Universidade de São Paulo

Escola de Engenharia de São Carlos

Departamento de Engenharia

Elétrica

Trabalho de Conclusão de Curso

Desenvolvimento de um robô móvel autônomo de alta velocidade

Autor

Pedro Henrique Mosquera Santos

Orientador

Eduardo do Valle Simões

São Carlos, 2015

PEDRO HENRIQUE MOSQUERA SANTOS

DESENVOLVIMENTO DE UM ROBÔ MÓVEL AUTÔNOMO DE ALTA VELOCIDADE

Trabalho de Conclusão de Curso
apresentado à Escola de Engenharia de
São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase
em Sistemas de Energia e Automação

Professor Orientador: Eduardo do Valle Simões

São Carlos

2015

Mosquera Santos, Pedro Henrique
M894d Desenvolvimento de um robô móvel autônomo de alta velocidade / Pedro Henrique Mosquera Santos; orientador Eduardo do Valle Simões. São Carlos, 2015.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2015.

1. Robótica móvel. 2. Raspberry Pi. 3. Motor brushless DC. 4. Sensores ultrassônicos. 5. Navegação autônoma. I. Título.

FOLHA DE APROVAÇÃO

Nome: Pedro Henrique Mosquera Santos

Titulo: "Desenvolvimento de um robô móvel autônomo de alta velocidade"

Trabalho de Conclusão de Curso defendido e aprovado em 19/06/2015,

com NOTA 8,0 (oito, zero), **pela Comissão Julgadora:**

Prof. Dr. Eduardo do Valle Simões - (Orientador - SSC/ICMC/USP)

Prof. Associado Evandro Luís Linhari Rodrigues - (SEL/EESC/USP)

Prof. Dr. Valdir Grassi Junior - (SEL/EESC/USP)

Coordenador da CoC-Engenharia Elótrica - EESC/USP:
Prof. Associado Homero Schiabel

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Veículos aéreos não tripulados, ou VANTs, é um termo que abrange desde aviões com asas de mais de 100 metros de envergadura até pequenos helicópteros que pesam 80g. Os VANTs também variam quanto ao seu controle, podendo ter um operador remoto pilotando a aeronave ou não. No último caso, são chamados autônomos. Aeronaves autônomas ainda estão ganhando espaço no Brasil, sendo que somente em 2013 foi emitida a primeira autorização de voo experimental pela ANAC. Testes autônomos nesses sistemas tendem a ser perigosos, pois há o risco de uma aeronave em alta velocidade sair e controle ou colidir. Com a finalidade de tornar testes aéreos mais seguros, propôs-se o desenvolvimento de um robô móvel autônomo terrestre de pequeno porte utilizando a placa *RaspberryPi*. Utilizando como ponto de partida um robô autônomo que segue essas orientações, foi proposto o uso de motores brushless DC capazes de alcançar altas velocidades para a tração do robô. Dessa forma, era esperado obter um sistema autônomo terrestre capaz de alcançar velocidades próximas ao voo de VANTs, e com isso, poder fazer testes de características autônomas em um ambiente mais seguro antes de aplica-las às aeronaves. Foram utilizados sensores de distância por ultrassom como entrada do sistema de navegação, e testes foram realizados a fim de validar sua aplicação no ambiente proposto. O sistema de navegação implementado seguiu a arquitetura de subsunção desenvolvida por Rodney Brooks. É uma estrutura que permite a adição de novos comportamentos com facilidade, e dessa forma, é útil para uma plataforma que será usada para diferentes testes. Novas funcionalidades podem ser adicionadas e editadas sem modificar o comportamento que foi desenvolvido nesse trabalho.

Palavras chave: Robótica móvel, *RaspberryPi*, motor *brushless* DC, sensores ultrassônicos, navegação autônoma.

Abstract

Unmanned aerial vehicle, or UAVs, is a description that covers everything from planes which wings measure more than 100 meters wide to helicopters as light as 80g. UAVs also vary in regard to their control; they can have an operator piloting the aircraft remotely or not. The last one is called autonomous. Autonomous aircrafts are still gaining ground in Brazil, and only in 2013 the first authorization for an experimental flight was emitted by ANAC. Autonomous testing tends to be dangerous because there is always the risk of the high-speed airplane losing control or crashing. In order to make aerial testing safer, it is proposed the development of an autonomous ground mobile robot of small proportions that uses the Raspberry Pi board. Using an autonomous robot that follows these guidelines as a startup point, it is proposed the use of high speed brushless DC motors to traction the robot. Thus, it is expected to obtain an autonomous ground system capable of moving close to UAVs speed. Therefore, be able to perform tests on autonomous characteristics in a safer environment before applying it to the aircrafts. Ultrasound distance sensors will be used as input do the navigation system, and tests will be performed to validate their use in the proposed scenario. The navigation system to be implemented will follow Rodney Brooks' subsumption architecture. It is a structure that easily allows the addition of new behaviors, and thus, it is useful for a platform that will perform different types of tests. New functionalities can be added and edited without modifying the behaviors developed in this work.

Keywords: Mobile robotics, Raspberry Pi, brushless DC motor, ultrasound sensors, autonomous navigation.

1. Introdução

Os Veículos Aéreos Não Tripulados (VANTs) têm sido utilizados para fotogrametria, a ciência de extrair fotografias métricas já há algumas décadas (Wester-Ebbinghaus, W., 1980)(Warner, E. S., Graham, R. W., Read, R. E., 1996.). Apesar de essas aeronaves terem sido amplamente empregadas para usos militares (Scheve, T. Julho, 2012), hoje, os VANTs já são comercializados para o uso civil. O DECEA (Departmento de Controle do Espaço Aéreo) classifica como VANT qualquer veículo aéreo sem um piloto embarcado e de caráter não-recreativo, ou seja, um veículo aéreo não tripulado utilizado como hobby ou esporte se enquadra na legislação pertinente a aeromodelos e não a VANTs (Circular de Informações Aéreas AIC N 21/10).

As aeronaves não tripuladas são divididas pelo DECEA em dois tipos: RPAs (*RemotelyPilotedAircraft* ou aeronave remotamente pilotada, em português) e "Aeronaves Autônomas". Na primeira subcategoria, o piloto não está a bordo, mas controla a aeronave remotamente de uma interface qualquer, seja um computador, um simulador, um rádio controle, etc. O segundo tipo de VANT, chamado "Aeronave Autônoma" é definido pelo ministério da defesa (DCA. 63-4, 2013) como sistema de aeronave sem piloto que, uma vez programado, não permite intervenção externa durante o voo. Esse sistema tem seu voo proibido no Brasil.

Hoje, existem diversas aplicações para os VANTs: fiscalização de áreas de preservação ambiental ou fronteiras; monitoramento de obras civis de usinas hidroelétricas, rodovias e plantas de captação; inspeção de linhas de transmissão de energia elétrica, oleodutos/gasodutos e plantas petrolíferas; fotografias esportivas; pesquisa atmosférica; e monitoramento para a polícia civil. Em 2013, a ANAC (Agência Nacional de Aviação Civil) emitiu o primeiro certificado CAVE (Certificado de Autorização de Voo Experimental) para uma empresa privada no Brasil (Xmobots, 2013) e iniciativas nessa área, como o GISA (Grupo de Interesse em Sistemas Autônomos e Aplicações do ICMC-USP – www.gisa.icmc.usp.br), estão se tornando mais populares.

Os VANTs podem ser de diferentes formatos e tamanhos, variando de aviões com asas fixas de 122 metros de envergadura para a distribuição de painéis solares (SolarEagle – Boeing, Setembro 2014) ou pequenos helicópteros de 80g que voam em grupo (Robobees – Harvard, 2015). Na figura 1, temos o perfil de diferentes tipos de aeronaves que receberam atenção da mídia recentemente (US Financial Times – Aerospace&Defence, Outubro 2013).

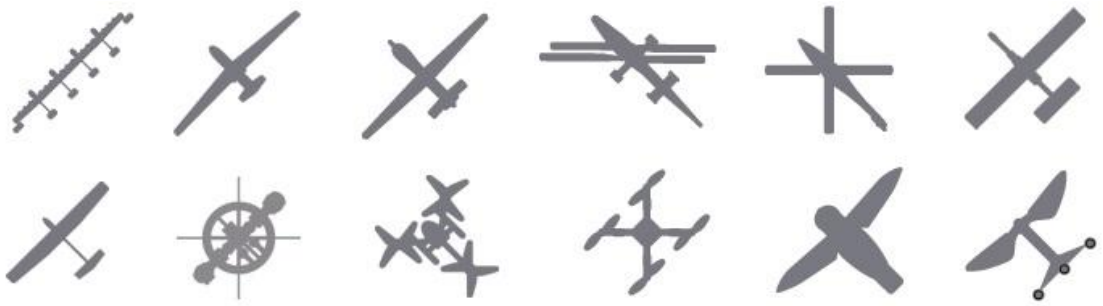

Figura 1 - Diferentes Perfis de VANTs

Na figura 1 são representados, na primeira fileira da esquerda para a direita: *SolarEagle* da Boeing, *GlobalHawk* da NorthropGrumman, *Predator* da General Atomics, K-MAX da Lockheed Martin, *Argus-IS* da BAE Systems e o *Falcon* da CLMaxEngineering. Na segunda fila, tem-se: *Raven* da AeroVironment, *T-Hawk* da Honeywell, X6 da Draganflyer, *Arducopter* da 3D Robotics, *Nano Hummingbird* da AeroVironment e o *Robobee* da Universidade de Harvard.

A aeronave que motivou essa pesquisa é o Projeto Ararinha do GISA (Ararinha – GISA ICMC –USP, Agosto 2013 -- www.gisa.icmc.usp.br). Esta aeronave foi desenvolvida pelo Laboratório de Computação Embarcada do ICMC-USP para servir de plataforma de projeto e teste de hardware e software embarcados para VANTs. Por possuir custo de produção muito baixo, de cerca de R$ 180,00, e ter seu projeto totalmente aberto a qualquer interessado, vários alunos de graduação e pós-graduação têm se interessado em aprender a construí-la e a desenvolver sistemas de controle para ele. Ela tem 160 cm de envergadura, 115 cm de comprimento e pesa 2,5 kg. A Ararinha chega a 80 Km/he em virtude de seu tamanho e da velocidade que atinge, testes de navegação autônoma com a mesma podem ser perigosos. Devido aos possíveis danos que um acidente com uma aeronave desse porte pode causar, surgiu a necessidade de se implementar um sistema terrestre no qual possam ser feitos testes autônomos de funcionalidades que serão acrescentadas ao avião depois de validadas.

Para a realização de testes em solo, idealizou-se uma plataforma autônoma que se movimentasse em velocidade próxima à velocidade de voo do Ararinha. Para atingir tais objetivos, esse trabalho propôs um veículo terrestre baseado no trabalho (GRANA A., 2013), mas que utilizasse os mesmos motores elétricos *brushless* que são encontrados em aeromodelos (D2836/9950Kv*Brushless* Motor – Hobby King). Esse motor de 950 Kv, ou seja, 950 rpm/V, possui tensão máxima de 12V, potência 243W e pode chegar a 11400 rpm. A velocidade teórica calculada para a roda de 4,5 cm de

diâmetro escolhida foi de aproximadamente 96,64 km/h, o que é suficiente para atender os requisitos dos testes terrestres.

Os motores *brushless* precisam de controladores eletrônicos de velocidade, conhecidos como ESCs (*Electronic Speed Control*), para comutar a corrente entre suas bobinas (40A ESC – Hobby King). Esses ESCs recebem um sinal de PPM (*Pulse Position Modulation*/ modulação por posição de pulso, em português), similar ao PWM (*Pulse Width Modulation* / modulação por largura de pulso, em português). Dessa forma, fez-se necessária a escolha de um microcontrolador que fornecesse o sinal para o controle dos motores.

Devido ao processamento embarcado necessário em diversas aplicações de maior complexidade, os microcontroladores de baixo custo muitas vezes não conseguem atender requisitos como processamento de vídeo e acesso à rede (GERKEY, VAUGHAN e HOWARD, 2003). Outra limitação é a maior complexidade dos algoritmos de navegação, os quais muitas vezes requerem mais memória e velocidade que os microcontroladores de baixo custo podem oferecer. Um sistema que utiliza, por exemplo, um algoritmo genético para atualizar os parâmetros de uma Lógica Fuzzy, de forma que o robô esteja adaptando seu comportamento constantemente ao ambiente (HOFFMANN, 2001) dificulta o uso de microcontroladores mais simples.

Com essa dificuldade em mente, e após testes com o sistema Arduino (microcontroladoresAtmega), a placa *RaspberryPi*(AboutUs | RaspberryPi, 2012) foi escolhida para o projeto do robô neste trabalho. A placa microprocessada *RaspberryPi*, originalmente voltada para o ensino de programação a crianças, ganhou atenção mundial pelo seu baixo custo e pequenas dimensões (8,6 cm de comprimento por 5,4 cm de largura). A placa foi introduzida no mercado em 2012 e se tornou popular pelo grande número de aplicações possíveis. Hoje, podem-se encontrar diversos projetos de hardware e software livre na Internet (RaspberryPi – AboutUs, 2012). A *RaspberryPi* tem um microprocessador baseado em um ARM de 700 MHz, com 512 MBytes de memória RAM e uma GPU (GraphicsProcessing Unit / Unidade de processamento de gráficos, em português) integrada ao processador. É considerada um *System On Chip* por trazer CPU, RAM e GPU integradas no mesmo chip. O armazenamento não-volátil é feito em um cartão SD, e a interface com o usuário é feita através de portas USB, HDMI e Ethernet, que facilitam a comunicação com outros hardwares. Acima de tudo, há uma grande comunidade envolvida com software livre oferecendo respaldo ao desenvolvimento com a *RaspberryPi*, por exemplo, existem

distribuições compiladas para o hardware que variam de centrais de mídia a servidores Web (RPiDistributions – Embedded Linux Wiki).

A escolha da *RaspberryPi* facilita a computação física do sistema, pois tem Linux embarcado. Dessa forma, a programação de baixo nível fica transparente e a conexão com outro hardware não apresenta maiores problemas. Para suprir a necessidade de interconexão com Internet, pode ser adicionado um módulo Wi-Fi à placa, e o gerenciamento do mesmo é realizado pelo próprio *Raspbian*, o sistema operacional escolhido para ser instalado na placa.

O acesso das portas de entrada e saída pode ser realizado através da biblioteca *WiringPi*. O objetivo da *WiringPi* é simular, na *RaspberryPi*, o mesmo acesso das portas de entrada e saída no sistema “wiring” do Arduino (About | WiringPi). Essa biblioteca pode ser usada por programas em C e C++, e torna intuitivo o acesso às portas de entrada e saída digital.

