlado do objetivo supracitado. Foi construído então um protótipo de veículo de tração independente gerenciado por Arduino – o qual controla o sistema de propulsão através do tratamento e análise de dados recebidos via sinal GPS, de maneira que um trajeto entre dois pontos declarados previamente ao computador de bordo fosse percorrido. Através da análise dos resultados verificados pelos experimentos, foi concluído que usar o sistema de GPS para a navegação de veículos autônomos pode ser uma opção viável em casos particulares e situações específicas de aplicação. Mesmo assim, a adição de dispositivos como bússolas e módulos ultrassônicos ao protótipo é fortemente recomendada para evitar colisões e acidentes.

Palavras-chave: Veículo autônomo. Arduino. GPS. Robótica móvel.

ABSTRACT

It is remarkable how mechatronic, computer technology and robotics have advanced in the past three decades. Among many branches descendant from the association between mechanical, electric and computer engineering are the researches and development of the autonomous vehicles – for civilian purposes or militaries right into the battlefield, to prevent or minimize situations where lives are threatened. Even though the Tesla Company emerges as an exception, most vehicles manufactured nowadays still do not have full assist ensured by modules and electronic control units, many are the examples of electronic devices that automatize a few functions and can be found in cars currently being commercialized: airbag system, cruise control, park assist and autopilot are four of them. As the technology advances and tends, in a not so far future, to give bases to build self-driving vehicles capable to travel between one waypoint to another, this monography has the purpose to describe an application – simplified and conducted into controlled environment – of the objective mentioned above. A prototype carrying an Arduino board – to control the propulsion system after signal and data treatment received through a GPS receptor, making possible that the robot could traffic between two waypoints previously programmed into the computer board. Analyzing the data provided by the experiments, it has been concluded that using a GPS system to guide autonomous vehicles could be an option in certain cases and specifically situations of application. Nevertheless, devices such as compasses and ultrasonic ranging module are strongly recommended to be added to the prototypes, to prevent collisions and accidents.

Keywords: Autonomous Vehicle. Arduino. GPS. Mobile robotics.

SUMÁRIO

RESUMO.....	9
ABSTRACT.....	10
1. INTRODUÇÃO.....	14
1.1. Objetivo geral.....	15
1.1.1 Objetivos específicos.....	15
1.2. Motivação.....	16
1.3. Organização da monografia	16
2. ESTADO DA ARTE	17
3. FUNDAMENTAÇÕES TEÓRICAS, MATERIAIS E MÉTODOS.....	19
3.1. Arduino	19
3.2. Sistema de Posicionamento Global (GPS).....	21
3.3. Motores elétricos e controle de acionamento	23
4. PROTOTIPAGEM	27
5. RESULTADOS	33
5.1. Objetivo Primário	34
5.2. Experimentação Secundária	40
6. CONCLUSÕES.....	43
7. REFERÊNCIAS BIBLIOGRÁFICAS	45

1. INTRODUÇÃO

Não mais tratada como ficção científica, a robótica móvel já se faz presente – ainda que de maneira tímida, é verdade – no cotidiano de um número considerável de pessoas ao redor do planeta. Desde aspiradores de pó e cortadores de grama até robôs organizadores de estoque e encomenda: o processo de terceirização de atividades comuns do dia-a-dia aos robôs com habilidade de movimentação autônoma caminha a passos largos nas mais variadas frentes imagináveis (ALECRIM, 2015). Empresas como a Honda e a SoftBank Robotics tem projetos em estado avançado de desenvolvimento, a segunda, inclusive, já comercializa o robô Pepper no Japão, capaz de interagir com humanos, realizar tarefas domésticas simples e manifestar emoções próprias (DUBEY et al., 2015). A empresa alemã, Kuka, que já é referência no mercado industrial há anos, para qual fornece braços robóticos e plataformas móveis autônomas, também pretende se aventurar levando seus braços robóticos, devidamente adaptados, para dentro dos lares familiares (McGEE, 2017). Uma das mais famosas e bem sucedidas companhias de delivery de pizza dos Estados Unidos, a Domino's, pretende até o início de 2018 realizar entregas na Alemanha e na Holanda por meio de um veículo autônomo, sem a necessidade de um humano intermediando o serviço (KAHN, 2017). De maneira análoga, a Amazon, empresa de e-commerce americana, desde 2016 realiza entregas pontuais de mercadoria através de drones autônomos (HERN, 2016).

É lógico, então, pensar que o setor automobilístico poderia ser também absorvido por esta propagação de automação que já abrange uma infinidade de outros ramos específicos. A locomoção rápida, sem qualquer tipo de preocupação ou necessidade de atenção ao volante propiciariam aos passageiros do veículo autônomo comodidade e possibilidade deles aproveitarem o tempo que desprendem no deslocamento, ajustando tarefas do trabalho, conversando entre si. No entanto, dentre todos os exemplos aqui citados, o ramo automobilístico é o mais problemático e difícil de aplicar os conceitos da robótica móvel. Se o aspirador de pó automático parar de funcionar ou se as encomendas solicitadas à Amazon ou à Domino's não forem entregues por falha de seus dispositivos de delivery, os inconvenientes gerados são infinitamente inferiores aos que poderiam ser causados se houvessem falhas de execução dos veículos autônomos tripulados. Os empecilhos para a rápida implementação da tecnologia nos automóveis não são exclusivamente então de cunho técnico: a partir do momento em que vidas são colocadas como fator no equacionamento de desenvolvimento de tais mecanismos, a complexidade e as ponderações que precisam ser levadas em consideração

para que se atinja patamar considerado de sucesso – e sem riscos às pessoas – aumentam consideravelmente (BONNEFON et al., 2015).

Tendo em mente as problemáticas envolvendo tal desafio, o presente trabalho visa estudar a utilização de dispositivos comerciais e comuns de *global positioning system*, GPS, no controle de rota destes veículos, uma vez que, quando utilizados, empregam-se dispositivos e sensores de alto preço devido à maior confiabilidade dos equipamentos. Ainda que sensores de movimento e posicionamento, por exemplo, sejam imprescindíveis para garantir a movimentação segura do automóvel, no presente trabalho apenas a eficácia do posicionamento via satélite será testada.

1.1. Objetivo geral

O objetivo principal do presente trabalho é verificar a eficiência na utilização de dispositivos GPS para o controle de rota de veículos autônomos, de maneira que possa ser averiguada sua aplicabilidade e constatação – positiva ou negativa – como alternativa aos métodos já utilizados atualmente que, além do emprego de dispositivos mais caros e mais precisos, compreende também àqueles sistemas que funcionam a partir de mapeamento em tempo real do ambiente através de câmeras, por exemplo.

1.1.1 Objetivos específicos

- Realizar revisão bibliográfica buscando trabalhos que apresentem diferentes soluções para o desenvolvimento de veículos autônomos;
- Especificar o princípio de funcionamento dos dispositivos GPS, bem como o detalhamento referente aos dispositivos de controle e automação que são empregados no protótipo aqui proposto;
- Desenvolver um código redigido em linguagem C capaz de promover a comunicação entre os sensores e atuadores do protótipo;
- Construir um protótipo simplificado de veículo capaz de se locomover entre dois pontos definidos via GPS sem a intervenção de um operador humano;

1.2. Motivação

Não é necessária demasiada reflexão para constatar que o desenvolvimento da tecnologia referente aos veículos autônomos tem ocorrido majoritariamente dentro dos departamentos de pesquisa de grandes montadoras, que investem consideravelmente para contar com o *know-how* referente ao assunto, criando assim um abismo desleal em relação às universidades que dispõem de laboratórios e cientistas capacitados que tentam contribuir também para o avanço deste ramo de pesquisa. Isso, de fato, não é um problema, já que o objetivo fundamental das montadoras se baseia exatamente na corrida pelo pioneirismo, alcançando cada vez mais visibilidade no mercado, enquanto as universidades abordam seus temas e objetivos com caráter menos agressivo, já que não há busca por lucro com os resultados obtidos. É um problema, porém, quando, mesmo com pouco recurso disponível, o assunto aqui abordado não seja discutido e pesquisado paralelamente dentro das universidades, já que se trata de um tópico em alta e que já se mostra como tendência de aplicação em massa em um futuro não tão distante. Não se trata de competir com as montadoras: trata-se de absorver a tecnologia atual e em processo de evolução e conduzir experimentações condizentes com o que tem de mais novo no mundo relacionado à engenharia mecânica, de computação e elétrica, diminuindo a disparidade que há atualmente, principalmente no Brasil, entre universidade e indústria.

1.3. Organização da monografia

Primeiramente, na seção do estado da arte, serão expostos estudos que se relacionam com a pesquisa de veículos autônomos ao longo dos últimos anos e qual o nível atual de desenvolvimento destes. Depois, os principais elementos eletrônicos agregados ao veículo deste trabalho serão brevemente abordados, de modo que fique claro o porquê do emprego deles na construção do protótipo. Na seção reservada à prototipagem, os elementos mecânicos utilizados serão definidos ao mesmo passo que a montagem geral do carro é detalhada. Embora o desenvolvimento do código possa ser considerado como parte do processo de montagem e prototipagem do veículo, optou-se por explicá-lo no capítulo dos resultados, pois, como será verificada pelo leitor, a resposta do veículo aos experimentos dependeu da iteração e variação de elementos relacionados à rotina carregada no computador de bordo. Por fim serão pontuadas considerações referentes ao trabalho e o que pôde ser concluído através dos resultados obtidos com o protótipo.

2. ESTADO DA ARTE

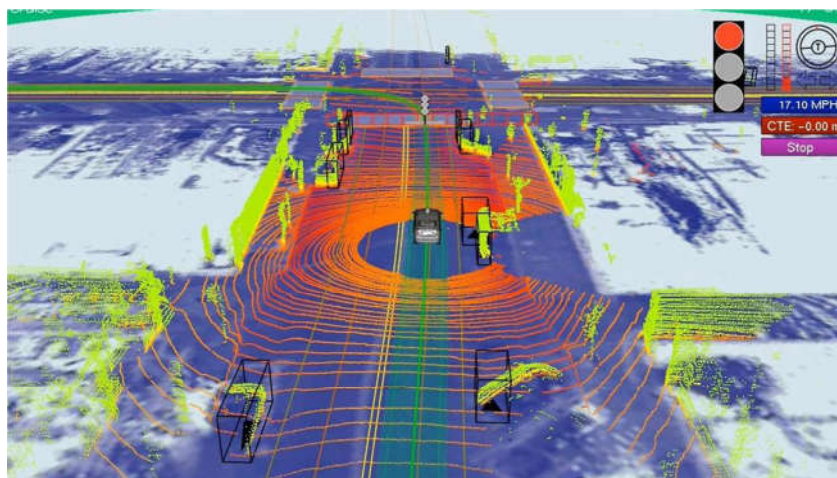
O ramo da indústria automobilística é, talvez, um dos campos de estudo mais abrangentes e desafiadores dentro do contexto da engenharia, uma vez que a concepção de um veículo envolve não só princípios técnicos e teóricos relacionados a esta ciência, mas também engloba fatores sociais, econômicos e culturais os quais estão associados ao mercado alvo de um novo projeto proposto. A grandiosidade do impacto desta porção industrial na sociedade pode ser exemplificada através, por exemplo, dos números que são vinculados a ela: de toda a produção mundial, em torno de 25% da fabricação de vidro, 50% da confecção de borracha e 15% da produção de aço são absorvidas pelas fábricas automotivas. Além disso, estima-se que estas representam nos países desenvolvidos 10% do valor de seus PIB (CASOTTI; GOLDENSTEIN, 2008). Os automóveis, por se tratarem então de um bem de consumo de fácil acesso atualmente, contribuem também para a manifestação de estatísticas negativas para a sociedade. Estima-se que, todo ano, mais de 1,2 milhões de óbitos são associados a acidentes de trânsito e, só nos Estados Unidos, 2,7 milhões de pessoas sofrem algum tipo de ferimento todo ano devido a colisões, o que causa um prejuízo demasiadamente grande aos cofres públicos anualmente no tratamento de feridos e no conserto de eventuais danos às vias (URMSON; WHITTAKER, 2008).