A desvantagem de se utilizar a *RaspberryPi* ao invés de um microcontrolador tradicional é o pequeno número de portas de entrada e saída para o acionamento de atuadores e leitura de sensores. São 8 pinos de entrada e saída, mas o usuário pode estender a 17 se usar os pinos das interfaces I2C (*Inter-IntegratedCircuit*), SPI (*Serial Peripheral Interface*) e UART (*Universal AsynchronousReceiver/Transmitter*) como entrada e saída. Contudo, dentre essas possíveis conexões, só há um pino disponível para gerar sinais de PWM/PPM para controlar a velocidade dos motores. Uma solução possível é a geração de um sinal PWM/PPM pela porta I2C (TOWNSEND/Adafruit, Agosto 2012). Para viabilizar essa solução, é necessária uma placa extra (AdafruitPiCobler), que não foi utilizada no decorrer deste trabalho.

O veículo terrestre proposto deverá possuir dois motores *brushless* DC com um controlador eletrônico de velocidade (ESC) para cada um. Dessa forma, a direção na qual o robô se desloca é dada pela diferença de velocidade nos dois motores. Contudo, a *RaspberryPi* só tem um PWM/PPM nos pinos de entrada e saída, e para a implementação do robô, como previsto acima, são necessários pelo menos dois. Uma forma de sanar essa dificuldade é se implementar o PPM com threads na linguagem C (Kar, R. | RPiBlog, Novembro 2012). Entretanto, a *RaspberryPi*não possui um RTC (Real Time Clock), e o PPM criado por software seria afetado pela execução de programas no processador, podendo fazer a largura e posição de pulso variar com a demanda de processamento. Essa temporização pode ser imprecisa, principalmente quando múltiplos processos estão sendo executados.

O sucesso do controle de robôs móveis depende em quão bem o mesmo interage com seus sensores e atuadores. No caso deste trabalho, os motores *brushless* DC e os sensores de distância por ultrassom. Essa interação pode ser feita das seguintes formas: deliberativo, reativo, híbrido ou hierárquico (WOLF, 2009). Na primeira, o controle é estabelecido como um plano prévio, baseado nos conhecimentos que o sistema possui sobre o problema. Esse controle apresenta desvantagens quando em frente a imprevistos. O controle reativo, por sua vez, é o mais simples, composto apenas por um laço de: (i) leitura de sensores; (ii) processamento de informações; (iii) comando para atuadores. No controle híbrido, ocorre uma fusão do controle deliberativo e reativo. Por exemplo, um robô móvel que usa o controle deliberativo para traçar a rota pela qual se movimentará e o controle reativo para desviar dos obstáculos no caminho. Por último, o controle hierárquico apresenta uma solução em módulos, os quais terão diferentes níveis de prioridade no sistema.

O controle que será utilizado nesse trabalho será do tipo reativo hierárquico. Módulos independentes de controle reativo do robô móvel serão dispostos em uma hierarquia vertical (BROOKS, 1896). Dessa forma, a primeira camada, ou seja, a de maior prioridade será sempre executada, fazendo a leitura dos sensores necessários e enviando comando aos atuadores. As camadas seguintes, também enviarão comandos aos atuadores, porém, com um nível de prioridade menor que a anterior. Por exemplo, um robô móvel cuja camada com prioridade máxima é “desvio de obstáculos” e possua outra camada: “vagar sem rumo”. Nesse caso, se o robô estiver vagando e um obstáculo surgir, o comando de parada, enviado pela primeira camada, será o final.

Esse projeto tenta prover um robô móvel autônomo de grande velocidade utilizando um sistema de controle reativo com decomposição vertical embarcado em uma placa de baixo custo *RaspberryPi*. O veículo terá dois motores *brushless* DC acoplados às suas rodas dianteiras e se locomoverá em velocidade próxima ao VANT Ararinha do Grupo de Interesse em Sisvants Autônomos e Aplicações. A motivação seria setornar uma plataforma 2D para testes de software e hardware de controle autônomo sem riscos de quedas antes de sua implementação na aeronave.

2. Objetivos

O objetivo deste trabalho é avaliar a aplicação da *RaspberryPi* para implementar um sistema de navegação autônoma para um robô móvel de pequeno porte e alta velocidade, tracionado por dois motores *brushless* DC sem redução, utilizados normalmente para a propulsão de aeromodelos e *drones*. Esse sistema utiliza sensores de distância por ultrassom para evitar a colisão com obstáculos e um módulo de comunicação wireless para o controle remoto. Esse controle remoto consiste somente da leitura da situação do robô e comandos de inicio e parada de navegação.

A proposta deste trabalho é oferecer uma plataforma segura que se movimente em alta velocidade para testes de características autônomas que serão implementadas em VANTs. O teste de navegação autônoma em um ambiente bidimensional é um passo essencial para a segurança de futuros testes em voo. Dessa forma, espera-se diminuir significantemente o risco de acidentes em fase experimental.

Essa plataforma é um robô terrestre de dimensões 35cmde comprimento por 25cmde largura. O veículo é tracionado por dois motores dianteiros *brushless* DC de 950 Kv (relação rpm/V) de 11,1 V, e sua direção será dada pela diferença de velocidade entre os dois motores. Para alimentar os motores, e o restante do sistema, é utilizada uma bateria de Li-Po de 2300 mAH.

O controle do sistema é feito pela placa *RaspberryPi* embarcada no sistema. Essa unidade de processamento deve ser responsável pela leitura dos sensores de distância e o envio de comandos na forma de PPM para os motores. Os comandos enviados pela *RaspberryPi*devem passar por um driver e pelos controladores eletrônicos de velocidade (ESC) dos motores. Devido à alta velocidade que o robô se movimentará, a precisão, o tempo de leitura dos sensores e o processamento das informações são avaliados neste trabalho.

Foram implementados drivers entre a saída da *RaspberryPi* e os controladores eletrônicos de velocidade. Opto-acopladores fazem parte desse módulo, de forma a isolar eletricamente os pinos de saída da placa. Uma placa de circuito impresso foi desenvolvida para os drivers dos motores e o circuito de entrada dos sensores de distância.

Para habilitar a comunicação com o robô, foi utilizado um módulo wireless padrão 802.11 conectado a uma das portas USB da *RaspberryPi*. Com esse módulo,

foipossível o acesso remoto ao sistema da placa. Uma rede privada virtual (VPN) foi criada entre o computador remoto e a placa embarcada no veículo.

O uso de sensores de distância por ultrassom também foi analisado. Foram feitos testes com a finalidade de validar a confiabilidade dos sensores em situações de alta velocidade. Seu tempo de resposta foi analisado para verificar sua viabilidade em sistemas autônomos nas condições apresentadas neste trabalho.

O algoritmo de navegação autônoma foi baseado na arquitetura de subsunção de Brooks, e avalia os dados dos sensores e enviar comandos para os motores. Com esse sistema de hierarquia vertical, foram feitos módulos independentes com diferentes prioridades. O foco do trabalho é a avaliação da segurança do sistema, para viabilizar a utilização deste robô em futuros testes de sistemas embarcados para VANTs.

2.1. Organização do documento

No terceiro capítulo deste documento, “Materiais e Métodos”, é abordado o estudo dos módulos necessários para o projeto. São descritas as ferramentas utilizadas e o embasamento teórico que possibilitou seu uso.

A implementação dos diferentes módulos do sistema é abordada no capítulo 4. Nele, são detalhados os passos do desenvolvimento do robô móvel, descrevendo os métodos utilizados para a leitura e acionamento dos seus diferentes módulos.

Os testes realizados para comprovar o funcionamento do robô são detalhados no capítulo 5. São detalhados os testes de funcionamento dos módulos independentes, os testes para aumentar a eficiência do design do robô e os testes do sistema de navegação já integrado com todos os módulos.

O sexto capítulo, “Conclusões”, traz uma discussão sobre o produto final deste trabalho, os resultados obtidos, os maiores obstáculos encontrados e as possibilidades de desenvolvimento futuro.

3. Materiais e métodos

Neste capítulo, são cobertos os temas e ferramentas estudados no decorrer do projeto. Um breve histórico da placa *RaspberryPi* será apresentado, seguido de uma introdução à biblioteca de acesso às portas de entrada e saída, *WiringPi*. É descrito o princípio de funcionamento dos sensores de distância por ultrassom, seguido pela apresentação dos motores *brushless* DC e o seus controladores eletrônicos de velocidade. Em seguida, os conceitos de modulação por largura de pulso são descritos e o driver de acionamento do controlador eletrônico de velocidade é apresentado. Por fim, os conceitos gerais da arquitetura de subsunção, base do sistema de navegação proposto, são introduzidos.

3.1. *RaspberryPi*, Linux embarcado e a biblioteca *WiringPi*

Em 2011, a Fundação *RaspberryPi* desenvolveu seu primeiro computador em placa única, a *RaspberryPi*. O objetivo era criar um computador de baixo custo para promover o estudo de ciências da computação em escolas. A fundação foi criada em 2009 no Reino Unido devido a uma preocupação de seus fundadores com o declínio no número de estudantes se candidatando a ciências da computação na Universidade de Cambridge (About | RaspberryPi, 2012). Seu objetivo era criar dois modelos de computadores, ambos de baixo custo – US$ 25,00 e US$ 35,00. Em 2012, foram lançados os modelos A e B ao público e foram feitos mais de cem mil pedidos antecipados somente no dia de lançamento (RPiBuyingGuide | ELinux, 2012). O primeiro modelo lançado foi o modelo B, mostrado na figura 2. Em seguida, a Fundação anunciou um modelo mais barato sem porta Ethernet, o modelo A.

Em Janeiro de 2012, muitas escolas no Reino Unido, públicas e privadas, já mostravam interesse na compra de placas *RaspberryPi* (Moorhead, J | The Guardian, 2012). Esse computador continua sendo de uso exclusivo para o aprendizado, e não pode ser vendido para outros fins. Hoje, existem escolas especializadas no ensino de programação, eletrônica e robótica usando *RaspberryPi* (Sobre Nós | Yadaa, 2014). A Fundação oferece tutoriais online para iniciantes e, em 2014, lançou um curso de capacitação para professores interessados em trazer a *RaspberryPi* para a sala de aula (Picademy | RaspberryPi, 2014).

Figura 2 - RaspberryPi modelo B

A *RaspberryPi* é um computador de placa única, ou seja, em uma única placa de circuito impresso, possui processador, memória, entradas e saídas e todas as características de um computador. É baseada no conceito SoC (*System On Chip*) BCM2835 da Broadcom, que inclui um processador da família ARM11 de 700 MHz, uma unidade de processamento gráfico dedicada e 512 MBytes de memória RAM em um único chip. Por apresentar uma porta Ethernet e duas portas USB, a *RaspberryPi* permite um grande número de periféricos possíveis, como mouse e teclados genéricos (RPiVerifiedPeripherals | ELinux, 2012). A placa ainda possui duas saídas de vídeo: HDMI e vídeo componente. Para a memória não-volátil e a mídia de inicialização, há uma entrada para cartões SD (*Secure Digital*).

Existem várias distribuições de Linux desenvolvidas para a *RaspberryPi*, ou seja, otimizadas para o conjunto de instruções do ARMv6 presente na placa (RPiDistributions | ELinux, 2015). A mais popular é a *Raspbian*, uma variante do *Debian Wheezy*. Essa distribuição oferece vários pacotes pré-compilados, facilitando o desenvolvimento de aplicações das mais variadas: desde uma central multimídia conectada à Internet, até o controle de hardware, como no caso deste trabalho (Raspbian | ELinux, 2014). Ainda por ser a mais popular, essa distribuição conta com maior respaldo da comunidade, o que facilita a instalação e a correção de erros no desenvolvimento de aplicações. Por esses motivos, a *Raspbian* foi a distribuição escolhida para o trabalho descrito aqui.

Com a finalidade de facilitar o acesso aos pinos de entrada e saída quando a *RaspberryPi* utiliza um sistema operacional de uso geral, foi criada uma biblioteca aberta chamada *WiringPi* (HENDERSON, G., 2013). O uso desta biblioteca permite uma programação mais natural de acesso ao hardware, como no caso de microcontroladores. A *WiringPi* se espelhou nas funções da placa Arduino, para facilitar a programação.

Devido às características descritas acima, a escolha da *RaspberryPi* para este projeto foi feita. Mais especificamente, por ser uma placa de baixo custo com processamento e memória suficientes para executar algoritmos de navegação e de tomada de decisão mais complexos que microcontroladores como o Arduino e o PIC.

3.2. Sensores de distância por som

No desenvolvimento de uma plataforma autônoma para testes de software e hardware em alta velocidade, é essencial garantir que não ocorram colisões. Para este fim, o sistema de navegação necessita de entradas sensoriais que lhe informem sua localização em relação aos obstáculos que o cercam. Os sonares cumprem esse objetivo enviando um pulso ultrassônico modulado em uma determinada direção e aguardando seu retorno. Quando um sinal com a mesma frequência e a mesma sequência de pulsos emitida é recebido, pode-se calcular a distância através do tempo que o som demorou a ser refletido de volta pelo obstáculo (ElecfreaksDatasheet, 2010).

O HC-SR04 é um sensor ultrassônico de pequenas dimensões (40×20×15 mm) e baixo custo (cerca de US$ 3,00) que funciona a uma frequência de 40 KHz. Sua entrada e saída funcionam com nível lógico TTL e é necessário um pulso de, pelo menos, 10 µs em nível alto na sua entrada (*Trigger*) para disparar o envio do pulso ultrassônico. Esse pulso será composto de um *burst* de oito ciclos a 40 kHz (ElecfreaksDatasheet, 2010). Um sensor piezoelétrico aguardará o pulso refletido, e caso o receba, comparará a frequência para garantir que aquele é o mesmo pulso que foi emitido pelo sonar. A saída (*Echo*) será um pulso em nível alto com largura proporcional ao tempo entre a emissão e a recepção do pulso ultrassônico.

Dessa forma, conhecendo a velocidade do som aproximada para as condições atmosféricas (altitude) na qual o sensor irá operar, é possível calcular a distância entre o sensor e o obstáculo através do tempo fornecido pelo pulso de saída do sonar. Portanto, fica a cargo do processador analisar a largura do pulso de saída e

determinar a distância equivalente. A figura 3 mostra o sonar. O cilindro da direita contém o trigger e o da esquerda, o sensor piezoelétrico receptor.

Figura 3 - Sensor ultrassônico HC-SR04

O HC-SR04 tem alcance de 4 m, com precisão de 3 mm (ElecfreaksDatasheet, 2010). Esse sensor é capaz de detectar obstáculos diretamente a sua frente ou até 15° para cada lado, fornecendo assim uma faixa de detecção de 30° para cada sensor. É importante, quando utilizando múltiplos sensores, separar as faixas de cada um, para evitar uma possível interferência entre os sonares. O fabricante também avisa que objetos com superfícies lisas e planas são detectados com maior facilidade.

3.3. Motores *brushless* DC e ESCs

Motores de corrente contínua sem escova, brushless DC ou simplesmente BLDC, são motores elétricos síncronos comutados eletronicamente. Esse tipo de motor ganhou popularidade rapidamente e hoje é aplicado em diversas indústrias como a de eletrodomésticos, automotiva, aeroespacial, equipamentos médicos, automação industrial e instrumentação. Os BLDC apresentam algumas vantagens quando comparados a motores DC com escovas ou motores a indução. Algumas delas são: melhor curva de velocidade × torque; alta resposta dinâmica; alta eficiência; longa vida útil; baixo ruído; e alcance de altas velocidades (Yedamale, P. | Microship, 2003).

Além dos pontos citados acima, os motores de corrente contínua sem escova apresentam outra vantagem, a qual os fez populares em aeromodelos e VANTs: eles possuem uma alta razão de torque entregue por tamanho, tornando-os muito úteis em aplicações nas quais peso e espaço são fatores críticos (Yedamale, P. | Microship, 2003).

Acima de tudo, são motores síncronos, ou seja, o campo magnético gerado pelo estator e o campo magnético gerado pelo rotor giram com a mesma frequência. Além disso, o BLDC não apresenta o escorregamento que é encontrado em motores à indução. Os motores *brushless* podem vir em configurações monofásicas, bifásicas ou trifásicas, sendo a última a mais comum (Yedamale, P. | Microship, 2013).

O estator de um BLDC é muito similar ao de um motor à indução. Ele consiste em lâminas de aço empilhadas com fendas no sentido axial por onde os enrolamentos do estator serão passados. A maior parte dos motores *brushless* DC tem três enrolamentos no estator, conectados em estrela (Y) ou em delta (Δ), sendo o primeiro o mais comum. A diferença entre os dois padrões é que, em Y, o motor fornece alto torque em baixas rotações, e em Δ, o motor fornece baixo torque em altas rotações. O rotor de um BLDC é composto de imãs permanentes e pode ter de dois a oito pares de polos com norte e sul opostos. O aumento no número de polos proporciona um maior torque, porém em detrimento da velocidade máxima que o motor alcança. Outro parâmetro importante é o material usado na construção do imã permanente. Ferrite é amplamente usado por ser um material de baixo custo, porém, para um determinado volume, apresenta densidade de fluxo magnético menor que ligas magnéticas como Nd, SmCo e NdFeB. Quanto maior a densidade de fluxo magnético, maior o torque. Dessa forma, essas ligas permitem um motor menor e mais leve com o mesmo torque que motores com rotor de Ferrite (Yedamale, P. | Microship, 2013). A figura 4 apresenta o estator e do rotor de um BLDC.

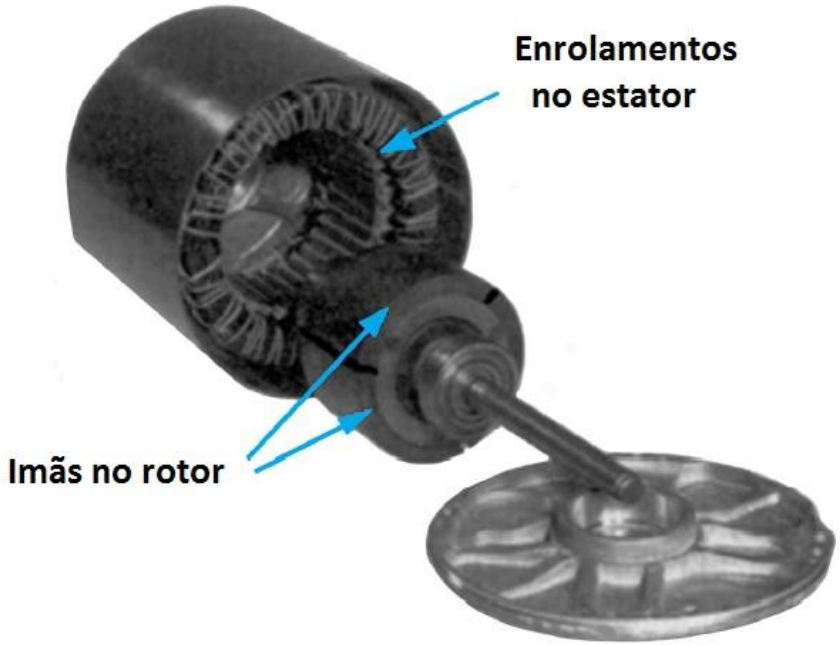

Figura 4 - Motor Brushless DC

O princípio de operação de um motor DC sem escovas é o mesmo de um motor com escovas, ou seja, há um *feedback* interno de posição do eixo, para que se saiba qual enrolamento será energizado. No caso do motor DC escovado, esse *feedback* é mecânico dado pelas escovas e pelo comutador. Já no BLDC, o *feedback* é dado pelo uso de sensores Hall na base do motor, ou pela força contra eletromotriz gerada pelo próprio motor (Yedamale, P. | Microship, 2013). Quando sensores Hall são usados, eles são, geralmente, posicionados no estator, no lado fixo do motor. Quando um pólo magnético passar por um dos sensores, esses irão enviar um sinal alto ou baixo indicando o pólo norte ou o sul. Nos motores que não usam sensores, a força eletromotriz gerada entre os enrolamentos é utilizada para a detecção da posição do rotor (Nolan, D. | ST Microelectronics, 2012).

Uma vez que a posição do eixo é conhecida, a sequência de energização pode ser iniciada. A cada comutação, uma bobina é energizada positivamente (corrente entrando no enrolamento) e outra é energizada negativamente (corrente saindo do enrolamento), e a terceira não é energizada. O torque é gerado pela interação entre o campo magnético gerado pelas bobinas do estator e o ímã permanente do rotor (Nolan, D. | ST Microelectronics, 2012). O procedimento é exemplificado no diagrama da figura 5, a seguir.

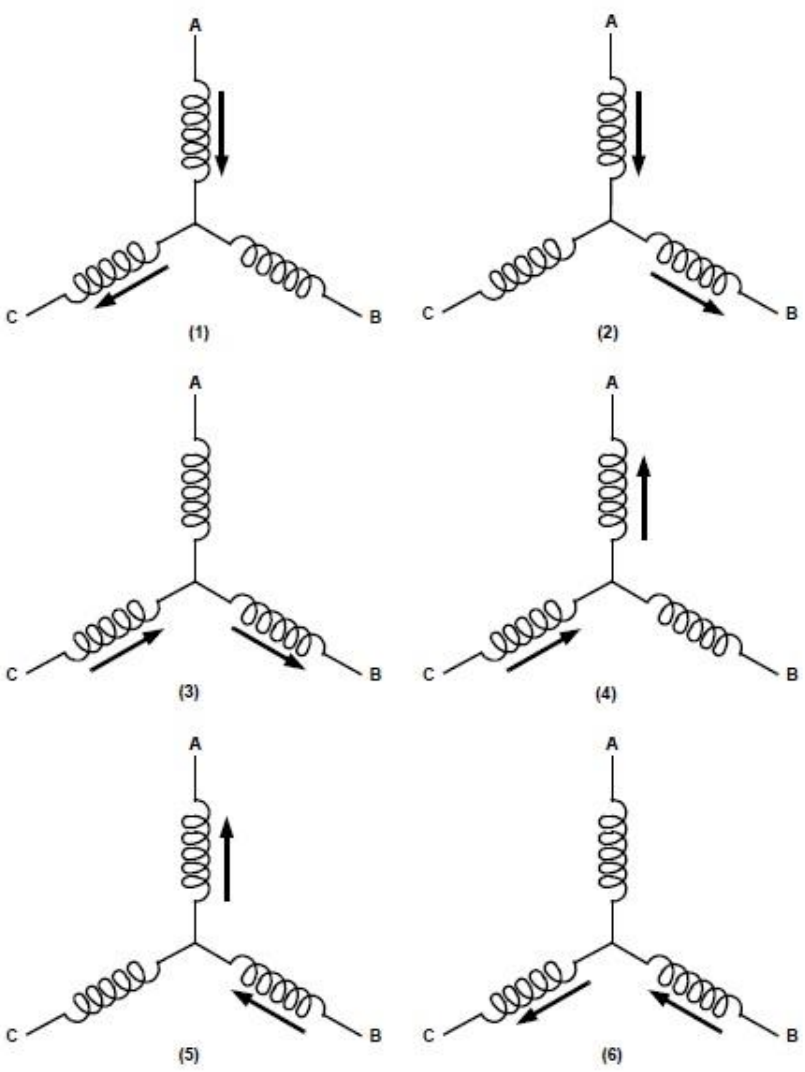

Figura 5 - Comutação dos enrolamentos de um BLDC

O agente responsável por fazer a comutação das bobinas é o controlador eletrônico de velocidade. O ESC receberá o sinal dos sensores Hall ou diretamente da força contra eletromotriz das bobinas do motor e dessa forma interpretará como iniciar a energização das bobinas. Alguns controladores podem ser programados pelo fabricante e fornecem opções para o usuário como o limite de corte para tensões baixas, aceleração e o sentido de rotação. Para regular a tensão vinda da bateria, a maior parte dos ESC modernos conta com um circuito regulador de tensão integrado (ESC Manual | Hobby King).

O controlador eletrônico de velocidade recebe, tipicamente, um sinal PWM de 50 Hz e aceita uma variação de 1 a 2 ms de nível alto para o controle da velocidade do motor. O controlador escolhido para o projeto possui pequenas dimensões (23 × 52 × 7 mm), corrente nominal 40 A e um circuito regulador de tensão especial para aeromodelos,

com a capacidade de suportar picos de demanda de carga momentânea mais altos para eliminar a possibilidade de desligamentos indesejados (ESC Manual | Hobby King).

O motor *brushless* DC escolhido para a montagem do veículo foi uma versão menos potente e com menor relação rpm × Volts que os usados nos aeromodelos fabricados pelo GISA.Os BLDC de 28 × 30 mm pesam 70g e relação volt/rpm 950 Kv foram escolhidos para tracionar o veículo. A sua potência nominal é 243W, equivalente a um empuxo de 850g. A tensão pode variar de 7.4V a 14.8V, e a corrente máxima é 23.2A. A seguir, uma foto do conjunto motor e controlador de velocidade utilizado:

Figura 6 - Motor brushless e ESC utilizados no projeto

3.4. Modulação por largura de pulso

O controlador eletrônico de velocidade requer, como entrada, o mesmo sinal de controladores de servo-motor (ESC Manual | Hobby King). Os parâmetros desse sinal são uma largura mínima do pulso, uma largura máxima, e uma taxa de repetição. Nesse caso e em muitos outros, como inversores de frequência, fontes chaveadas e controle de potência a modulação por largura de pulso, ou PWM (*pulse width modulation*), é utilizada.

O conceito de modulação por largura de pulso é simples, por exemplo, considerando a figura 7:

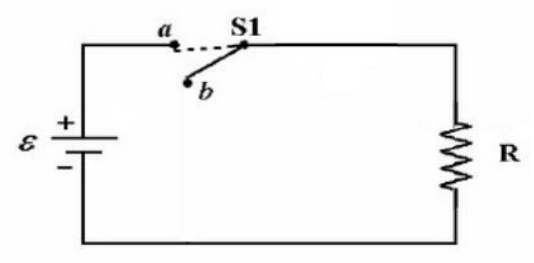

Figura 7 - Circuito chaveado com carga resistiva

Quando a chave S está aberta, não há corrente na carga R e a potência aplicada é nula. No instante que a chave S é fechada, a carga recebe a tensão total da fonte e a potência aplicada é máxima. Logo, se abrirmos e fecharmos a fonte em alta frequência, a tensão média será dada por:

$$V_{rmd} = \frac{1}{T}\int_0^T V_{R(t)}dt = \frac{1}{T}\int_0^{t1} E\,dt = \frac{t1}{T}E$$

A relação entre $t1$ e T é corresponde ao ciclo de trabalho da onda. Em aplicações de potência, o ciclo de trabalho é diretamente proporcional à potência entregue à carga.

No entanto, em servo-motores e controladores eletrônicos de velocidade, o interesse não é na tensão média fornecida. Nesse caso, é importante saber a largura do pulso em relação a um determinado período fixo. Nos servo-motores, o ângulo de rotação será definido pela largura do pulso. A maior parte de servo-motores e ESC esperam um sinal de entrada de 50 Hz, onde um pulso de 1ms representa um sinal mínimo e um pulso de 2 ms representa um sinal máximo. Como mostrado na figura 8 abaixo:

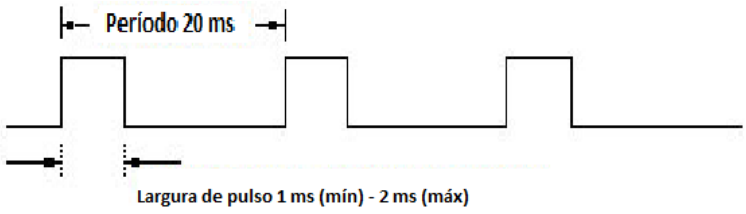

Figura 8 - Sinal de controle dos ESC

A diferença fundamental entre o PWM utilizado no controle de velocidade de um motor DC diretamente e o PWM de entrada de um ESC é que, no primeiro, a potência entregue é o fator crítico, no segundo, é a largura do pulso. Por exemplo, um servo-motor, que se movimenta a uma certa posição quando recebe um pulso de 1,5 ms a cada 6 ms, irá mostrar o mesmo ângulo de rotação caso receba um pulso de 1,5 ms a cada 25 ms. Ambas são ondas com a mesma largura de pulso, porém com períodos muito diferentes. Se essas ondas fossem o sinal de um motor DC, no primeiro caso ele

apresentaria 25% da velocidade nominal e, no segundo, somente 6% da mesma, relação diretamente correspondente ao ciclo de trabalho de cada onda.