A proporção colossal da indústria dos automóveis é alimentada e movida não só pelo consumo da sociedade, mas também por ela própria. Bagloee et al. (2016) estimam que são investidos cerca de 77 bilhões de euros por empresas automobilísticas de todo o mundo em áreas de pesquisa e desenvolvimento de modo a fomentarem a inovação de seus projetos e se manterem competitivas dentro o mercado. Dentre tantos objetivos, tais investimentos são feitos também com o intuito de reduzir os prejuízos físicos e financeiros anteriormente citados. Fagnant e Kockelman (2015) citam as tecnologias de *parking assist* e *cruise control* – respectivamente, auxílio computadorizado no estacionamento e manutenção de velocidade do veículo – como exemplos claros de inovação recente da indústria automotiva que tem como enfoque a redução de acidentes. Estes pontuam ainda que algumas empresas foram capazes de desenvolver veículos quase totalmente autônomos, capazes de se locomoverem por rodovias e ambientes urbanos com auxílio mínimo de um condutor humano.

Há pelo menos dois casos que exemplificam a tentativa de total adaptação de veículos para transitarem de maneira autônoma. A empresa de tecnologia Uber realizou entre

os anos de 2016 e 2017, testes de locomoção independente de alguns de seus veículos, mas após um acidente envolvendo um destes protótipos no mês de março de 2017, a companhia decidiu suspender os experimentos (LEE, 2017). Por outro lado, a Google obteve sucesso em seus ensaios: desde 2010 até 2013, seus veículos autônomos completaram a marca de quinhentas mil milhas percorridas sem que qualquer tipo de colisão ocasionada por erro computacional tenha sido registrada (LUTIN; KORNHAUSER; LENER-LAM, 2013). Isto só foi possível devido ao alinhamento entre mapas de vias – desenvolvidos pela própria companhia – e um sistema integrado complexo de sensores e atuadores instalados no veículo que, em conjunto, garantem a condução segura deste.

Figura 1 – visualização do sistema Google de locomoção autônoma.



Fonte: <http://vus.virginia.gov/land/science-and-innovation/> (2017).

Ainda que a Google tenha obtido êxito em seus testes, o fato da primeira companhia ter enfrentado problemas graves – colocando em risco inclusive a vida de pedestres e outros condutores no acidente ocorrido – coloca em cheque a confiabilidade do sistema como um todo. Como adição, Sivak e Schoettle (2015) argumentam sobre uma numerosa lista de fatores limitantes para a inserção de automóveis autônomos no contexto atual de trânsito, como a necessidade de precisão e atualização das informações em tempo real para o veículo, já que a utilização de mapas e sensores não seria suficiente para que o mesmo soubesse da existência de vias interditadas, por exemplo.

De modo a contornar algum desses problemas que viriam a ser enfrentados por veículos civis, pode-se então buscar soluções alternativas aplicadas a robôs de resgate que atuam em ambientes inóspitos e sem rotas pré-definidas. Bahadori et al. (2005) desenvolveram um robô com tecnologia de mapeamento 3D instantâneo, através de “scans” a

laser e uma série de componentes sensitivos, que o possibilita trafegar em segurança no mapa recém gerado e promover a busca e salvamento de pessoas em perigo dentro deste ambiente, tudo isso de forma autônoma, livre de intervenção humana em seu controle e tomada de decisão.

3. FUNDAMENTAÇÕES TEÓRICAS, MATERIAIS E MÉTODOS

3.1. Arduino

A placa de Arduino, de ideia original concebida a partir da colaboração de diversos engenheiros e cientistas do *Interaction Design Institute Ivrea* na Itália, é um componente eletrônico simples munido de um processador e que tem como principal funcionalidade servir como controlador a artistas e amadores no ramo da mecatrônica os quais tem o desejo de criar ou replicar projetos onde se faz necessário o uso de microcontroladores, mas que não dispõem de capital para investir em computadores mais potentes, muitas vezes até desnecessários para o porte do projeto que pretendem executar (HUGHES, 2016). Ainda que apresentem baixa capacidade de processamento comparadas, por exemplo, com as computadores portáteis ingleses da Raspberry Pi, as placas de Arduino são mais do que capazes de promover a gestão de inputs e outputs para o controle de servo-motores, obtenção de dados provenientes de sensores de temperatura ou presença, itens bastante comuns nos projetos executados por amadores. No entanto, mesmo com limitações de hardware e software, as placas de Arduino podem ser empregadas em projetos de automação residencial, pequenas automações industriais e em projetos relacionados a servidores web, já que apresenta características de controle via wireless e bluetooth (MARGOLIS, 2011).

Hoje, a SparkFun Electronics, responsável pelas vendas das placas, comercializa em torno de 15 versões de Arduino, os quais são diferenciados principalmente pela quantidade de entradas e saídas disponíveis além do microprocessador que é instalado nelas. Todas as placas são munidas de processadores ATmega, produzidos pela Atmel, sendo a maioria da série AVR de 8 bits, exceto pela Arduino Due, que faz uso de um processador do tipo ARM. Neste projeto é utilizada a placa Uno, a qual conta com 6 portas analógicas mais 14 digitais e processador ATmega328, mais do que o suficiente para a execução do programa desenvolvido (ATMEL, 2017). A Figura 2 mostra a placa aqui descrita.

Figura 2 – placa Arduino Uno.



Fonte: <http://www.hobbytronics.co.uk/image/cache/data/arduino/> (2017).

Como o veículo terá locomoção externa, a alimentação da placa é feita através do uso de baterias de 12 volts e 9 Ah, que é conectada diretamente ao *jack* disponível no próprio Arduino. Embora a tensão e corrente de trabalho do circuito interno da placa sejam inferiores aos de entrada (na ordem de 5 volts e 50 mA, no máximo), ela conta com retificadores eletrônicos que regulam os valores de entrada para os limites suportados, o que evita que a placa seja danificada ou até mesmo inutilizada.

A programação da placa e upload de códigos se dão através do Ambiente de Desenvolvimento Integrado (IDE), o qual aceita rotinas escritas em linguagem própria baseada em C e C++. Trata-se de um ambiente intuitivo, de fácil operação e que permite o gerenciamento de informações de maneira clara e direta para o usuário. De acordo com Hugues (2016), isto pode ser observado, por exemplo, na administração das bibliotecas: para o controle de atuadores, sensores, displays ou qualquer outro periférico conectado à placa, são inseridas bibliotecas aos códigos programados as quais ampliam consideravelmente as funcionalidades do Arduino, fazendo com que a comunicação entre placa e periférico seja

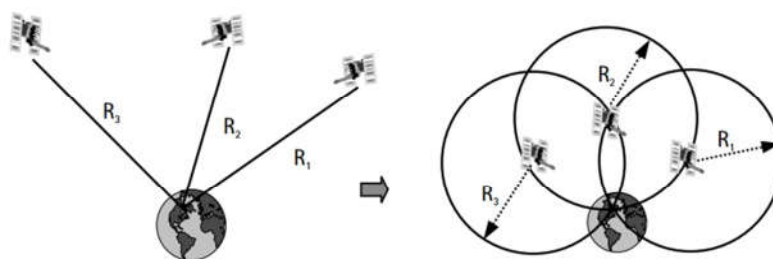
realizada de maneira mais confiável, já que parte da programação da comunicação e interface entre os dois é detalhada dentro dessas bibliotecas, disponíveis gratuitamente na internet.

3.2. Sistema de Posicionamento Global (GPS)

O Sistema de Posicionamento Global, ou Global Positioning System GPS, é um sistema de navegação baseado em uma rede de satélites que fornece dados referentes à posição e tempo em toda a superfície do globo terrestre para dispositivos capazes de receber tais dados. Embora sejam necessários apenas quatro satélites para determinar com precisão o local e horário de um dispositivo GPS, em 2015 já eram contabilizados 74 satélites em órbita, pertencentes à grupos americanos, russos e chineses, mas que são requisitados pelos aparelhos terrestres sem qualquer distinção entre eles, o que melhora consideravelmente a acurácia e precisão dos valores captados (GEOEDUC, 2015).

Ainda que o tratamento dos sinais provenientes dos satélites e a própria comunicação entre estes e as centrais de controle na Terra seja deveras complexa e até mesmo segredo de estado (originalmente o sistema de coordenadas GPS fora criado para uso militar, como pode ser consultado em EL-RABBANY, 2006), não cabe aqui o estudo teórico de como o sistema funciona, mas como um civil comum pode ter acesso a este tipo de informação. Cada um dos satélites no espaço envia constantemente para a Terra informações através de sinais em formato de ondas senoidais que são recebidas pela antena do dispositivo GPS ligado. Tais ondas contêm as coordenadas do satélite em órbita e o dispositivo trata de calcular, através de um software interno, a distância entre os dois. A necessidade de no mínimo quatro satélites serem necessários para a determinação exata da localização do dispositivo se deve ao fato de que, na verdade, o dispositivo está no ponto de intersecção entre o raio de alcance entre três satélites, que também comunicam-se entre si através das centrais de controle no solo, podendo assim calcular em tempo real, geometricamente, a posição relativa entre os pontos (EL-RABBANY, 2006).

Figura 3 – disposição e posicionamento dos satélites do sistema GPS.



Fonte: El-Rabbany (2006).

Para obter a posição do veículo aqui projetado em relação à superfície da Terra e orientá-lo através de pontos específicos no globo, foi utilizado o receptor u-blox neo 6m pois apresenta dimensões bastante reduzidas (totalizando volume de aproximadamente 470 mm^3) e compatibilidade total com o Arduino. Além da capacidade de interpretação de dados provenientes dos satélites, o dispositivo tem ainda funções para cálculo de medições através do efeito Doppler para determinação precisa de altitude e cinemática em tempo real, ainda que venham a ser funcionalidades não exploradas neste projeto (U-BLOX, 2017).

Figura 4 – receptor u-blox neo 6m.



Fonte: <http://www.hellasdigital.gr/images/detailed/> (2017).

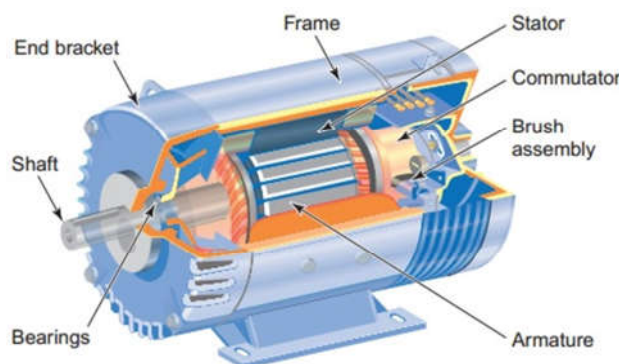
Como citado anteriormente, o Arduino faz uso de extensões de códigos denominadas bibliotecas. Para que a placa reconheça o receptor GPS e o receptor reconheça a placa e ocorra troca de informação entre as duas, é necessário o emprego de uma destas bibliotecas. Mais a frente esta biblioteca será discutida com mais profundidade, no entanto, cabe aqui destacar que ela faz uso de protocolos fornecidos pela *National Marine Electronics Association*, NMEA. A norma pela associação determina especificações de interface e formato dos dados que são compartilhados entre dispositivos que requerem informações

referentes à posição, velocidade e tempo, que é o princípio de funcionamento do sistema GPS (NMEA, 2008). Como se pode imaginar, originalmente os protocolos foram criados visando a comunicação na indústria marinha devido a dificuldade e importância de localização em alto mar, mas logo o protocolo passou a ser adotado como padrão nos mais variados ramos da comunicação por GPS, devido à confiabilidade da padronização imposta pelo grupo.

3.3. Motores elétricos e controle de acionamento

Neste projeto são utilizados dois motores do tipo CC (corrente contínua) para dar movimento ao carro, facilmente encontrados no mercado e que apresentam baixo custo, o que viabiliza muito a construção do protótipo.

Figura 5 – esquema genérico de motor CC.

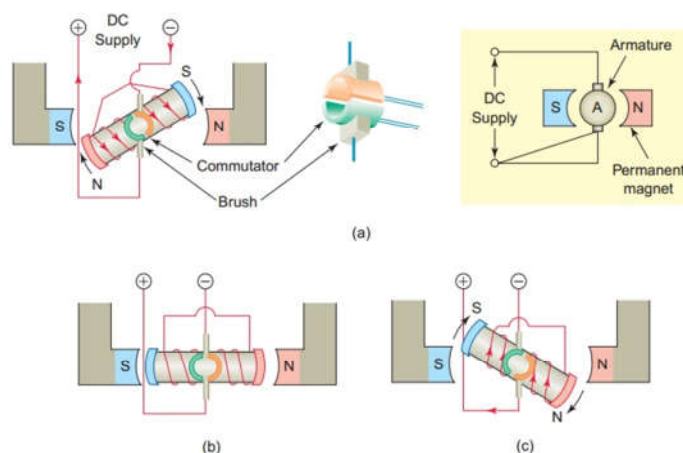


Fonte: Petruzella (2015).

Os motores de corrente contínua tem como princípio de funcionamento a interação entre um núcleo condutor e polos magnéticos dispostos ao redor dele. Nos modelos em que se utilizam ímãs permanentes, a corrente passa pelas espiras da armadura o que permite que estas passem a se comportar como um eletroímã. A armadura então é atraída pelos polos opostos dos ímãs permanentes (Figura 6a) dispostos ao redor da própria armadura, promovendo a rotação da mesma até que as linhas de campo magnético das duas estruturas alinhem-se (Figura 6b). Neste momento as buchas entram em um espaçamento presente no comutador que cessa a passagem de corrente nas espiras, extinguindo qualquer tipo de interação entre o ímã permanente e o eletroímã. Por um breve momento, no que pode-se considerar um período neutro em relação às interações magnéticas dentro do motor, a armadura continua girando mas única e exclusivamente devido à inércia do movimento induzido anteriormente. Ao passar pelo ponto neutro, as buchas voltam a entrar em contato com o comutador e permitem,

mais uma vez, a passagem de corrente pelas espiras mas agora em sentido contrário, causando agora não a atração mais sim a repulsão entre a armadura e os ímãs permanentes (Figura 6c) (PETRUZELLA, 2015).

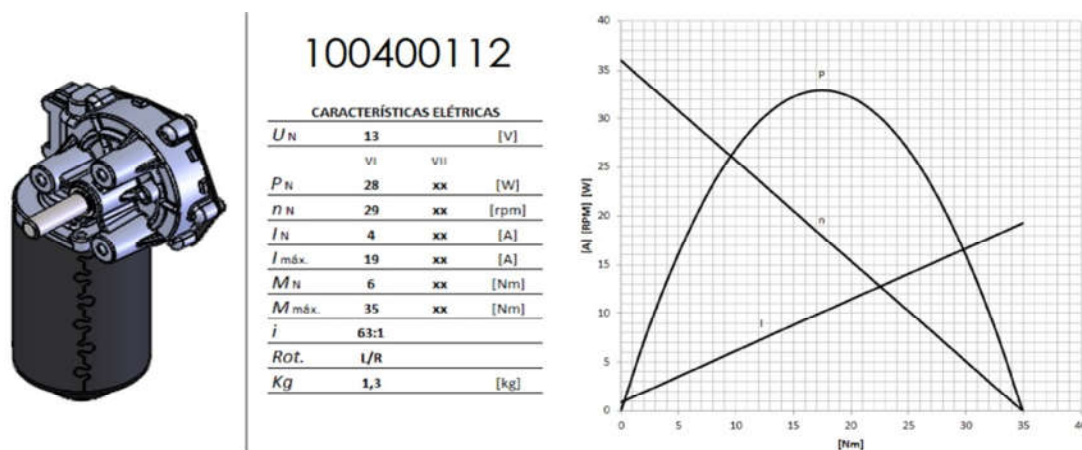
Figura 6 – etapas de operação genéricas do motor CC.



Fonte: Petruzella (2015).

A velocidade de rotação e torque desenvolvidos no eixo principal estão intimamente relacionados a aspectos construtivos do motor, como quantidade de espiras, de ímãs permanentes, da tensão de alimentação e da corrente que é desenvolvida nas espiras. Por se tratar de um dispositivo real, é lógico constatar que o rendimento do aparelho não é ideal, já que há perdas de potência devido à geração de calor, principalmente devido ao atrito entre os componentes internos do motor. Mesmo assim, motores robustos do tipo CC tendem a desenvolver rotações elevadas no eixo de saída, sendo necessário o emprego de redutores para o controle não só da velocidade mas do torque de saída também. Alguns fabricantes disponibilizam motores com redutores embutidos de fábrica, o que facilita o dimensionamento destas variáveis. A Imobras, Indústria de Motores Elétricos é uma destas companhias que vendem o dispositivo completo, e que fora escolhida aqui como a fabricante dos motores utilizados no projeto. O modelo em questão se trata do 100400112 pois, além de ser compatível com a bateria disponível para o veículo, tem curvas de velocidade e potência desenvolvidas suficientes para movimentar a plataforma.

Figura 7 – características do motor 100400112 da Imobras, utilizado no projeto.

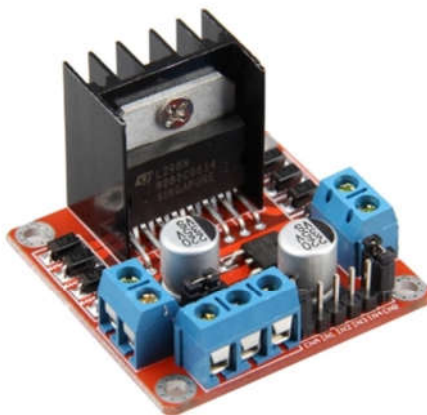


Fonte: Imobras (2017).

É importante salientar que os dois motores aqui empregados estão superdimensionados para o veículo. A redução de 63:1 e torque máximo de 35 Nm são muito superiores às requisitadas pelo carro, que tem massa aproximada de 6 kilogramas e requisição por, aproximadamente, 12 Nm de torque por roda para entrar em movimento – considerando que a massa do veículo se concentra no centro de cada roda, definidas como cilindros perfeitos e coeficiente de atrito estático entre roda e solo igual a 0,4. A propulsão, portanto, é ponto passível de reestudo em caso de aplicação futura deste projeto.

Para controlar o acionamento dos motores optou-se pela utilização de relés compatíveis com a corrente proveniente do Arduino e da requisitada por aqueles. A placa em questão é munida de dois relés, um para cada motor, modelo SRD-05VDC-SL-C, que suportam corrente de até 10 A, valor superior aos 4 A nominais desenvolvidos no motor.

Figura 9 – ponte H.

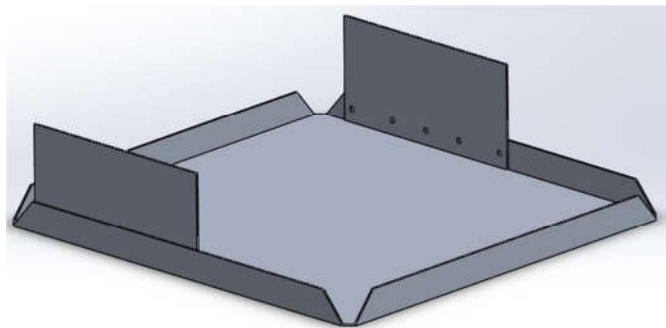


Fonte: <https://www.filipeflop.com/wp-content/uploads/> (2017).

4. PROTOTIPAGEM

Para a execução prática do projeto, foi construído um protótipo de veículo a fim de viabilizar os testes propostos ao algoritmo de movimentação de percurso deste. A fim de manter o projeto do veículo com caráter experimental, de fácil construção e possíveis modificações rápidas, decidiu-se que o volume total do bólido não ultrapassaria o envelope de um metro cúbico e deveria conter, obviamente, todos os componentes necessários para seu bom funcionamento como rodas, suporte para componentes eletrônico, baterias e qualquer outro adendo que se desejasse. Para a carroceria optou-se pelo uso de uma chapa de alumínio, com espessura de 2 milímetros e dimensões de comprimento e largura iguais a 450 e 465 milímetros, respectivamente. O alumínio é uma boa escolha para a plataforma devido a sua densidade relativamente baixa comparada a outros metais, além disso, o trabalho, como furos e dobras no material, são feitos de maneira simples e com instrumentos convencionais de oficina. As dimensões definidas para a placa comportaram com sobra os equipamentos montados nela, portanto, é totalmente plausível a diminuição do volume do carro através da substituição da chapa aqui usada por outra de proporções inferiores. Pensando ainda na montagem das rodas, de modo que fosse possível seu acoplamento com a plataforma foram fixados por parafusos dois suportes – também de alumínio – nos batentes laterais da chapa.