Nos controladores eletrônicos de velocidade, uma largura de pulso mínima corresponde à velocidade mínima do motor, e uma largura de pulso máxima, à máxima velocidade. Os valores de largura máxima e mínima para o ESC utilizado neste trabalho também foram de 2ms e 1 ms, respectivamente, em um período de, aproximadamente, 20 ms.

3.5. Driver dos motores

O acoplamento dos comandos da *RaspberryPi* com o módulo de potência do robô é feito utilizando opto-acopladores e buffers. Dado que os pinos de entrada e saída da placa não apresentam nenhuma proteção contra sobre-tensão (RPiLowLevelPeripherals | ELinux), é esperado do desenvolvedor a conexão de buffers, conversores AC/DC, reguladores de tensão, entre outros. Há outro motivo para o cuidado com o acoplamento entre a *RaspberryPi* e os demais módulos do sistema, o fabricante da placa só garante operação segura até 16 mA nos pinos de entrada e saída (BCM2835 ARM Peripherals | Broadcom).

Opto acopladores, ou opto isoladores, são componentes que transferem sinais elétricos entre dois circuitos isolados usando luz. Integrados dentro do mesmo encapsulamento, encontram-se LEDs e foto-transistores, o que permite o acionamento desacoplado entre as parte (OptocouplerSolutions | Fairchild, 2015). Como um curto entre a alimentação de 5 V do sistema e um dos pinos de entrada e saída da *RaspberryPi* seria suficiente para queimar o devido pino, a isolação elétrica fornecida pelos opto acopladores foi uma vantagem significante comparado a outros drivers. Além de foto-transistores, existem opto acopladores que utilizam foto-resistores e fotodiodos, sendo a última categoria a mais rápida (4N25 | VishaySemiconductors, 2004). Existem ainda configurações com dois LEDs, fazendo uma ligação bidirecional. Contrária às outras configurações onde um sinal na entrada acende um LED que ativará um foto- transistor (diodo ou resistor) na saída, na configuração bidirecional os dois LEDs são sensíveis à luz produzida pelo outro. A figura 9 mostra o princípio de funcionamento de um opto-acoplador unidirecional com um LED e um foto-transistor.

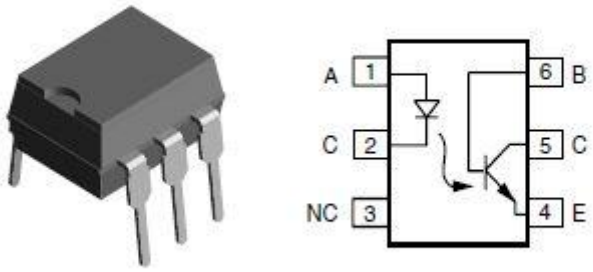

Figura 9 - Optoacoplador formado por um LED e um foto-transistor

Outro componente importante para o acoplamento dos comandos da *RaspberryPi* com os demais módulos do robô é o buffer. Um buffer é um dispositivo usado para transferir tensão de um circuito com impedância de saída alta para um segundo circuito com impedância de entrada baixa. Desta forma, um buffer previne o segundo circuito de carregar o primeiro, interferindo com a sua operação. Um buffer ideal teria uma resistência de entrada infinita e resistência de saída nula. Logo, são utilizados buffers no projeto, pois ajudam a proteger os pinos da placa e, com eles, é possível o fornecimento de correntes mais altas que são inviáveis somente com a *RaspberryPi*.

Buffers podem ser feitos com amplificadores operacionais, como na figura 10. Caso a tensão de saída seja igual à de entrada, ou seja, um amplificador de ganho unitário, tem-se um seguidor de tensão. Mesmo a tensão sendo a mesma, o ganho de corrente é grande, fornecendo, dessa forma, um ganho de potência. Como a impedância de entrada do amplificador é muito alta, ele pode puxar correntes altas sem carregar o circuito que precede o buffer.

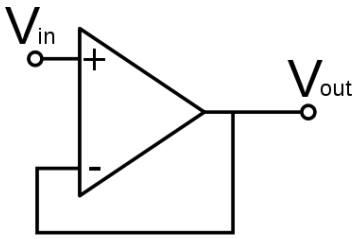

Figura 10 - Amplificador operacional usado como buffer

3.6. Arquitetura de Subsunção

Um veículo autônomo, como o proposto neste trabalho, está fadado a encontrar diversos obstáculos não previstos em seu projeto. Por esse motivo, um controle deliberativo pode apresentar desvantagens frente a situações não esperadas. Portanto, o controle instituído foi um controle reativo.A arquitetura de subsunção (BROOKS, 1986) se diferencia de outros modelos de comportamentos de robôs, pois foge da estrutura tradicional de percepção do ambiente:

$$leitura\ de\ sensores \rightarrow modelagem \rightarrow planejamento \rightarrow controle\ de\ atuadores$$

Preocupado com a resposta em tempo real de um sistema que tenta adquirir todas as informações possíveis do ambiente para então planejar uma execução, como a estrutura apresentada na figura 11, Brooks sugere a arquitetura de subsunção. Nesse modelo, módulos de cada tarefa que o robô precisa realizar tem acesso à leitura dos sensores e ao controle dos atuadores, porém, com prioridades diferentes. Portanto, considera-se essa arquitetura um controle reativo hierárquico.

Para ilustrar esse conceito, consideremos um veículo autônomo simples com dois motores elétricos, um módulo GPS, um compasso eletrônico e sensores de distância. Suponhamos que ele tenha que chegar a uma determinada posição dada pela coordenada GPS, porém, existam obstáculos no caminho. Nessa situação, o modelo tradicional fará a leitura dos sensores de distância e da coordenada GPS, e em seguida, fará o planejamento de acordo com todas as informações disponíveis. Um sistema de navegação com arquitetura de subsunção para o robô descrito acima teria dois módulos com prioridades diferentes. O primeiro utiliza as informações dos sensores de distância para efetuar uma lógica de desvio de obstáculo, caso haja algum. O segundo módulo traçará a menor rota entre a coordenada GPS atual e destino, utilizando o compasso eletrônico. Ambos módulos nesse caso trabalham paralelamente, tem acesso aos sensores e podem enviar comandos aos motores. No entanto, seus comandos terão pesos diferentes, visto que o módulo de desvio de obstáculos deve ter uma prioridade maior. A figura 12 mostra dois módulos enviando sinais de controle para os motores. Em um primeiro momento, o módulo com maior prioridade não é suprimido, e no esquema abaixo, ele é suprimido pelo segundo módulo.

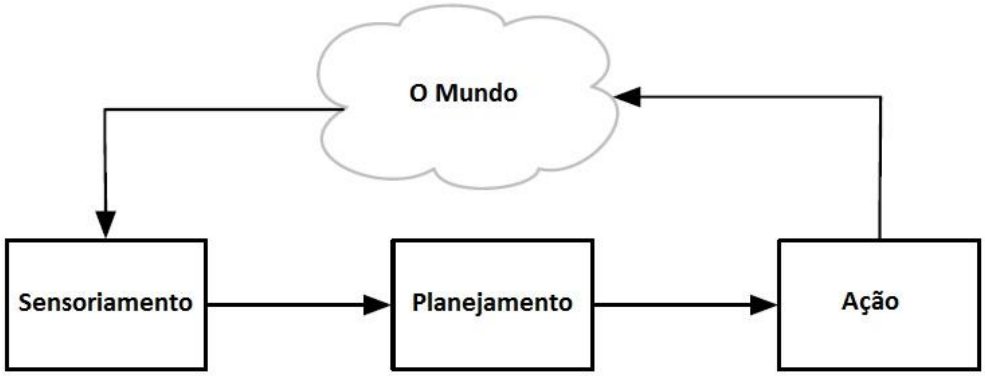

Figura 11 - Estrutura tradicional de navegação

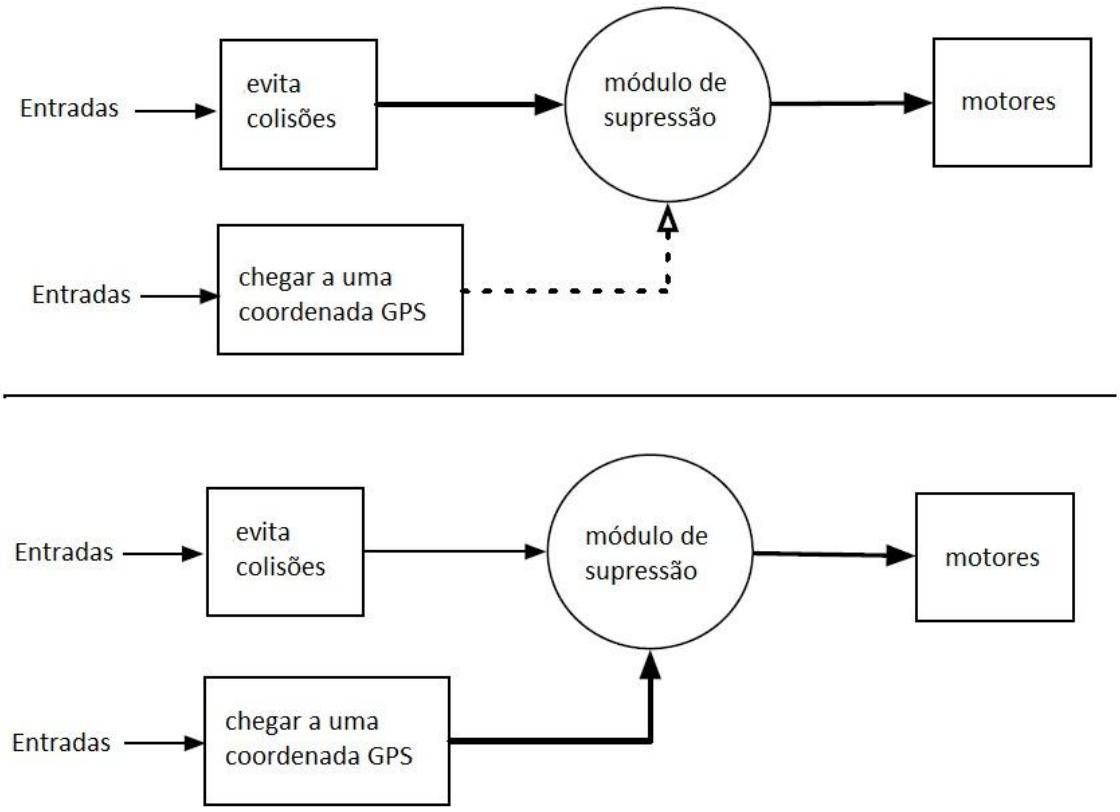

Figura 12 - Estrutura de subsunção. Módulos podem suprimir a saída de outros

Uma situação onde o robô deve executar mais tarefas, comparadas a somente se locomover até um determinado destino, revelará ainda mais a importância de uma arquitetura com diversos módulos, todos focados em tarefas,que são processados de forma paralela. O comportamento final do robô é dividido em camadas de competência, e cada camada é responsável por uma atividade do robô. Os

comportamentos de cada camada funcionam de forma concorrente e independente. Camadas com maior prioridade suprimem ou até inibem completamente o comportamento de outras camadas, seguindo a hierarquia projetada.

A arquitetura de subsunção endereça pontos muito importantes no desenvolvimento de um robô autônomo: múltiplas metas, múltiplos sensores, robustez e aditividade. Devido a sua estrutura de camadas independentes, o robô pode trabalhar no cumprimento de múltiplas tarefas sendo executadas simultaneamente. No caso de metas concorrentes, pode-se esperar o resultado de todas as camadas para a decisão do comportamento. As camadas distintas também tornam o tratamento de múltiplos sensores transparente. Dado que nem todos os comportamentos do robô necessitam da leitura de todos os sensores, os módulos podem fazer a leitura somente das variáveis relevantes. Ademais, mais de um comportamento pode requerer dados de um mesmo sensor e processá-los de forma diferente e concomitante. A robustez advém do fato que, uma vez implementada uma camada inferior – como desvio de obstáculos, a adição de uma camada superior não interferirá com os comportamentos prévios já estabelecidos do sistema. Isso acontece, pois uma camada superior não interfere nas inferiores, somente sua saída pode suprimir a saída das demais. Por fim, a possibilidade de adicionar um novo comportamento, ou uma nova meta, a um robô funcional sem interferir com resultados de comportamento já obtidos é uma vantagem considerável durante o projeto do robô.

4. Implementação do sistema de controle e navegação

O hardware utilizado nesse projeto é composto de módulos. Esses módulos permitem o sensoriamento, o controle e a comunicação sem fio entre o robô e o computador responsável pela telemetria. Todos os módulos foram testados e comprovados separadamente para que fosse possível uma montagem segura do robô. Nessa seção, serão descritos os passos no desenvolvimento dos módulos e do sistema de navegação, já incorporando os módulos de sensoriamento, controle e comunicação.

4.1. Sonares

O módulo de sensoriamento do veículo é composto por três sensores de distância por ultrassom HC-SR04. O HC-SR04 é um tipo de sonar de baixo custo e curto alcance, somente 400 cm (ElecfreaksDatasheet, 2010). A dificuldade a ser transposta no acoplamento dos sonares com a RaspberryPi foi a diferença na tensão de operação de ambos. A *RaspberryPi* opera com 3,3 V e os HC-SR04 com 5 V. A interface entre ambos é feito de forma que a *RaspberryPi* envia um sinal de gatilho, ou TRIGGER, e recebe uma resposta do sonar, o ECHO, correspondente ao tempo que o som levou a chegar no obstáculo, refletir e ser reconhecido pelo sensor na volta.

Na folha de dados do sensor, é verificado que ele opera com níveis lógicos TTL.Isso possibilitou a conexão direta do pino de saída da placa à entrada gatilho do sensor. Caso o sonar operasse com nível lógico CMOS essa ligação direta talvez não fosse possível, visto e o nível lógico alto seria 3,33V e a tensão fornecida nos pinos de saída da *RaspberryPi* fica um pouco abaixo dos 3,3V nominais.

Tabela 1 - Níveis lógicos CMOS e TTL

Tecnologia	**Nível Baixo**	**Nível Alto**	**Nota**
CMOS	0V a 1/3 V_{DD}	2/3 V_{DD} a V_{DD}	V_{DD} = alimentação
TTL	0V a 0,8V	2V a V_{CC}	V_{CC} = 5V ± 10%

Uma vez solucionado a ligação do gatilho, faltou garantir que o pino de saída do HC-SR04 não enviasse 5V diretamente à placa, o que danificaria o pino de entrada, dado que os pinos da *RaspberryPi* não possuem proteção contra sobre-tensão. Portanto, foi feito um divisor resistivo, visto que a entrada digital da placa é de alta impedância e não drenaria um valor de corrente que pudesse alterar o valor lido pelo pino. A figura

abaixo apresenta o circuito utilizado para alimentar os sonares e fazer sua interface com a placa.

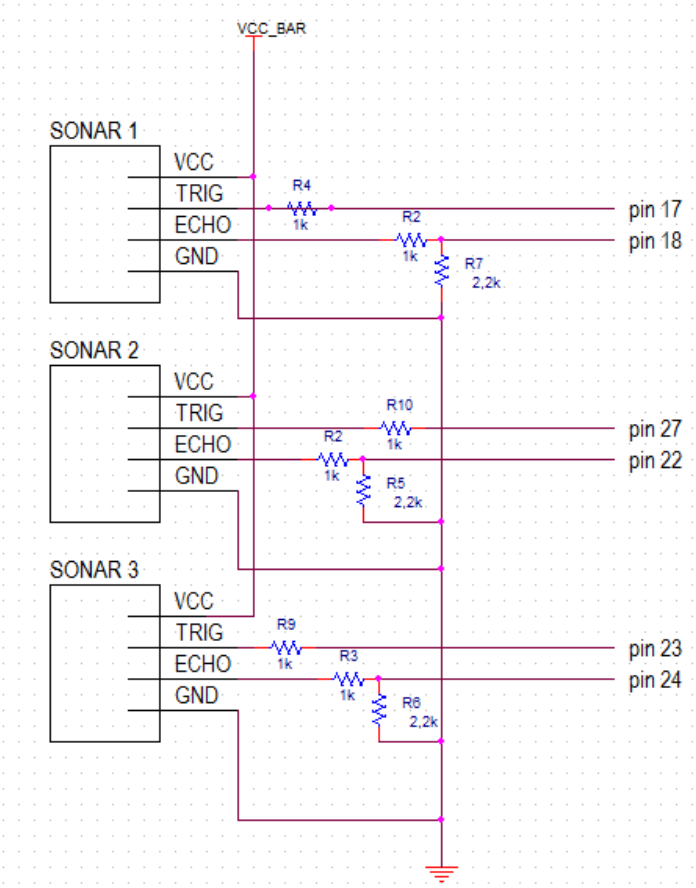

Figura 13 - Circuito de acoplamento dos Sonares

O funcionamento especificado para o HC-SR04 na folha de dados do componente requer um pulso de, no mínimo, 10 microssegundos para ativar o gatilho do sonar. Após o término do pulso no pino TRIGGER, o sonar enviará um sinal sonoro de oito ciclos com frequência 40kHz. Uma vez emitido o sinal sonoro, o sensor fica no aguardo dos pulsos refletidos. Quando o sensor reconhece que recebeu o pulso enviado, ele leva a saída ECHO para nível lógico alto e mantém alto pelo mesmo intervalo de tempo decorrido entre a emissão e recepção do pulso ultrassônico. Fica a cargo da unidade de processamento contar o tempo do pulso de saída do sonar e calcular, dada a velocidade do som no ar, qual foi a distância percorrida pela onda. O sinal obtido será correspondente ao dobro da distância entre o sensor e o obstáculo, pois corresponderá ao tempo de ida e volta do pulso. O diagrama a seguir representa o funcionamento dos sonares.

4.1.1. Tempo de resposta dos sonares

Uma preocupação quanto ao uso de sonares em uma plataforma para testes em alta velocidade é o tempo de resposta dos sensores. Sabendo que o alcance do HC-SR04 é curto, e o robô estará em alta velocidade, a janela de tempo entre reconhecer um obstáculo e atuar de forma eficiente é um fator crítico para o sucesso do projeto.

Será explicado na próxima sessão o porquê da leitura intercalada dos sonares, por agora, será considerado o caso de um obstáculo que entra no alcance dos sensores. Ou seja, há um objeto a 4m do robô se quer saber quanto tempo será gasto até que comandos possam ser enviados para os motores e efetuar uma operação de parada ou desvio. Cada sonar precisa de um trigger de 10 microssegundos e a velocidade do som no ar será aproximada para 340m/s. O pior cenário é quando os três sensores retornam 4m. Lembrando que o tempo do pulso é correspondente ao dobro da distância lida:

$$cada\ sensor: 10\ \mu s + \frac{8m}{340m/s} = \ 0{,}02354s$$

$$tempo\ para\ três\ sensores: 3 \times 0{,}02354s = \ 0{,}07061s$$

Dado que o processador usado na placa *RaspberryPi* opera a 700Mhz e a frequência da leitura dos sonares no pior caso é de 14,3Hz, o tempo de processamento da resposta dos sonares é desprezível. Logo, em menos de 0,1 segundo a placa já pode enviar comandos para os motores.

4.1.2. Interferência

Com o objetivo de reduzir o tempo de resposta dos sensores, foi considerada a leitura simultânea dos sonares. Para isso, foram efetuados dois testes diferentes. O primeiro para descobrir se o sinal de gatilho de um sensor afetaria a leitura de outro sensor posicionado ao lado. O segundo teste tentava verificar o efeito da resposta de um sonar na leitura de outros.

No primeiro teste, foram colocados dois sonares lado a lado. Em seguida, foram enviados sinais de gatilho idêntico para os dois sonares, porém lido somente o ECHO do primeiro. Novamente, foram enviados sinais de TRIGGER para ambos, mas somente feita a leitura da resposta do segundo. Ao comparar o resultado deste teste com a leitura intercalada normal, enviando o pulso de gatilho somente para o ultrassom que será lido, não foram observadas discrepâncias. Este teste demonstrou

que o pulso de ultrassom emitido por um sonar não é lido como a resposta de um segundo sonar colocado a menos de 5 cm do primeiro.

Já no segundo teste, foram enviados os pulsos de gatilho ao mesmo tempo, e o programa aguardou o primeiro sinal de resposta para iniciar a contagem de tempo. Desta forma, o programa calcularia a distância do primeiro sinal de resposta, em seguida o segundo e o terceiro. Porém, ao efetuar esses testes, estima-se que durante a leitura de um sinal de ECHO, perde-se a condição de mudança de nível lógico dos outros, o que inviabilizou o teste. Desta forma, os sensores continuaram a ser intercalados. Enviando o gatilho do primeiro, aguardando sua resposta, e em seguida, o segundo e o último.

4.1.3. Ângulo dos Sonares

Uma consideração importante quando usando múltiplos sonares é o ângulo que o sensor piezoelétrico consegue por reconhecer o pulso de 40kHz refletido em objetos planos, como uma parede. A folha de dados do HC-SR04 indica que o sensor responde bem para obstáculos em um campo de 30° de abertura. Ou seja, 15° para cada lado do sensor.

Foram feitos testes práticos (figura 14) a fim de comprovar o melhor ângulo para o posicionamento dos sensores no robô. Para isso, o robô foi colocado à frente de uma parede em duas distâncias diferentes, e para cada uma, variou-se o ângulo de ataque do robô a partir de 0°. O passo desse experimento foi de 10°, e notou-se que a partir de 20° para distâncias maiores que 1 metro, o sensor retornava valores muito diferentes da distância real. Para cada situação foram feitos três testes, cada um com 100 medidas de distância. Os dados recolhidos são apresentados na tabela 2, a seguir:

Tabela 2 - Resposta do sonar para diferentes ângulos dos obstáculos

Distância	**70cm**			**135cm**		
Ângulo	0°	10°	20°	0°	10°	20°
Média	70,00	70,32	73,46	135,01	134,46	2162,97
Desvio padrão	0,02	0,46	1,73	0,03	0,66	0,10

Foi verificado que, para as distâncias até um metro, o sensor poderia estar atéa 45° do obstáculo que ainda apresentaria resposta com desvio padrão baixo e a média das

medidas próxima à distância real, como pode-se observar na tabela 3. Porém, a partir de 1,35 metro de distância, as medidas a partir de 20° apresentavam média com mais de 100% de erro da distância real.

Tabela 3 - Resposta do sonar a 1m de uma parede

Ângulo	**0°**	**10°**	**20°**	**30°**	**40°**	**45°**	**60°**
Média	100	100	100,25	101	102,8333	103,5833	167,6667
Desvio padrão	0	0	0,375	0	0,277778	0,583333	0,444444

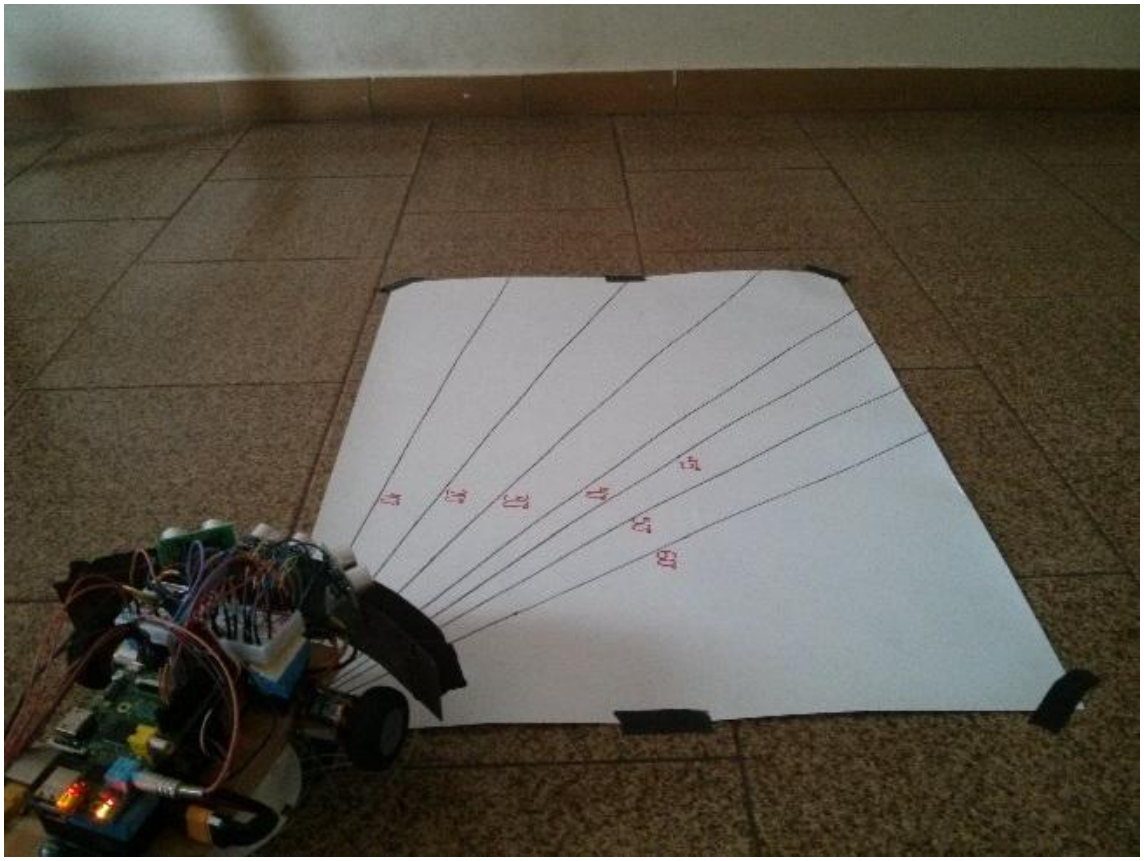

Figura 14- Teste dos ângulos dos sonares

Portanto, esse teste definiu o ângulo dos sensores ultrassônicos no para-choque do robô, de forma a abranger o maior ângulo possível no total sem ter pontos cegos. Os sensores da direita e da esquerda foram então colocados a 30° do sensor central. Espera-se poder medir com precisão de até 90% obstáculos em um arco de 90° da frente do robô, com 4m de raio.

4.2. Acionamento dos motores com o Arduino

Antes do acionamento do módulo dos motores com a *RaspberryPi*, foi efetuado um teste com a placa Arduino, seguindo o exemplo de controle de um ESC de um motor *brushless* de Chiaretta, S. (Novembro, 2012). Seguindo o exemplo, é usada a biblioteca de controle de servo-motores para o controle do controlador eletrônico de velocidade. Essa biblioteca, servo.h (Servo Library | ArduinoReference), tem uma função que envia ângulos de 0 a 180 para o motor.

Dessa forma, com um Arduino e um osciloscópio, foi feita a ligação com o controlador de velocidade e o BLDC. Foi feito um programa que permitia a variação do sinal, sinal que seria um ângulo para um servo-motor, porém corresponde à velocidade do BLDC quando ligado ao ESC.Esse teste permitiu que fossem medidos no osciloscópio quais os comprimentos de pulso e frequência que permitiam a direção do motor.

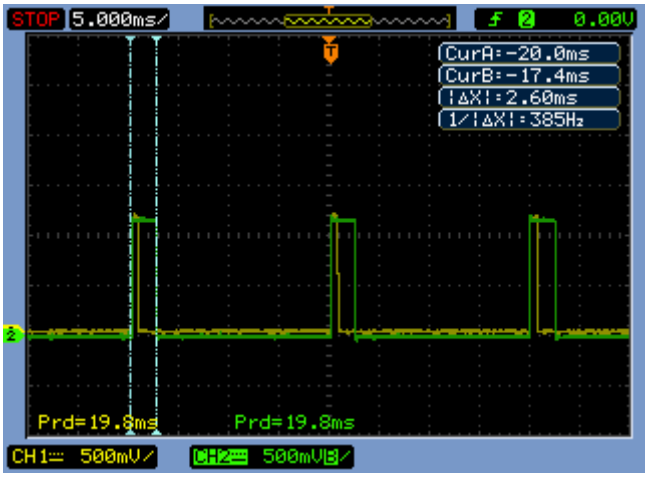

Figura 15 - Sinal de PWM gerado pelo Arduino

Na figura 15 são mostrados os sinais que são enviados para o ESC correspondentes às velocidades mínima e máxima do motor. O sinal amarelo corresponde à mínima velocidade do motor e o verde à máxima. Conhecendo o sinal necessário para dirigir os motores, o passo seguinte foi obter a mesma onda na *RaspberryPi*.

4.3. Acionamento dos motores com a *RaspberryPi*

Uma vez conhecido qual era o sinal de PWM que deveria ser enviado pela *RaspberryPi*, procurou-se implementá-lo. Como a placa só possui um pino que pode ser programado com um PWM por hardware, a primeira solução proposta foi programar os dois sinais PWM por software. Entretanto, foi possível usar a saída de

áudio estéreo da placa e com os sinais PWM gerados por hardware controlar ambos os motores (GRANA, A. Dezembro, 2012).