Figura 10 – esquema simplificado da plataforma do veículo.



Fonte: próprio autor.

As rodas utilizadas na tração do protótipo são fabricadas em plástico duro, com diâmetro de 214 milímetros e espessura total de 57 milímetros. Além de cavidades internas para alívio de peso, a roda conta com um furo central de 10 milímetros de diâmetro que comporta perfeitamente o eixo de saída do motor elétrico. Para o balanço da plataforma, uma terceira roda foi utilizada, central, oposta ao lado das rodas de tração, do tipo semelhante ao que se encontra em cadeiras de escritório.

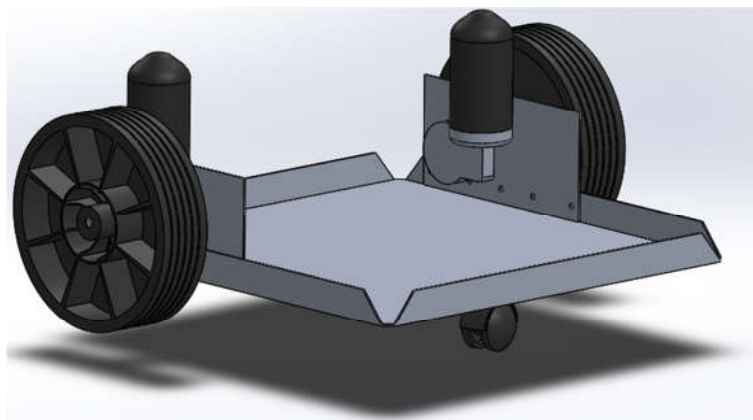
Figura 11 – rodas utilizadas no veículo.



Fonte: próprio autor.

Como brevemente explanado na seção de materiais e métodos, os motores modelo 100400112 da Imobras foram instalados nas chapas laterais de alumínio, centrados com as rodas de tração, também fixados por meio de parafusos. Nas figuras a seguir, além da representação final em CAD do que foi proposto para o chassi do veículo, algumas imagens mostram como as fixações dos componentes foram feitas na prática.

Figura 12 – esquema de montagem final da plataforma.



Fonte: próprio autor.

Figura 13 – fixação da chapa lateral e motor.



Fonte: próprio autor.

Figura 14 – fixação da roda de apoio.



Fonte: próprio autor.

Devido à massa da bateria, de cerca de dois kilogramas, não houve necessidade de um sistema de fixação complexo. Como a velocidade desenvolvida pelo veículo não ultrapassa 1 metro por segundo, em terrenos planos – onde foram feitos os testes – a bateria não se move devido à locomoção ou às trepidações do carro. Bastou então prender suas bases à plataforma com fita isolante, apenas por segurança extra.

Figura 15 – bateria empregada para alimentação elétrica do protótipo.

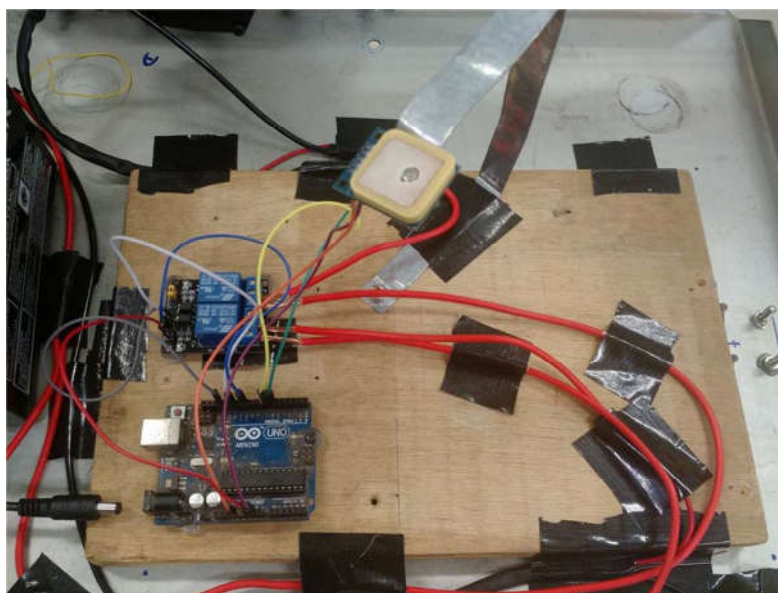


Fonte: <http://unipower.com.br/produto/bateria-estacionaria-vrla-12v-9ah-mod-up1290/> (2017).

Para a disposição dos elementos eletrônicos do veículo, fora utilizada uma base de madeira colada à plataforma. A madeira, além de poder ser facilmente furada para a colocação de parafusos, é má condutora de eletricidade, o que evita qualquer tipo de falha

catastrófica devido a curto-circuito em caso de mau contato ou fuga de corrente por parte de um dos componentes presos a ela. Uma haste de alumínio fora instalada na base madeira a fim de posicionar o receptor GPS acima dos demais componentes do veículo, de modo a evitar interferências eletromagnéticas provenientes possivelmente dos motores elétricos.

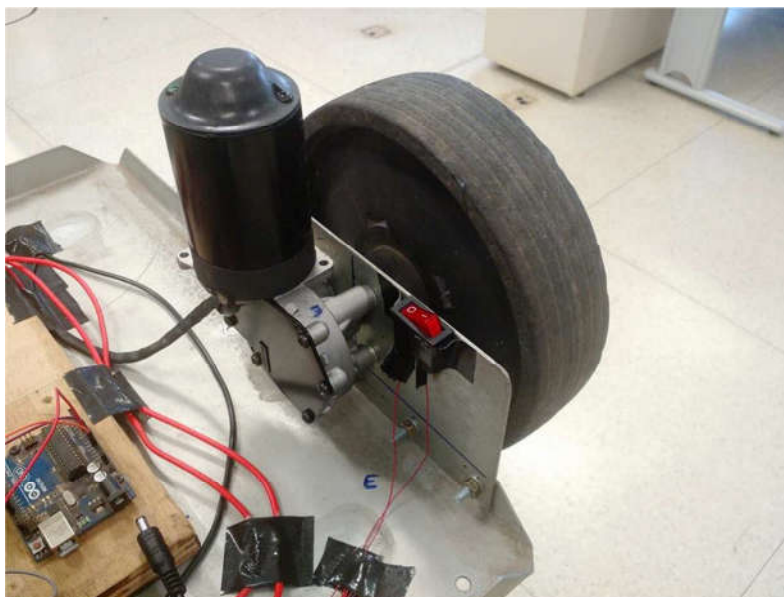
Figura 16 – base para os componentes eletrônicos.



Fonte: próprio autor.

Por fim, foi preciso apenas conectar os aparelhos por meio de cabeamento. Os motores têm os polos negativos conectados diretamente na bateria, enquanto o sinal positivo é chaveado através dos relês de comando. Nesta conexão foram utilizados cabos de dois milímetros de diâmetro, padronizados pela Imobras. Para as conexões entre Arduino, receptor GPS e a placa de relês, foram empregados cabos jumpers comuns para o trabalho com este tipo de equipamento. Tais cabos são de classificação 24 AWG, o que representa uma bitola igual a 0,5 milímetros. Como também citado anteriormente, o Arduino é alimentado diretamente pela bateria através do jack disponível na placa sem qualquer preocupação com tensões e correntes de entrada, já que ela possui retificadores para ambas. Para facilitar a ligação e desligamento do veículo, um botão do tipo on/off fora adaptado ao circuito.

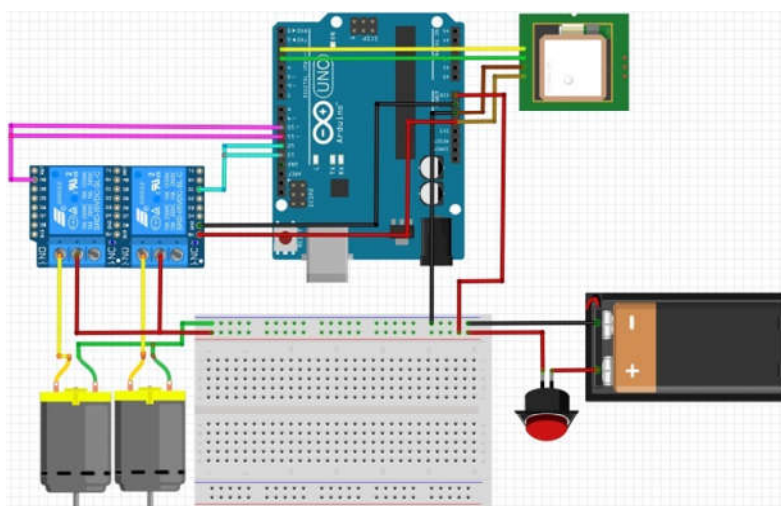
Figura 17 – detalhe do botão aplicado no sistema.



Fonte: próprio autor.

Abaixo, na Figura 18, o esquema elétrico do protótipo é representado. Os pinos utilizados podem ser melhor identificados através dos códigos anexados nos apêndices deste trabalho. Nas figuras posteriores o carro é mostrado completamente montado.

Figura 18 – esquema elétrico do veículo.

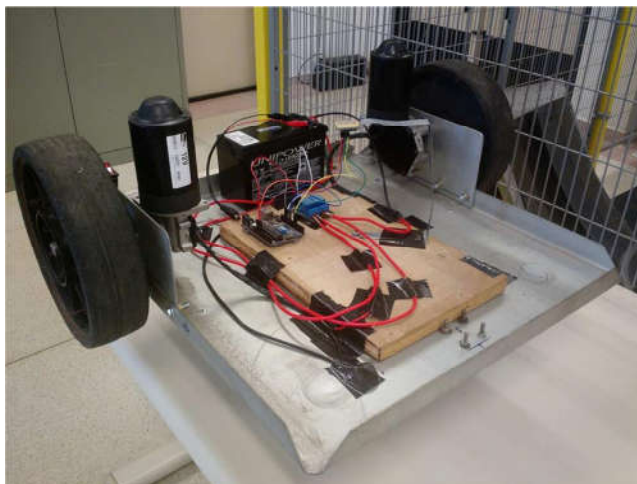


Fonte: próprio autor.

Para averiguar o bom funcionamento dos componentes do sistema, alguns procedimentos foram tomados. No caso do cabeamento e alimentação de modo geral, um multímetro simples foi usado para verificar a integridade dos fios e os valores de corrente e

tensão em cada um dos dispositivos. Não apenas isso: sub-rotinas genéricas foram compiladas a fim de testar individualmente cada um dos equipamentos em regime de trabalho. Esta conduta fora tomada para que, ao começar os testes em bancada da rotina, os erros atribuídos ao mau funcionamento dos aparelhos fossem descartados automaticamente e fosse claro que o problema seria única e exclusivamente devido a falhas de programação.

Figura 19 – vista geral do protótipo.