4.3.1. Controle do PWM por software

O acesso aos pinos de entrada e saída digital da *RaspberryPi* é feito com a biblioteca *WiringPi* e outra biblioteca desenvolvida por Gordon Henderson, a *softPwm* (HENDERSON, G., 2012). Com ambas as bibliotecas em um mesmo programa, é possível gerar um sinal PWM em qualquer um dos pinos da *RaspberryPi*. Porém, algumas limitações devem ser consideradas para manter um baixo uso da CPU. Por exemplo, é recomendado um pulso com largura mínima de 100 microssegundos. Outra variável que o programador pode escolher é a variação, ou seja, o passo quese pode variar a onda. O valor padrão de variação é 100, dessa forma, pode-se variar o nível alto da onda entre 0 e 100. No caso da variação ser 100 (que é o recomendado), um pulso de largura 100 microssegundos resulta em uma frequência máxima de 100Hz. Caso o pulso tenha largura menor que 100 microssegundos, o programa fará o atraso em loops de software, aumentando o uso da CPU. E caso seja necessário mais de um sinal nessas condições, o uso da CPU também será comprometido.

Outra consideração a ser feita é que o controlador eletrônico de velocidade opera a 5V e os pinos de entrada e saída digital da *RaspberryPi* operam a 3,3V. Nesse caso, foram usados buffers com pinos de habilitação (*enable*), 74HC244 (NXP SemiconductorsDatasheet, 2012). O buffer é alimentado com 5V do ESC e os pinos da *RaspberryPi* são conectados aos pinos de habilitação do buffer, os quais operam sob a lógica TTL, ou seja, atingem nível alto acima de 2V.

Os testes realizados com os motores sendo acionados por um sinal gerado por software funcionaram corretamente, e não afetaram a leitura dos sensores ultrassônicos, os quais utilizam o *clock* da CPU para determinar a largura do pulso ECHO dos sonares. Porém, o custo desse método foi um passo de variação menor, ou seja, a velocidade dos motores poderia assumir menos valores do que no Arduino, onde o PWM era gerado por hardware. Ademais a onda gerada apresentava maior ruído, como podemos ver na figura 16 a seguir:

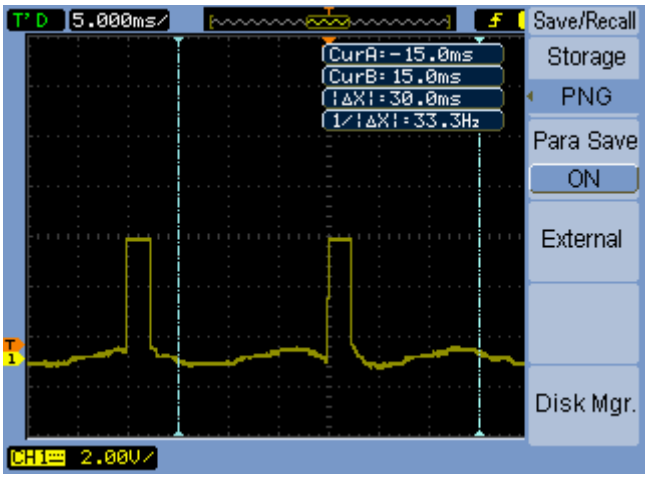

Figura 16 - Sinal PWM gerado por software com a RaspberryPi

4.3.2. Controle do PWM por hardware

Apesar de a placa *RaspberryPi* possuir somente um pino que gera sinal PWM por hardware, existe um meio de contornar essa limitação utilizando a saída de áudio da placa (GRANA, A. Dezembro, 2012). O sinal de áudio da *RaspberryPi* é, essencialmente, formado por dois PWMs, dado que é uma saída estéreo. Estudando o esquemático da placa, pode-se notar que os pinos GPIO 40 e 45 do ARM2835 estão conectados a saída de áudio.

O objetivo dessa abordagem é controlar os motores com os sinais gerados originalmente para áudio. De forma que na saída do conector P2, tipicamente usado em fones de ouvido, encontrem-se os sinais modulados por largura de pulso. Porém, entre os pinos 40 e 45 do processador e a saída, existem filtros de faixa de áudio. A figura 16mostra o esquemático de saída de áudio.

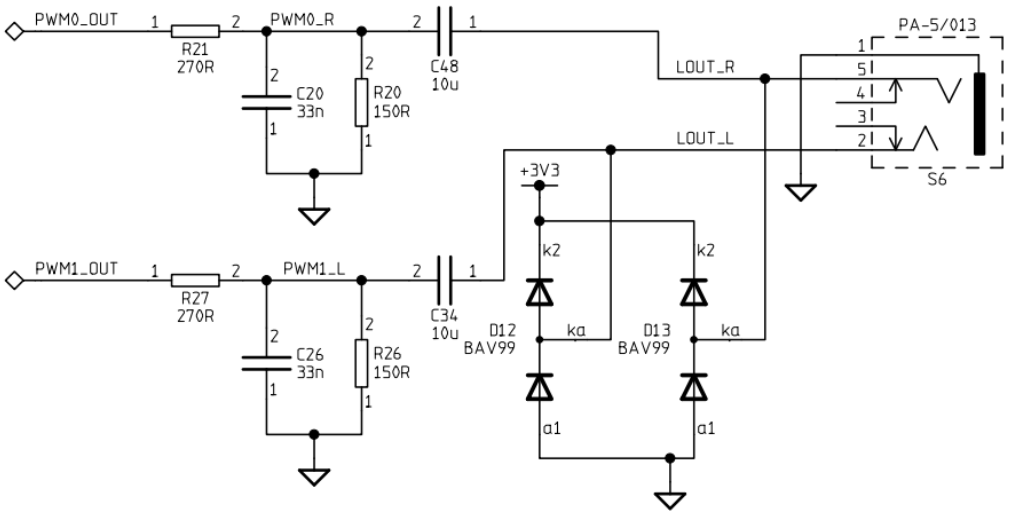

Figura 17 - Esquemático da saída de áudio da RaspberryPi (PBL., 2012)

Pode-se notar no esquemático acima que, entre a saída do processador e o conector P2, existem filtros passa-baixa e passa-alta em série. Os capacitores de acoplamento do filtro passa-alta, C34 e C48, fazem com que o nível DC do sinal seja levado a zero. Essa condição impossibilitaria um sinal de PWM. Portanto, fez-se necessária uma modificação na placa com a finalidade de desabilitar os capacitores C34 e C48. Sem removê-los da placa, usou-se um loop de fio para curto-circuitar seus dois terminais.

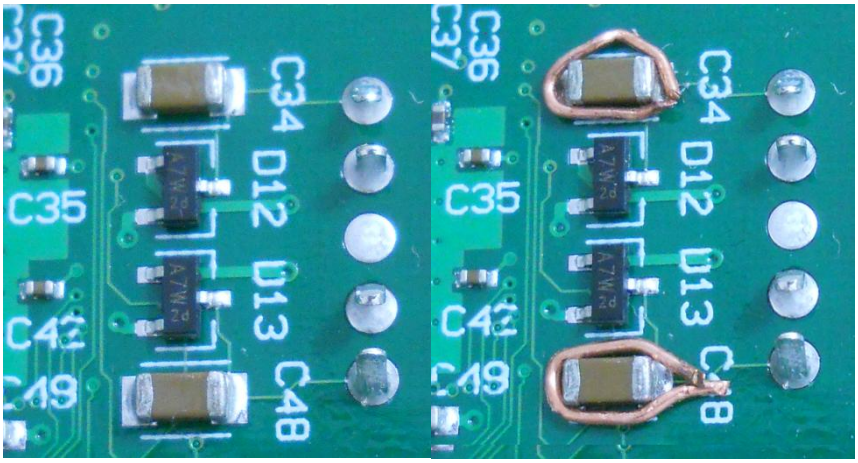

Figura 18 - À esquerda, os capacitores do filtro passa-alta C34 e C48 na placa. À direita, a adaptação feita pra desabilitá-los (GRANA, A., 2012)

Após a modificação da placa apresentada na figura 18, verificou-se se a frequência do pulso necessário para controlar os ESC seria cortada pelo filtro passa-baixa, o que levaria a uma modificação adicional. Porém a onda obtida na saída estava dentro da faixa de passagem do filtro. O sinal obtido na saída deste teste foi uma onda quadradade frequência 50Hz e amplitude 1,2V. A onda gerada por esse método se

mostrou menos ruidosa que a gerada por software, como podemos ver na figura a seguir:

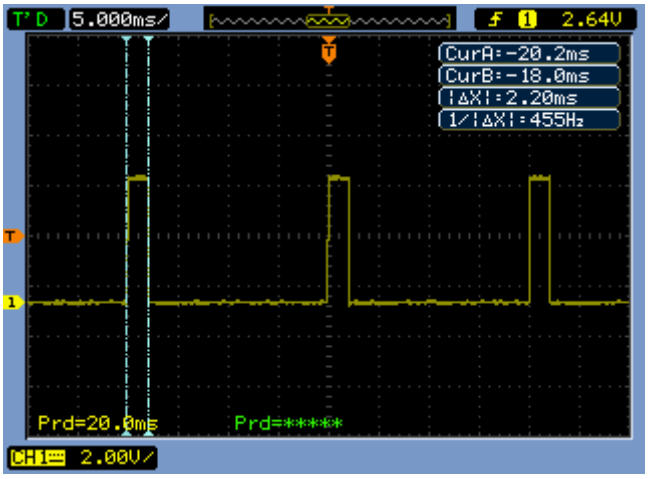

Figura 19 - Sinal gerado pela RaspberryPi por hardware após passar pelo optoacoplador

A figura 19mostra a onda na saída do circuito de acoplamento.O acoplamento do sinal obtido com a saída de áudio da placa e os controladores eletrônicos de velocidade foi feito com optoacopladores. Para garantir que a corrente necessária para acender o LED do optoacoplador fosse atingida, foi usado um transistor bipolar de junção como um amplificador linear entre o sinal da *RaspberryPi* e o LED do optoacoplador. O esquema da conexão entre a unidade de processamento e os ESC é dado na figura 20.

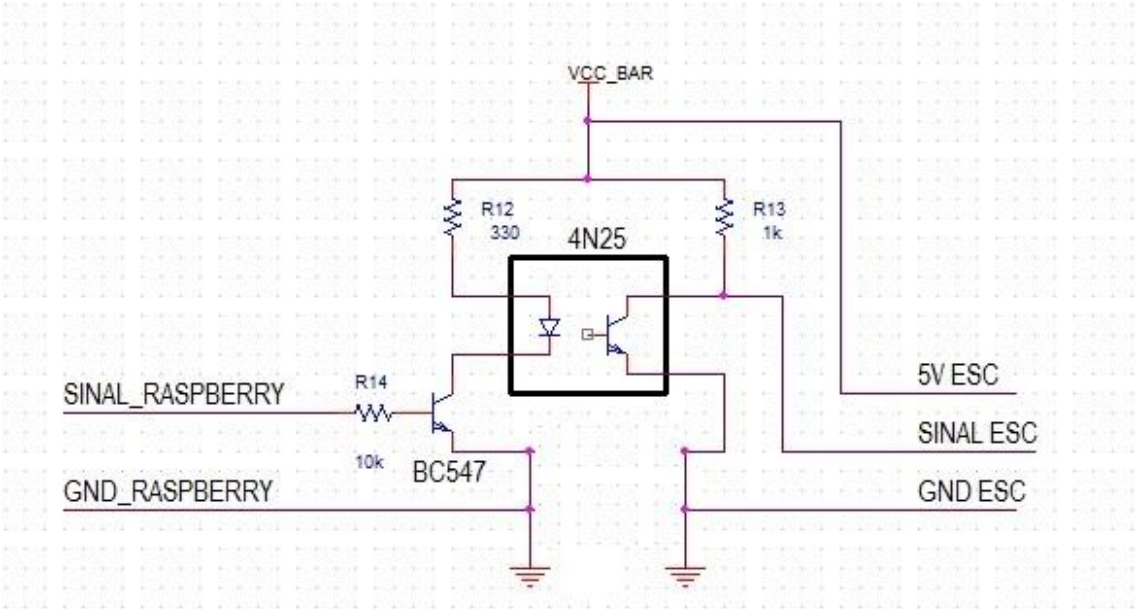

Figura 20 - Circuito de acoplamento dos ESC utilizando o optoacoplador

Os controladores eletrônicos de velocidade usados neste projeto fornecem uma tensão regulada de 5V, que foi utilizada para alimentar o circuito acima. Antes de ligar a saída dos optoacopladores aos ESC, a onda obtida foi analisada e comparada com os sinais enviados pelo Arduino e pela *RaspberryPi* com o PWM gerado por software. O controle de velocidade dos motores com o sinal da saída de áudio da *RaspberryPi* se provou satisfatória e com duas grandes vantagens em relação ao outro método. Ao usar o hardware para gerar o sinal, não é consumida constantemente capacidade de processamento da CPU. Ademais, consegue-se um menor passo de variação de velocidade do motor, podendo assim controla-lo de forma mais sutil.

4.4. Módulo de comunicação wireless

A *RaspberryPi* usa distribuições de Linux, logo, é possível fazer o acesso remoto do terminal da placa por SSH (*secureshell*). SSH é um protocolo da suíte TCP/IP da camada de aplicação. Ele permite a administração remota de servidores tipo Unix. No caso de sistemas operacionais Windows, o software PuTTY é normalmente utilizado para fazer a conexão.

A envio de comandos para a *RaspberryPi* e a telemetria do robô foram feitos por um computador utilizando o software PuTTY para acessar o terminal da placa embarcada por SSH. Essa solução só é viável se ambos a *RaspberryPi* e o computador estiverem conectados a alguma rede. Foi adquirido então o módulo wireless N300 da Encore Electronics, apresentado na figura 21. Ele possui entrada USB e modo de instalação *plug& play*, ou seja, não é necessário um driver específico para sua instalação.

Figura 21 - Módulo adaptador wireless N300 da Encore Electronics

4.5. Sistema de navegação

Uma vez que os módulos de acionamento dos motores, de sensoriamento e de comunicação foram testados e sua operação em conjunto com a *RaspberryPi* foi verificada, um sistema de navegação baseado na arquitetura de subsunção foi desenvolvido. A escolha dessa arquitetura é importante, pois permite que projetos futuros adicionem módulos sem afetar a plataforma existente e interferir na segurança do controlador de mais baixo nível, que provê o desvio de obstáculos.