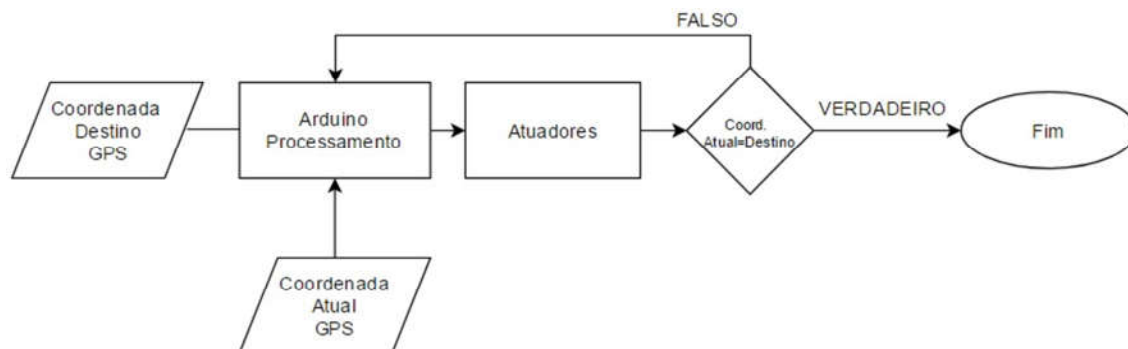
Fonte: próprio autor.

5. RESULTADOS

Uma vez que a parte física do veículo estava completa, o trabalho de desenvolvimento de códigos e experimentação pode ser iniciado. Esta seção do texto será dividida em duas partes: a primeira consistirá na descrição da realização em si do projeto proposto a esta monografia, onde o veículo parte de um ponto A arbitrário e se locomove até um ponto B pré-definido; a segunda parte tratará da tentativa de expansão do código, possibilitando o veículo trafegar por múltiplos pontos pré-definidos.

Primeiramente foi preciso definir a lógica que será processada no Arduino. A rotina escrita precisa ser capaz de monitorar a movimentação do veículo por comparação de valores de coordenadas globais, e corrigir a rota por meio do acionamento dos servomotores acoplados às rodas do veículo. Pela Figura 20 é possível ter um melhor entendimento da forma de processamento e funcionamento do sistema mecatrônico do protótipo.

Figura 20 – gráfico de fluxo de tomada de decisão do mecanismo de controle do protótipo.



Fonte: próprio autor.

A coordenada de entrada proveniente do receptor GPS (Coordenada Atual no fluxograma) é atualizada constantemente, em intervalos de dois segundos, e fornecida ao processador para tratamento enquanto a Coordenada Destino é pré-definida na rotina do programa. Enquanto o processador identificar diferença de posicionamento entre a coordenada de entrada e a coordenada de destino, os atuadores são acionados – de maneira que o veículo ande para frente, faça curvas para a direita e esquerda ou pare, mas não ande em sentido reverso – até que este chegue ao seu destino e a diferença entre coordenadas seja, em metros, zero ou próxima deste valor. Os dois formatos de locomoção, entre dois pontos e entre múltiplos pontos descritos logo abaixo, utilizam da mesma lógica para programação.

5.1. Objetivo Primário

Tendo como referência o fluxograma da Figura 20, o processo de desenvolvimento da rotina visando a locomoção do bólido entre dois pontos foi iniciado. Sabendo da necessidade de utilizar uma biblioteca para facilitar a escritura do programa, dentre algumas opções disponíveis gratuitamente para uso civil, optou-se pelo emprego da TinyGPS++, versão mais atualizada das bibliotecas do tipo TinyGPS, desenvolvidas por Mikal Hart. A principal função da TinyGPS++ é fazer a comunicação entre o módulo da U-blox e a placa Arduino através do fornecimento de uma série de funções pré-programadas ao usuário a fim de simplificar tarefas como, por exemplo, a obtenção de posição, velocidade, altitude, data e curso provenientes dos dados captados através do protocolo NMEA (HART, 2013).

Embora cada uma das linhas do código – exposto na íntegra no Apêndice A – seja imprescindível para o bom funcionamento dele, dois dos comandos utilizados precisam ser melhor explanados pois definem o princípio de funcionamento básico de todo o sistema.

- `TinyGPSPlus::distanceBetween()`: como o nome sugere, a função calcula a distância d entre dois pontos da superfície da Terra através da latitude e longitude (lat , $long$) de cada um deles. Então, dados dois pontos conhecidos, a biblioteca executa a seguinte ordem de cálculo:

$$\Delta_1 = long1 - long2 \quad (5.1)$$

$$\Delta_2 = \{\cos(lat1) * \sin(lat2) - [\sin(lat1) * \cos(lat2) * \cos(\Delta_1)]\}^2 \quad (5.2)$$

$$\Delta_3 = \Delta_2 + [\cos(lat2) * \sin(\Delta_1)]^2 \quad (5.3)$$

$$\Delta_4 = \sqrt{\Delta_3} \quad (5.4)$$

$$\Delta_5 = \sin(lat1) * \sin(lat2) + \cos(lat1) * \cos(lat2) * \cos(\Delta_1) \quad (5.5)$$

$$d = \arctan\left(\frac{\Delta_4}{\Delta_5}\right) * 6372795 \quad (5.6)$$

O fator multiplicativo 6372795 equivale à aproximação para o raio terrestre, assim, o valor retornado em d é computado em metros. Como a Terra não é uma esfera perfeita, o valor do fator é aproximado, portanto, há um erro de cálculo da ordem de 0,5% (HART, 2013).

- `TinyGPS::courseTo()`: tem como finalidade calcular e retornar a posição angular entre dois pontos da superfície da Terra de modo similar à função `distanceBetween`, utilizando das latitudes e longitudes destes pontos. O resultado, θ , em graus, é calculado da seguinte forma:

$$\varepsilon = long2 - long1 \quad (5.7)$$

$$\alpha = \sin(\varepsilon) * \cos(lat2) \quad (5.8)$$

$$\beta_1 = \sin(lat1) * \cos(lat2) * \cos(\varepsilon) \quad (5.9)$$

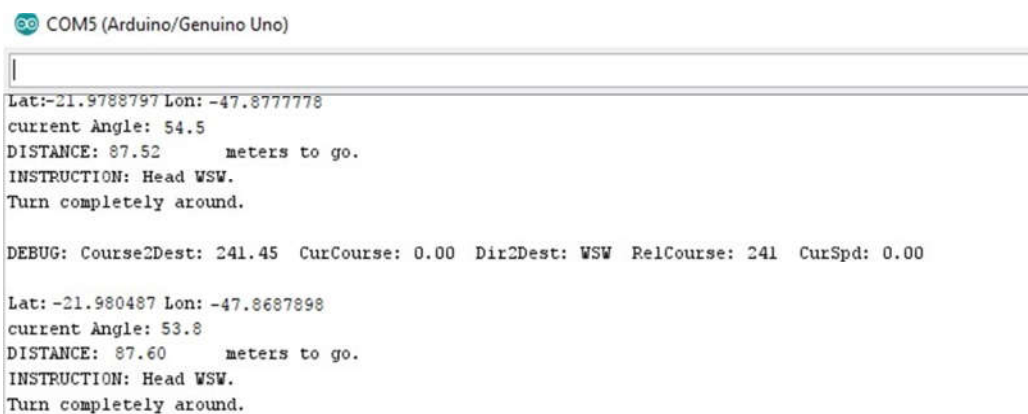
$$\beta_2 = \cos(lat1) * \sin(lat2) - \beta_1 \quad (5.10)$$

$$\theta = \arctan\left(\frac{\alpha}{\beta_2}\right) * \left(\frac{180}{\pi}\right) \quad (5.11)$$

Os comandos acima descritos necessitam de quatro entradas, que são as latitudes e longitudes dos dois pontos na Terra. O primeiro ponto é o pré-programado na rotina e o segundo é atualizado – em intervalos de um em um segundo, definido na rotina do Apêndice

A – conforme o veículo se movimenta, já que corresponde à localização receptada pelo dispositivo U-blox e transferida ao processador por intermédio da biblioteca. Em caso de falha de recepção de sinal por parte do U-blox, o carro para por falta de dados para processamento. De acordo com os valores das funções `courseTo` e `distanceBetween`, o carro consegue perceber se sua distância relativa com o ponto pré-programado está diminuindo ou aumentando conforme se movimenta, além disso, se a direção em que está se locomovendo tende a diminuir ou aumentar essa distância, já que o ângulo relativo de movimento também é calculado periodicamente. Inclui-se então ao programa, funções de esterçamento. Entre faixas de valores para o ângulo relativo, a rotina executa linhas de comando a fim de esterçar o carro para a direita ou esquerda, buscando alcançar valor de ângulo relativo igual à zero – ou inferior à 15 graus, como definido nesta rotina – para então acionar os dois motores igualmente e fazer com que o veículo movimente-se para frente. Como já mencionado, a Ponte H facilitaria o comando de curva regulando a tensão de alimentação dos motores de tração. Ao fazer uso dos relês, é possível apenas chavear a tensão proveniente da bateria, impossibilitando o controle de velocidades deles. Desta forma, a ação de esterçamento foi resolvida ativando o motor de uma roda por mais tempo que o motor da roda oposta, a fim de, gradativamente, acertar a direção do carro. Para monitoramento geral dos parâmetros, foi incluso na rotina funções para mostrar ao usuário – em tempo real – os valores referentes à latitude e longitude onde o veículo se encontra, ângulo relativo ao ponto de destino e quantos metros faltam para o carro chegar nele, como pode ser conferido na Figura 21.

Figura 21: display das funções processadas no Arduino.



```
COM5 (Arduino/Genuino Uno)

Lat:-21.9788797 Lon:-47.8777778
current Angle: 54.5
DISTANCE: 87.52      meters to go.
INSTRUCTION: Head WSW.
Turn completely around.

DEBUG: Course2Dest: 241.45  CurCourse: 0.00  Dir2Dest: WSW  RelCourse: 241  CurSpd: 0.00

Lat:-21.980487 Lon:-47.8687898
current Angle: 53.8
DISTANCE: 87.60      meters to go.
INSTRUCTION: Head WSW.
Turn completely around.
```

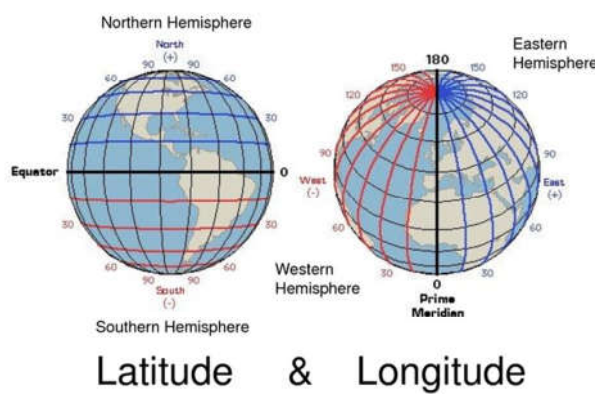
Fonte: próprio autor.