Dessa forma, um sistema de navegação simples com três módulos foi montado. A diferença da arquitetura proposta para a clássica é que cada módulo irá notificar que certo comportamento deve acontecer e não vai produzir aquele comportamento diretamente. No sistema desenvolvido o módulo de mais alta prioridade é a parada. Caso os sensores indiquem que haja um obstáculo a uma distância inferior a 50cm a frente, o robô será parado. O próximo módulo é o de desvio de obstáculos, caso um obstáculo seja detectado pelos sensores, esse módulo tentará efetuar uma manobra de desvio. Como o módulo de desvio tem um nível de prioridade mais baixo, os comandos enviados por ele podem ser suprimidos pelos comandos enviados pelo módulo de parada. O último módulo, com menor prioridade, é o de seguir em frente. Dessa forma, o robô só seguirá adiante se nenhum dos módulos com maior prioridade for ativado. O diagrama da figura 22 mostra o sistema de navegação implementado.

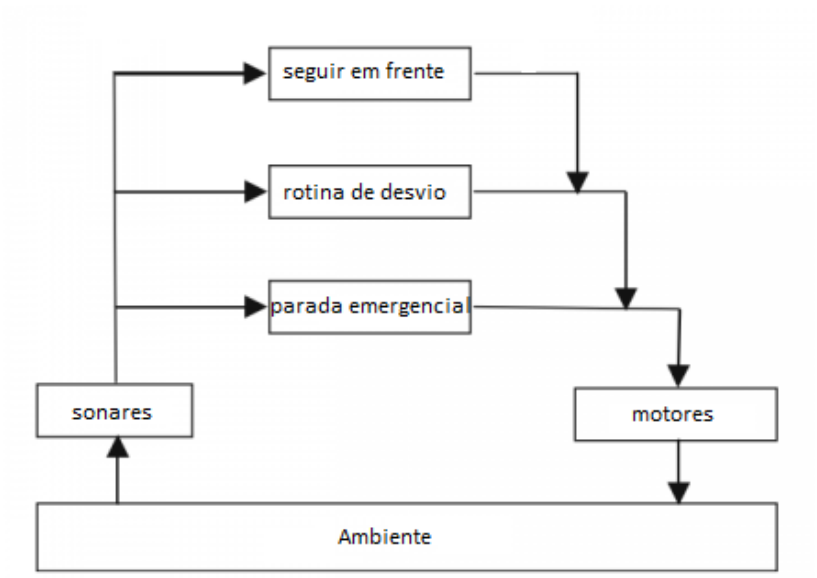

Figura 22 - Diagrama do sistema de navegação desenvolvido

4.6. Montagem do robô

O robô deve ser uma plataforma móvel que permita o suporte de todos os módulos descritos acima. Mais especificamente, ele deve carregar uma bateria de 12.6V, dois controladores eletrônicos de velocidade de 40A, dois motores *brushless* de 950Kv e as rodas de espuma de 4.5cm de diâmetro acopladas a eles, três sensores de distância por ultrassom, a placa de circuito impresso com os circuitos de acoplamento, o módulo de conexão wireless e a *RaspberryPi*. Buscando acomodar todos os componentes descritos e os adicionais, roda traseira e para-choque, montou-se o robô em uma plataforma de madeira de dimensão 20cm x 9cm x 2cm. O esquema abaixo mostra a plataforma sobre a qual os demais componentes foram dispostos. O para-choque é feito de uma chapa de aço e uma camada de EVA colada à sua frente. A figura 23 apresenta um diagrama do chassi desenvolvido.

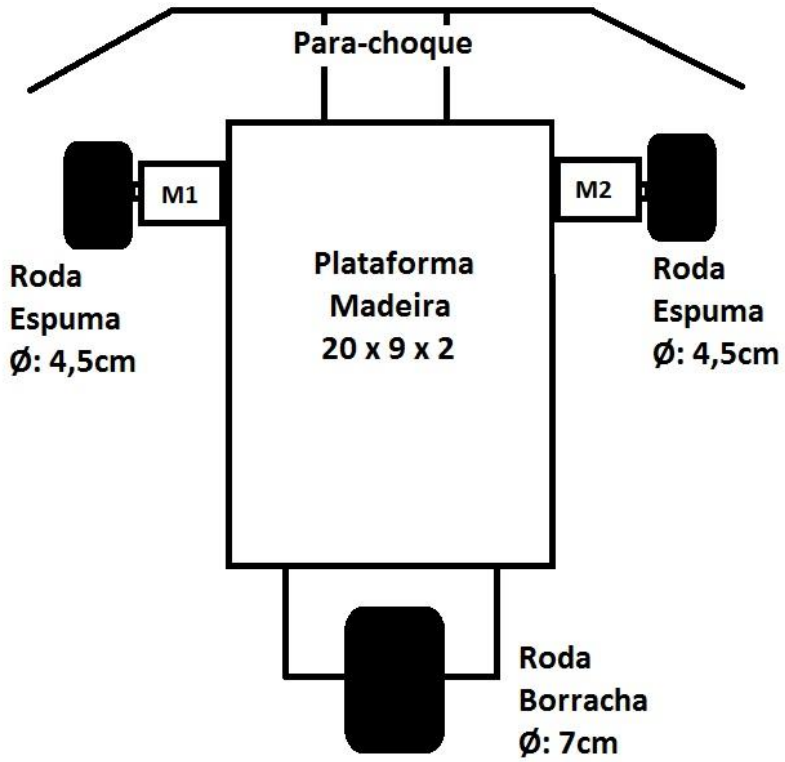

Figura 23 - Diagrama do chassi do robô

Após o chassi do robô, foi montado o módulo de potência. Esse consistia das baterias e dos dois ESC. A bateria foi colada diretamente na madeira e cada ESC foi colado em uma lateral da bateria. Foi utilizada fita adesiva dupla face para esse procedimento. O esquema da figura 24 mostra o “módulo de potência” do robô.

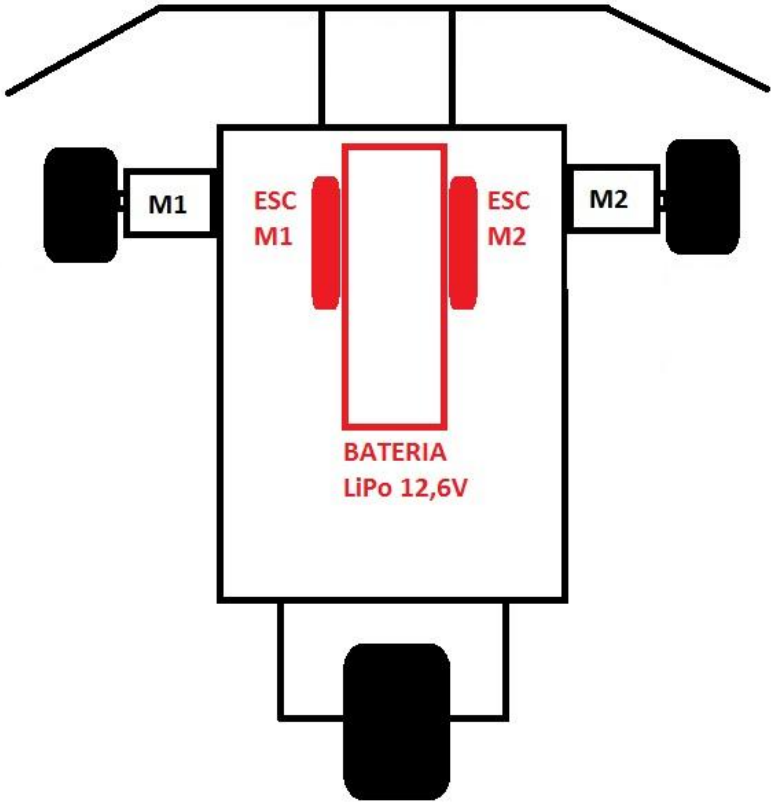

Figura 24 - Diagrama da posição do módulo de potência do robô

Foi montado o “módulo de controle” do robô acima da bateria. A *RaspberryPi* e a placa de circuito impresso com os circuitos de acoplamento foram colados acima de EVA para que ficassem em um nível mais alto que os componentes de potência. Já os três sensores ultrassônicos foram montados no para-choque com Durepoxi. O esquema da figura 25 descreve a disposição desses componentes.

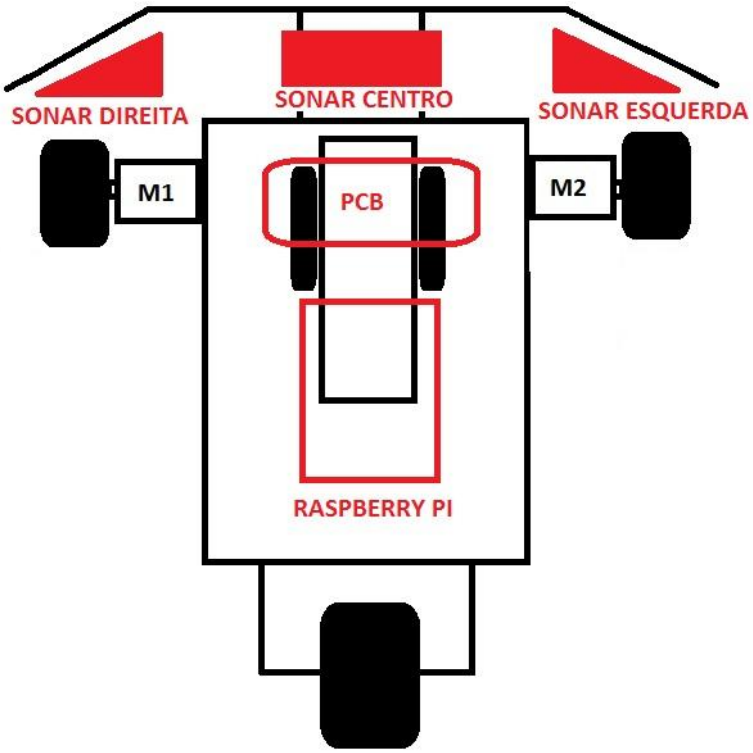

Figura 25 - Diagrama da posição do módulo de controle do robô

Como um último recurso de segurança, foram presos dois parafusos na lateral da plataforma de madeira, e a eles foi amarrada uma linha de pesca de nylon de 1mm de diâmetro. Dessa forma, foi possível garantir que mesmo em teste de maiores velocidades o robô permanecesse dentro do espaço previsto para o teste.

5. Resultados

As imagens a seguir são fotos das vistas superior, frontal e lateral do robô.

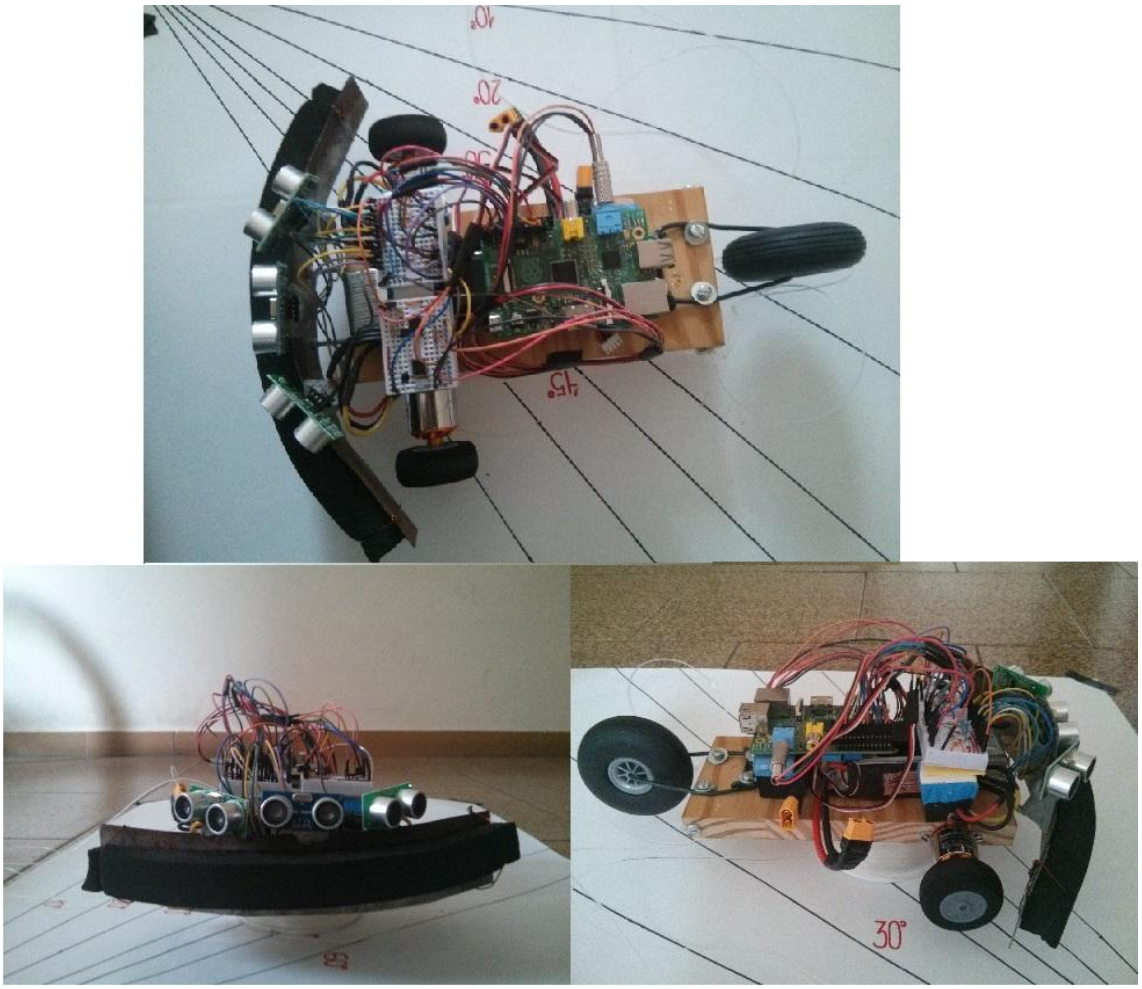

Figura 26 - Vistas superior, fontral e lateral do robô

5.1. Teste de velocidade "*headless*"

Para verificar a premissa que poderia ser feito um robô autônomo que alcançasse altas velocidades, foi feito um teste dos motores em uma estrutura sem uma unidade de processamento. A estrutura foi desenvolvida como a mostrada acima, uma placa de madeira com motores *brushless* nas laterais. Neste caso, a estrutura só suportava as baterias e os dois controladores eletrônicos de velocidade. A velocidade atingida nesse teste foi 86km/h.