O ponto pré-programado de destino é definido logo nas primeiras linhas do código como parâmetros constantes. Foram testados diversos caminhos com a rotina aqui desenvolvida, variando ora ponto de partida, ora ponto de chegada ou ainda os dois ao mesmo tempo. A biblioteca TinyGPS++ converte os sinais protocolados pela NMEA provenientes dos satélites em órbita e realiza todos os seus cálculos utilizando coordenadas em formato decimal. De forma geral, a conversão de coordenadas de formato grau/minuto/segundo para decimal é:

$$Coordenada_{DECIMAL} = Grau + \frac{Minutos}{60} + \frac{Segundos}{3600} \quad (5.12)$$

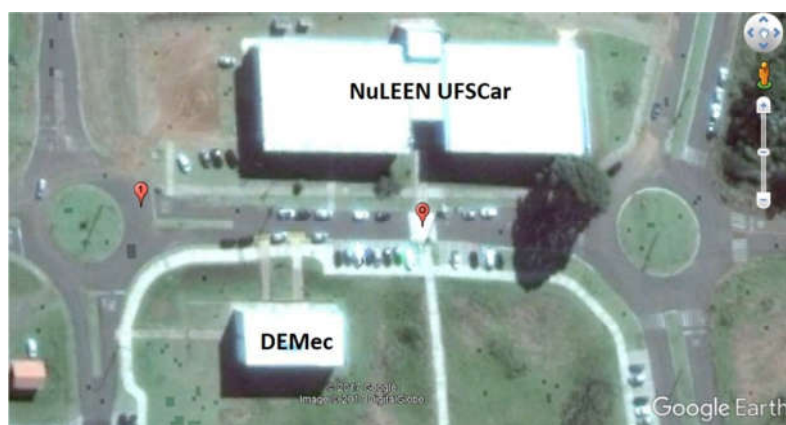
O sinal que acompanha o valor numérico da coordenada também é extremamente relevante. Para demonstrar a importância de determiná-lo corretamente, temos situada na latitude -22.018388967187764 e longitude -47.89106726646423 a Catedral de São Carlos, situada na principal avenida da cidade. Na latitude 22.018388967187764 e longitude 47.89106726646423 encontra-se o oásis Al-Aflaj, na região Riyadh, no centro da Arábia Saudita. Logo, a má definição do ponto na rotina pode acarretar em resultados totalmente discrepantes dos esperados ao executar o programa. Para evitar tais erros, utiliza-se a regra de sinal expressa na Figura 22 para adotar o sinal correto à coordenada decimal do ponto de destino do veículo, onde: os sinais positivos acompanham as latitudes ao norte da linha do Equador e longitudes à direita do Meridiano de Greenwich, e sinal negativo em caso contrário a estas normalizações.

Figura 22 – regra de sinal de coordenadas de acordo com a posição no globo.



Tendo em mente a necessidade de estabelecer com precisão os dados referentes às coordenadas do ponto de chegada, primeiramente o caminho a ser percorrido pelo veículo teve que ser estipulado. Dois dos caminhos testados são mostrados abaixo, nas Figuras 23 e 24, onde procurou-se verificar a capacidade do veículo se orientar corretamente e se locomover em sentidos opostos, no caso, Oeste e Leste, respectivamente.

Figura 23 – percurso número 1.



Fonte: Google Earth (2017).

Figura 24 – percurso número 2.

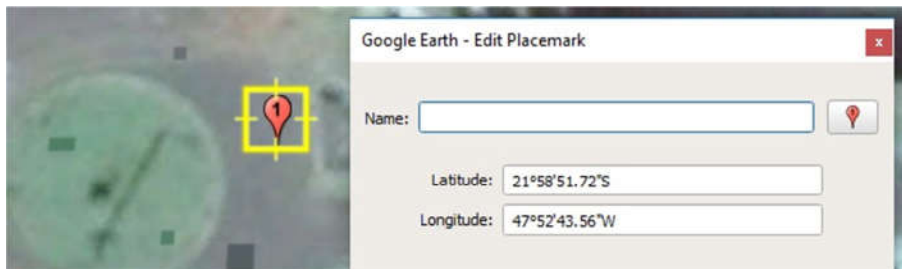


Fonte: Google Earth (2017).

O software Google Earth foi importante não só para a obtenção das imagens acima. No próprio software é possível demarcar *placemarks*, que são os balões vermelhos nas Figuras 23 e 24. O balão de número 0 representa o ponto de partida arbitrário onde o

carro foi ligado e os balões 1 e 2 são os destinos nos respectivos caminhos de teste. Ao demarcar o ponto de chegada, o software automaticamente define as coordenadas do ponto no sistema grau/minuto/segundo, como pode ser verificado na imagem a seguir. Em posse destes dados e aplicando a Equação 5.12, pode-se determinar o valor equivalente das coordenadas no sistema decimal e, com a Figura 25, definir os sinais destas.

Figura 25 – recurso de visualização de coordenadas no Google Earth.



Fonte: Google Earth (2017).

Ao ligar o veículo pelo botão on/off do circuito, observou-se pelo display programado na rotina que há um atraso na recepção dos dados dos GPS de cerca de dois minutos. Neste intervalo de tempo, o carro não se locomove, já que não há dados suficientes para serem processados pelo Arduino. Passado este tempo, o receptor começa a enviar dados à placa que por sua vez opera de acordo com as expectativas. Foram realizados cinco repetições em cada um dos percursos acima mostrados e o veículo percorreu o caminho sem maiores problemas, inclusive realizando pequenas curvas para ajuste de trajetória. A diferença de distância entre a posição real e a posição de destino determinada no programa foi, em todos os casos, inferior a cinco metros, o que comprovou a eficiência da rotina, já que este valor fora determinado na linha 59 do programa.

Em relação ao atraso no recebimento dos dados do GPS, não foi possível fazer conclusões concretas a respeito. Não há informações no *datasheet* do instrumento da Ublox que pontue tal comportamento e também não foram encontrados outros tipos de referência que descrevessem anomalia semelhante. Como não havia possibilidade de substituição do receptor, não é possível determinar com certeza se o problema é causado devido ao aparelho ou a rotina, nem mesmo se é um problema de fato. No entanto, ao testá-lo individualmente com a sub-rotina de aferição, ora os dados chegavam instantaneamente ao ligar o aparelho, ora apresentava atraso, sem qualquer tipo de razão aparente, já que o dispositivo fora testado tanto dentro dos laboratórios quanto a céu aberto, apresentando os dois comportamentos nos dois ambientes.

Mesmo com o funcionamento anormal do receptor GPS ao ligar o veículo, após o intervalo de tempo necessário para o estabelecimento de seu estado normal de funcionamento, ele e todos os outros componentes funcionaram de acordo com as expectativas, levando o bólido do ponto de partida arbitrário ao ponto de destino sem qualquer interrupção no caminho ou erros aparentes de programação da rotina. O veículo inclusive cessou o movimento dentro da região estabelecida para que fosse realizada tal ação. Por isso, mesmo com o *delay* para o funcionamento pleno do protótipo, pode-se concluir que o veículo atendeu aos requisitos impostos no início deste projeto.

5.2. Experimentação Secundária

Uma vez que o programa apresentava as funcionalidades requeridas para os objetivos principais atribuídos a este trabalho e existindo o desejo de melhorar os resultados obtidos, partiu-se para tentativas de aperfeiçoamento do código. Dentre tantas possibilidades de melhora, o foco aqui foi voltado para a criação de uma rota mais complexa a ser desenvolvida pelo veículo, de modo que fossem incluídas curvas e mudanças de direção aleatórias ao percurso a ser executado.

A lógica de programação para este desafio é a mesma utilizada anteriormente, portanto, segue à risca o fluxograma da Figura 20 em termos de mentalidade para desenvolvimento do código. Sabendo disso, percebeu-se que ao invés de escrever uma rotina completamente nova, seria possível apenas adaptar a rotina já comprovada funcional para atender a demanda de rota composta por múltiplos pontos. O desafio aqui consistiu em substituir o que antes era declarado como constante e passou a ser variável ao longo do percurso, que são as coordenadas de destino. Observemos a Figura 26 retirada do Google Earth onde o percurso de teste foi projetado para melhor entendimento da problemática.

Figura 26 – recurso de visualização de coordenadas no Google Earth.



Fonte: Google Earth (2017).

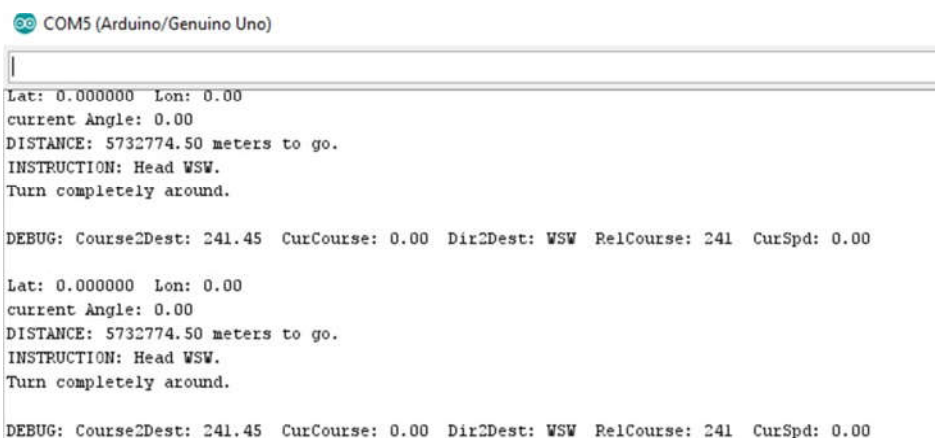
Assim como no experimento anterior, os balões numerados de 1 até 10 correspondem aos pontos de destino final e intermediários que, juntos, completam o percurso. Suas respectivas coordenadas decimais foram também definidas através do software e da utilização da Equação 5.12. Logo, o que deveria ser adicionado ao programa era a capacidade de trocar as variáveis associadas ao ponto de destino toda vez que o veículo passasse por uma das demarcações intermediárias.

Ao pesquisar sobre os recursos disponíveis para realizar tal tarefa, verificou-se que a IDE do Arduino conta com uma função pré-programada denominada *switch()*, que a princípio pareceu bastante promissora devido a facilidade e possibilidade da troca de valores referentes a uma determinada variável de acordo com alguns parâmetros pré-estipulados. Para o caso aqui em específico, a tentativa de implementação ocorre entre as linhas 118 e 169 do programa presente no Apêndice B. Neste bloco de comando, as coordenadas dos pontos de 1 até 10 foram definidas em *cases*, requeridos pela função *switch*. Aliado à ela, recorreu-se novamente ao uso da função *distanceBetween* e uma condição de distância para troca de pontos da seguinte forma: conforme o carro se movimenta em direção ao ponto 1, a distância relativa entre ele e o ponto reduz gradativamente; no momento que a distância é menor do que um valor, em metros, definido na rotina, a função *switch* troca o ponto de destino, passando a ser agora o balão de número 2, e assim sucessivamente até o balão final, no caso da Figura 26, de número 10, quando o carro para.

No experimento prático, no entanto, os resultados foram insatisfatórios. O veículo não conseguiu se localizar através das coordenadas GPS. Em qualquer direção que o carro fosse colocado, a única ação que foi tomada pelo processador foi de ativar os dois motores a

fim de guiar o carro sempre em linha reta. Ao inspecionar o display de variáveis do protótipo, verificou-se uma porção de anomalias, como pode ser observado na Figura 27.

Figura 27 – display das funções processadas no Arduino.



```
COM5 (Arduino/Genuino Uno)

Lat: 0.000000 Lon: 0.00
current Angle: 0.00
DISTANCE: 5732774.50 meters to go.
INSTRUCTION: Head WSW.
Turn completely around.

DEBUG: Course2Dest: 241.45 CurCourse: 0.00 Dir2Dest: WSW RelCourse: 241 CurSpd: 0.00

Lat: 0.000000 Lon: 0.00
current Angle: 0.00
DISTANCE: 5732774.50 meters to go.
INSTRUCTION: Head WSW.
Turn completely around.

DEBUG: Course2Dest: 241.45 CurCourse: 0.00 Dir2Dest: WSW RelCourse: 241 CurSpd: 0.00
```

Fonte: próprio autor.

Como o veículo foi programado para parar os movimentos em caso de não recepção de dados provenientes do receptor mas mesmo assim se locomovia indicando valores de latitude e longitude em tempo real nulas, constatou-se que, de alguma forma não identificada pelo autor da rotina, que o programa recebia mas zerava tais variáveis, sendo a única resposta para o movimento do veículo. Anomalias nos dispositivos periféricos foram descartadas após a verificação individual com as sub-rotinas. No entanto, mais uma vez, além de problemas com a rotina, as suspeitas voltaram-se também para o módulo GPS, já que havia apresentado comportamento inesperado nos experimentos utilizando apenas um ponto de destino.

Realizando testes em bancada, a estrutura do código foi exaustivamente modificada, de maneira que a função switch fosse escrita e alocada nos blocos de formas mais variadas possível. Seja trabalhando com o comando em bloco separado, em um comando de declaração específico (*void*), ou atribuindo a função a um loop de processamento já existente – e funcional, como verificado nos experimentos da rotina do Apêndice A – não houve êxito na tentativa de fazer com que os valores nulos atribuídos às variáveis longitude e latitude fossem alterados pelos valores da posição real do veículo. Assim, não é possível constatar de fato que o problema é definitivamente devido à estrutura do programa ou a defeitos na placa de recepção de dados GPS, mas de uma maneira ou de outra, não foram alcançados resultados positivos para esta tentativa de aperfeiçoamento.

6. CONCLUSÕES

A construção e desenvolvimento deste protótipo o qual culminou nesta monografia resultaram em frutos muito mais do que satisfatórios de maneira geral. Para o autor do projeto, não há dúvidas de que ao final dos experimentos a bagagem teórica e experimental relacionada a elementos de mecatrônica e computação cresceu enormemente, o que agrega muito a nível de conhecimento e aprendizado. Acredita-se também que este trabalho é mais uma maneira de trazer o assunto de robótica móvel como linha de pesquisa para dentro do Departamento de Engenharia Mecânica da UFSCar que, como grande universidade que é, não pode ficar atrás em um dos ramos de estudo mais promissores do futuro, que atualmente recebe bilhões de dólares em investimentos para o avanço de pesquisa e desenvolvimento de veículos autônomos.

Em relação aos resultados obtidos, pode-se constatar que o protótipo respondeu muito bem e dentro das expectativas quando sujeito às condições estipuladas para o experimento que deu motivação a este trabalho, onde o carro deveria partir de um ponto arbitrário e se locomover de forma autônoma até outro ponto B pré-definido na rotina de programação. No entanto, os resultados não foram satisfatórios ao executar tentativas de melhoria ao código, principalmente no que se diz respeito à implementação de múltiplos pontos de destino, para que fosse possível estabelecer não um caminho em linha reta ao veículo, mas sim uma trajetória mais complexa, composta por curvas e mudanças de direção ao longo do percurso.

As limitações envolvendo o receptor GPS infelizmente não puderam ser melhor estudadas a ponto de identificar com precisão a causa dos problemas que foram enfrentados ao longo do desenvolvimento das rotinas de processamento do veículo. Alterações no código foram de fato realizadas para eliminar possíveis erros de comunicação entre o Arduino e a antena U-blox, mas não seria correto atribuir as falhas dos experimentos totalmente a antena sem que houvesse a possibilidade de, utilizando um mesmo código, variar os receptores GPS empregados ao veículo. Os indícios são fortes, mas não é constatada neste trabalho uma resposta definitiva a este problema.

Não há dúvidas de que o protótipo aqui construído é passível de melhorias. Além de novos experimentos visando à adaptação do veículo para percorrer trajetos mais complexos, outras duas melhorias imediatas surgem como opção para o avanço da qualidade deste

veículo: o acréscimo de sensores de colisão e de uma bússola ao receptor GPS, ambos de nível de complexidade relativamente baixo para implementação. Os sensores de colisão de funcionamento baseado em ultrassom – como os módulos HC-SR04 – são imprescindíveis para o bom funcionamento do veículo em uma rota que apresente obstáculos não esperados no meio do caminho e poderia, inclusive, evitar acidentes devido à colisão com objetos ou até mesmo civis. A bússola compatível com a maioria dos módulos GPS e placas Arduino, HMC5883L, também agregaria muito ao projeto. Com ela, o processador identifica rapidamente e com precisão a direção que o veículo se movimenta, permitindo que o controle desta variável e o acionamento dos motores para a realização de curvas sejam feitos de maneira muito mais confiável e direta.

Por fim, conclui-se que a utilização de sensores GPS comerciais pode ser uma alternativa barata e muito confiável para o monitoramento e funcionamento de um veículo autônomo em alguns níveis de aplicação. Ainda que a distância final entre o carro e o ponto de destino tenha sido inferior ao estipulado no programa, em nenhum caso a distância foi igual ou muito próxima de zero. Isto mostra que, em situações em que a precisão de distância é um fator fundamental para a locomoção do robô, o emprego do GPS comercial não é uma alternativa viável ao projeto.

Sabendo das limitações que cercaram o desenvolvimento deste projeto e também do potencial que os estudos aqui iniciados têm perante não só à universidade, mas também ao ramo automobilístico em constante desenvolvimento fora do ramo acadêmico, é lógico cogitar que os trabalhos aqui iniciados tem grande potencial para serem não só revistos mas também continuados em novos projetos de conclusões de cursos e até mesmo em níveis de mestrado, por exemplo.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, Emerson. **Robô doméstico: você ainda vai ter um:** A indústria parece finalmente ter encontrado o caminho para o sucesso dessas máquinas. Você está pronto para dar “bom dia” ao seu robô?. 2016. Disponível em: <<https://tecnoblog.net/185818/robos-domesticos/>>. Acesso em: 20 nov. 2017.

ARTHUR PAIVA (São José dos Campos). Instituto Geoeduc. **Mais de 70 satélites de posicionamento global já estão em órbita. Entenda.** 2015. Disponível em: <<http://www.geoeduc.com/mais-de-70-satelites-de-posicionamento-global-ja-estao-em-orbita-entenda/>>. Acesso em: 6 nov. 2017.

ATMEL CORPORATION. *Arduino Uno datasheet*. 2017. Disponível em: <<http://datasheet.octopart.com/A000066-Arduino-datasheet-38879526.pdf>>

BAGLOEE, S. A. et al. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. **Journal of Modern Transportation**, Germany, v. 24.4, p. 284-303, 2016.

BAHADORI, S. et al. Autonomous systems for search and rescue. **Rescue Robotics**, Berlin, 2005. In press.

CASOTTI, B. P.; GOLDENSTEIN, M. Panorama do setor automotivo: as mudanças estruturais da indústria e as perspectivas para o Brasil. **BNDES Setorial**, Rio de Janeiro, v. 28, p. 147-187, set. 2008.

DUBEI, Shival; PRATEEK, Manish; SAXENA, Mukesh. Robot Locomotion - A Review. **International Journal Of Applied Engineering Research**, Delhi, v. 10, n. 3, p.7357-7369, nov. 2015.

EL-RABBANY, Ahmed. **Introduction to GPS: the global positioning system**. Norwood, Massachusetts: Artech House, 2006. 176 p.

FAGNANT, D. J.; KOCKELMAN, K. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations for capitalizing on self-driven vehicles. **Transportation Research Part A: Policy and Practice**, Amsterdam, v. 77, p. 167-181, 2015.

GOOGLE INC. **Google Earth, version 7.1**. Santa Clara, California. 2015. Disponível em: <<https://www.google.com/earth/download/gep/agree.html>>. Acesso em: 12 mai. 2017.

HART, Mikal. **TinyGPS++**: A new Full-featured GPS/NMEA Parser for Arduino. 2013. Disponível em: <<http://arduiniiana.org/libraries/tinygpsplus/>>. Acesso em: 5 ago. 2017.

HERN, Alex. **Amazon claims first successful Prime Air drone delivery**. 2016. Disponível em: <<https://www.theguardian.com/technology/2016/dec/14/amazon-claims-first-successful-prime-air-drone-delivery>>. Acesso em: 18 out. 2017.

HUGHES, John M.. **Arduino: A Technical Reference**. Sebastopol, California: O'reilly Media, 2016. 920 p.

IMOBRAZ MOTORES ELÉTRICOS. *Motorreductor 100400112 & 100400124*. 2017. Disponível em: <<http://www.imobras.ind.br/site/por/admin/arquivos/0df5f5d8222ffd1ed3a60ce8002f825e>>

KAHN, Jeremy. **Domino's will begin using robots to deliver pizzas in Europe**. 2017. Disponível em: <<http://www.chicagotribune.com/bluesky/technology/ct-dominos-robots-pizza-delivery-europe-20170329-story.html>>. Acesso em: 15 out. 2017.

LEE, D. Uber suspends self-driving cars after Arizona crash. *BBC News Technology*, Califórnia, 26 mar. 2017. Disponível em: <<http://www.bbc.com/news/technology-39397211>>. Acesso em: 30 mai. 2017.

LUTIN, J. M.; KORNHAUSER, A. L.; LERNER-LAM, E. The revolutionary development of self-driving vehicles and implications for the transportation engineering profession. **Institute of Transportation Engineers Journal**, Washington DC, v. 83.7, p. 28-32, 2013.

MARGOLIS, Michael. **Arduino Cookbook**. Sebastopol, California: O'reilly Media, 2011. 633 p.

MCGEE, Patrick. **Germany's Kuka plans move into world of personal assistant robots**: Industrial robotics maker will team up with Chinese parent Midea. 2017. Disponível em: <<https://www.ft.com/content/f80c390c-5293-11e7-bfb8-997009366969>>. Acesso em: 18 nov. 2017.

NATIONAL MARINE ELECTRONICS ASSOCIATION. **0183-2**: Standard For Interfacing Marine Electronic Devices. 3.01 ed. Saverna Park, Maryland: Nmea, 2002. 88 p.

PETRUZELLA, Frank D.. **Electric Motors and Control Systems**. New York: Mcgraw Hill, 2008. 280 p.

SIVAK, M.; SCHOETTLE, B. **Road safety with self-driving vehicles: general limitations and road sharing with conventional vehicles**. Michigan: The University of Michigan, 2015. 13 p. (report number: UMTRI-2015-2).

SONGLE RELAY. *SRD ISO9002*. 2017. Disponível em: <http://img.filipeflop.com/files/download/Datasheet_Rele_5V.pdf>

UBLOX. *NEO-6 u-blox 6 GPS Modules datasheet*. 2017. Disponível em: < https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf>

URMSON, C.; WHITTAKER, W. Self-Driving Cars and the Urban Challenge. **IEEE Intelligent Systems**, v. 23.2, p. 66-68, 2008.

APÊNDICES

APÊNDICE A – SKETCH PARA UM WAYPOINT

```
#include "TinyGPS++.h"
#include <SoftwareSerial.h>
static const int RXPin = 6, TXPin = 7;
static const uint32_t GPSBaud = 38400;
TinyGPSPlus gps;
SoftwareSerial ss(RXPin,TXPin);
#define dest_latitude -21.981028
#define dest_longitude -47.878767
unsigned long lastUpdateTime = 0;
int mA = 10;
int mB = 12;
float dir;
void setup() {
  Serial.begin(115000);
  ss.begin(GPSBaud);
  pinMode(mA,OUTPUT);
  pinMode(mB,OUTPUT);
}
void loop() {
  // Enquanto chegarem dados no GPS,
  // enviar para o tratamento via biblio TinyGPS++
  while (ss.available() > 0)
    gps.encode(ss.read());
  // Update de coordenadas a cada 1 seg.
  if (millis() - lastUpdateTime >= 1000)
  {
    lastUpdateTime = millis();
    Serial.println();
  }
}
```

```

// Estabelecendo status atual
double distanceToDestination = TinyGPSPlus::distanceBetween(gps.location.lat(),
gps.location.lng(), dest_latitude, dest_longitude);
double courseToDestination = TinyGPSPlus::courseTo(gps.location.lat(),
gps.location.lng(), dest_latitude, dest_longitude);
const char *directionToDestination =
TinyGPSPlus::cardinal(courseToDestination);
int courseChangeNeeded = (int)(360 + courseToDestination - gps.course.deg()) %
360;

// debug
Serial.print("DEBUG: Course2Dest: ");
Serial.print(courseToDestination);
Serial.print(" CurCourse: ");
Serial.print(gps.course.deg());
Serial.print(" Dir2Dest: ");
Serial.print(directionToDestination);
Serial.print(" RelCourse: ");
Serial.print(courseChangeNeeded);
Serial.print(" CurSpd: ");
Serial.println(gps.speed.kmph());
float l = gps.location.lat();
Serial.println();
Serial.print("Lat: "); Serial.print(l,6); Serial.print(" Lon: ");
Serial.println(gps.location.lng());
Serial.print("current Angulo: "); Serial.println(atan2(gps.location.lat(),
gps.location.lng())*180/M_PI);

// Dentro de um raio de 5 metros do destino
if (distanceToDestination <= 5.0)
{
Serial.println("Chegou");
para();
exit(1);
}

Serial.print("DISTANCIA: ");

```

```

Serial.print(distanceToDestination);
Serial.println(" metros restantes.");
Serial.print("INSTRUCAO: ");
// se parado, indicar direcao de curso.
if (gps.speed.kmph() < 2.0)
{
    Serial.print("Frente");
    Serial.print(directionToDestination);
    Serial.println(".");
    //return;
}
if (courseChangeNeeded >= 345 || courseChangeNeeded < 15)
{
    Serial.println("Continue a frente");
    drive_forward();
}
else if (courseChangeNeeded >= 315 && courseChangeNeeded < 345)
{
    Serial.println("Estercar ligeiramente para esquerda.");
    left();
}
else if (courseChangeNeeded >= 15 && courseChangeNeeded < 45)
{
    Serial.println("Estercar ligeiramente para direita.");
    right();
}
else if (courseChangeNeeded >= 255 && courseChangeNeeded < 315)
{
    Serial.println("Vire à esquerda.");
    left();
}
else if (courseChangeNeeded >= 45 && courseChangeNeeded < 105)
{
    Serial.println("Vire à direita.");

```

```
        right();
    }
    else
    {
        Serial.println("Vire 180 graus.");
        right();
    }
}

void drive_forward(){
    digitalWrite(mA, LOW);
    digitalWrite(mB, LOW);
    delay(25);
}

void para(){
    digitalWrite(mA, HIGH);
    digitalWrite(mB, HIGH);
    delay(25);
}

void right(){
    digitalWrite(mA, HIGH);
    digitalWrite(mB, LOW);
    delay(25);
}

void left(){
    digitalWrite(mA, LOW);
    digitalWrite(mB, HIGH);
    drive_forward();
    delay(25);
}
```

APÊNDICE B – SKETCH PARA MULTIPLOS WAYPOINTS

```

#include "TinyGPS++.h"
#include <SoftwareSerial.h>

static const int RXPin = 6, TXPin = 7;
static const uint32_t GPSBaud = 38400;
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

int wpt = 0; // define variavel wpt e seta valor igual a zero para ela
double dest_latitude; // define variavel flutuante para latitude de destino
double dest_longitude; // define variavel flutuante para longitude de destino */
unsigned long lastUpdateTime = 0;
int WPT_RANGE = 3; // define alcance que o carro tem que estar (em metros) para
avançar para próximo destino

int mA = 10;
int mB = 12;
void setup() {
  Serial.begin(115200);
  ss.begin(GPSBaud);
  pinMode(mA, OUTPUT);
  pinMode(mB, OUTPUT);
}
void loop() {
  WayPts();
  // Enquanto chegarem dados no GPS,
  // enviar para o tratamento via biblio TinyGPS++
  while (ss.available() > 0)
    gps.encode(ss.read());
  // Update de coordenadas a cada 1 segundos.
  if (millis() - lastUpdateTime >= 1000)
  {
    lastUpdateTime = millis();
    Serial.println();
  }
}

```

```

// Estabelecendo status atual
double distanceToDestination = TinyGPSPlus::distanceBetween(gps.location.lat(),
gps.location.lng(), dest_latitude, dest_longitude);
double courseToDestination = TinyGPSPlus::courseTo(gps.location.lat(),
gps.location.lng(), dest_latitude, dest_longitude);
const char *directionToDestination =
TinyGPSPlus::cardinal(courseToDestination);
int courseChangeNeeded = (int)(360 + courseToDestination - gps.course.deg()) %
360;

// debug
Serial.print("DEBUG: Course2Dest: ");
Serial.print(courseToDestination);
Serial.print(" CurCourse: ");
Serial.print(gps.course.deg());
Serial.print(" Dir2Dest: ");
Serial.print(directionToDestination);
Serial.print(" RelCourse: ");
Serial.print(courseChangeNeeded);
Serial.print(" CurSpd: ");
Serial.println(gps.speed.kmph());
float l = gps.location.lat();
Serial.println();
Serial.print("Lat: "); Serial.print(l,6); Serial.print(" Lon: ");
Serial.println(gps.location.lng());
Serial.print("Ang. atual: "); Serial.println(atan2(gps.location.lat(),
gps.location.lng())*180/M_PI);
/*
// Se distância do veículo é menor que X metros, fim do percurso
if (distanceToDestination <= 5.0)
{
Serial.println("Fim do percurso!");
para();
exit(1);
}

```

*/

```

Serial.print("DISTANCIA: ");
Serial.print(distanceToDestination);
Serial.println(" metros restantes.");
Serial.print("INSTRUCAO: ");
// Se parado, indicar direção onde ir.
if (gps.speed.kmph() < 0.5)
{
    Serial.print("Head ");
    Serial.print(directionToDestination);
    Serial.println(".");
    //return;
}
if (courseChangeNeeded >= 345 || courseChangeNeeded < 15)
{
    Serial.println("Continue em frente!");
    drive_forward();
}
else if (courseChangeNeeded >= 315 && courseChangeNeeded < 345)
{
    Serial.println("Estercar ligeiramente para esquerda.");
    left();
}
else if (courseChangeNeeded >= 15 && courseChangeNeeded < 45)
{
    Serial.println("Estercar ligeiramente para direita.");
    right();
}
else if (courseChangeNeeded >= 255 && courseChangeNeeded < 315)
{
    Serial.println("Vire à esquerda.");
    left();
}

```

```

else if (courseChangeNeeded >= 45 && courseChangeNeeded < 105)
{
    Serial.println("Vire à direita.");
    right();
}
else
{
    Serial.println("Vire 180 graus.");
    right();
}
}
}

void WayPts () {
/* DEFINIR COORDENADAS DE GPS AQUI */
switch(wpt) {
    case 0:
        dest_latitude = -21.981028;
        dest_longitude = -47.878767;
        break;
    case 1:
        dest_latitude = -21.980925;
        dest_longitude = -47.878908;
        break;
    case 2:
        dest_latitude = -21.980731;
        dest_longitude = -47.878936;
        break;
    case 3:
        dest_latitude = -21.980567;
        dest_longitude = -47.878969;
        break;
    case 4:
        dest_latitude = -21.980339;
        dest_longitude = -47.878975;

```



```

        break;
    case 5:
        dest_latitude = -21.980272;
        dest_longitude = -47.879039;
        break;
    case 6:
        dest_latitude = -21.980253;
        dest_longitude = -47.879411;
        break;
    case 7:
        dest_latitude = -21.980339;
        dest_longitude = -47.879514;
        break;
    case 8:
        dest_latitude = -21.980594;
        dest_longitude = -47.879511;
        break;
    default:
        dest_latitude = -21.981028;
        dest_longitude = -47.878767;
        break;
    }

    double distanceToDestination = TinyGPSPlus::distanceBetween(gps.location.lat(),
gps.location.lng(), dest_latitude, dest_longitude);
    if (distanceToDestination < WPT_RANGE)
    {
        wpt++; // IMPORTANTE - muda para proxima coordenada
    }
}

void drive_forward(){
    digitalWrite(mA, LOW);
    digitalWrite(mB, LOW);
    delay(25);
}

```

```
void para(){  
  digitalWrite(mA, HIGH);  
  digitalWrite(mB, HIGH);  
  delay(25);  
}
```

```
void right(){  
  digitalWrite(mA, HIGH);  
  digitalWrite(mB, LOW);  
  delay(400);  
  digitalWrite(mA, LOW);  
  digitalWrite(mB, LOW);  
  delay(600);  
}
```

```
void left(){  
  digitalWrite(mA, LOW);  
  digitalWrite(mB, HIGH);  
  delay(400);  
  digitalWrite(mA, LOW);  
  digitalWrite(mB, LOW);  
  delay(600);  
}
```

APÊNDICE C – ESQUEMA ELÉTRICO DO VEÍCULO