Depois o robô foi completado com a *RaspberryPi*, os sonares e os módulos de acoplamento conforme mostrado acima. Os testes do sistema de navegação foram

feitos com a velocidade mínima dos motores, que foi calculada em 5,4m/s, aproximadamente 19,5 km/h. Para velocidades menores do que esta, os motores perdem muito torque e não são capazes de movimentar o veículo no solo.

5.2. Teste da interferência da velocidade nos sonares

Uma preocupação que surgiu durante o desenvolvimento do robô foi a interferência da velocidade dos obstáculos no plano perpendicular à distância sendo medida. Para sanar quaisquer dúvidas, os sonares foram levados a um automóvel, conectados à *RaspberryPi*. Um sonar foi preso no retrovisor esquerdo de um Ford Fiesta e testes em cinco velocidades foram feitos para observar se haveria mudança na leitura da distância do retrovisor ao chão.

Tabela 4 - Resultados dos testes de interferência da velocidade do obstáculo no plano perpendicular à medida

Velocidade	**20 km/h**	**40 km/h**	**60 km/h**	**80 km/h**	**100 km/h**
Altura média:	97,25	96,975	96,525	96,275	98,975
Desvio padrão	0,7625	2,27625	1,29375	4,31375	1,47875

O resultado desse teste, apresentado na tabela acima, mostra que, apesar de erros, não houve uma interferência significativa da velocidade na resposta.

5.3. Teste do sistema de navegação

Figura 27 - Teste do sistema de navegação

Os testes do sistema de navegação do robô foram feitos em um corredor estreito, como mostrado na figura 27. Por isso, limitou-se o alcance dos sensores de distância por software, para que detectassem obstáculos somente mais próximos que 80cm. O módulo de maior prioridade, o de parada emergencial, irá operar com o desligamento dos motores caso um obstáculo esteja muito próximo da frente do robô. Caso contrário, o módulo de desvio irá operar, evitando que o robô colida. A seguir, temos um gráfico mostrando a leitura dos sensores e o sinal de velocidade enviado para os ESC. O sinal varia de 0, motor parado, a 5, velocidade de testes.

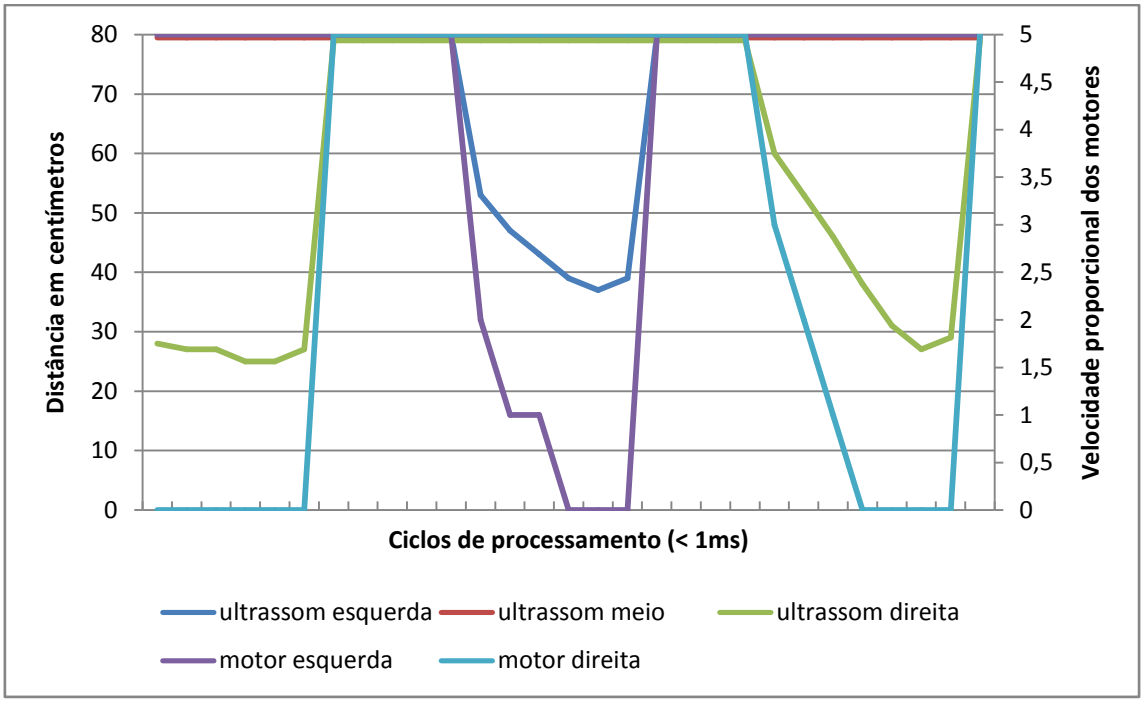

Figura 28 - Resposta dos motores proporcional a distância de obstáculos

É possível observar que o controle dos motores é proporcional a obstáculos laterais. Caso uma parede à direita esteja ficando mais próxima, o sistema irá diminuir a velocidade do motor da esquerda, fazendo com que assim o robô curve para a esquerda. Se o robô conseguir se manter exatamente no meio, o último módulo entrará em ação de enviará para os motores o comando de seguir em frente.

A figura 29 mostra o teste do módulo de maior prioridade, o módulo de parada emergencial. Nesse teste o robô foi colocado a uma distância de 2 metros da parede e a velocidade máxima dos motores, correspondente ao número 5 no gráfico, equivalia a 18km/h. A figura 30 mostra a resposta dos motores e as leituras dos sonares. A resposta nesse caso não é proporcional, como a do módulo de desvio apresentado na figura 28. Esse módulo visa a segurança do próprio robô, visto que uma batida de frente pode comprometer o módulo de controle. No teste o robô encostou na parede, porém já desacelerado e com os motores desligados.

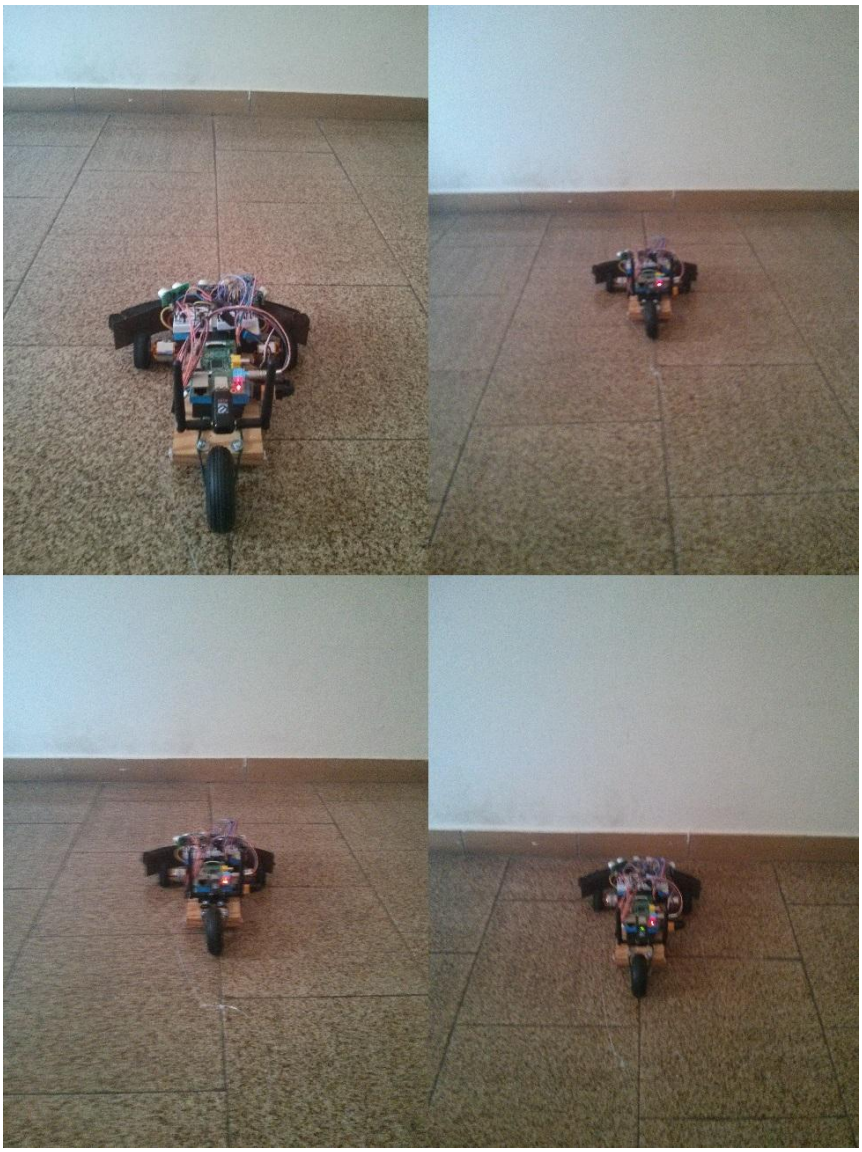

Figura 29 - Teste de parada emergencial

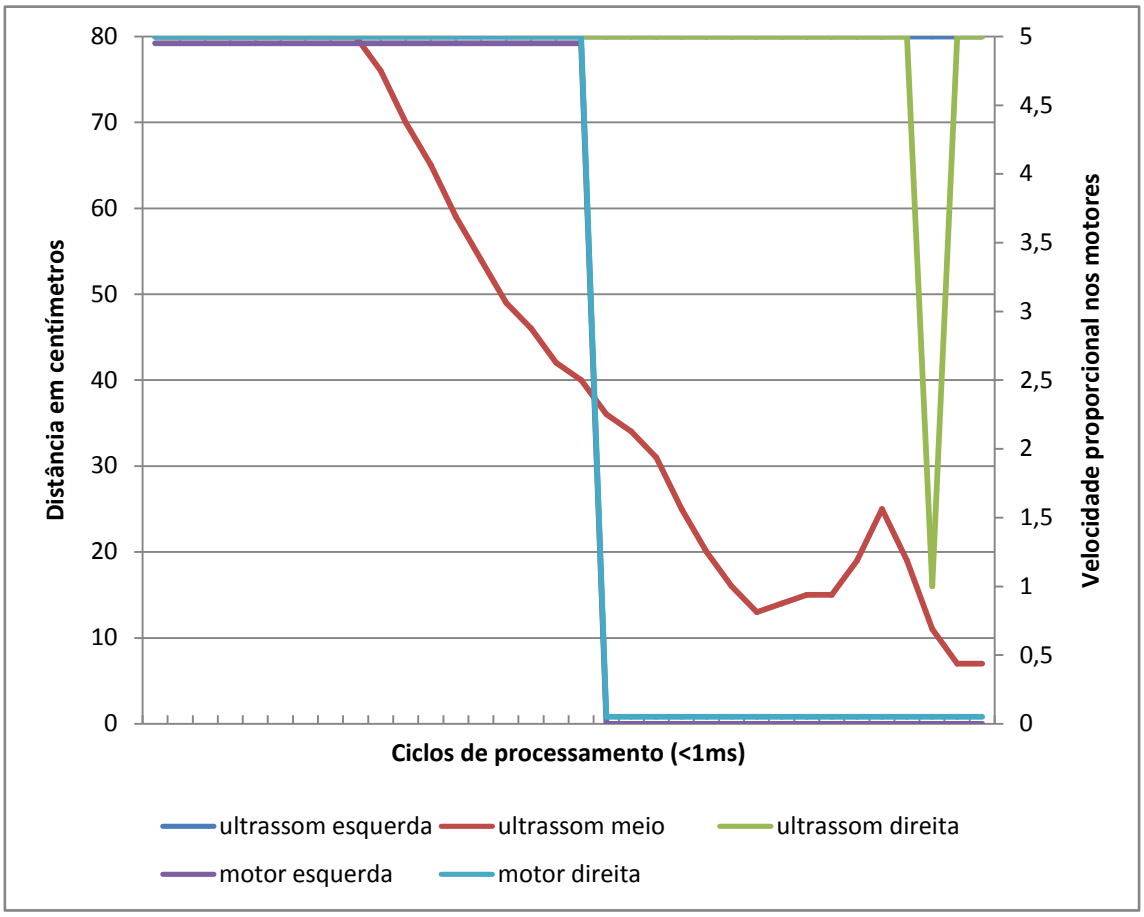

Figura 30 - Resposta dos motores ao módulo de parada emergencial

6. Conclusões

O principal fundamento da proposta do trabalho realizado era a viabilidade de um robô autônomo em alta velocidade que utilizasse motores brushless DC e uma RaspberryPi como unidade de processamento. Os motores se mostraram capazes de atingir uma velocidade satisfatória e seu acionamento por meio dos controladores eletrônicos de velocidade não apresentou dificuldades uma vez que se conseguiu gerar o sinal de controle por hardware. A RaspberryPi foi capaz de lidar com o sistema desenvolvido sem apresentar problemas de memória ou velocidade de processamento.

A RaspberryPi, no entanto, não se mostrou robusta, especialmente em relação ao uso dos pinos de entrada e saída digital. Durante alguns testes não foi possível o controle dos sensores de distância, pois um pino estava flutuando, apesar de ser configurado por software. Esse problema era resolvido reiniciando-se a placa. É uma consideração para um trabalho futuro que a RaspberryPi se comunique com outra placa microcontrolada, uma placa mais robusta e responsável somente por lidar com as entradas e saídas do sistema, como um Arduino Uno, por exemplo.

Os motores mostraram velocidade satisfatória, porém, apresentaram problemas na partida e na manutenção de velocidades mais baixas (inferiores a 20Km/h). Para partir o robô nos testes de menor velocidade, é preciso segura-lo na mão, sem que as rodas estejam tocando o chão, ligar o sistema até que os motores acelerem e então colocar o robô no chão empurrando-o para frente, para acelerá-lo manualmente. Portanto, para futuros testes, seria indicado a compra de motores com a mesma relação rpm/Volt, porém com maior torque ou com caixa de redução. Para o robô proposto isso não é problema, pois seu objetivo será testar o desenvolvimento de software e hardware para aviões, portanto não irá se movimentar com menos de 40 Km/h.

Os sensores de distância por ultrassom foram testados em diferentes condições e se mostraram confiáveis. No entanto, durante os testes do sistema de navegação, foi necessário aumentar o ângulo entre os três sonares. Perdeu-se precisão nas medidas da frente, pois um obstáculo pode estar em um espaço intermediário entre o sensor do meio e o da lateral, de forma a se encontrar a mais de 15° de qualquer um dos dois. Porém, com essa modificação, se ganha na percepção lateral. Nos testes, o robô se comportou melhor na atividade de desviar obstáculos laterais depois da modificação do ângulo dos sensores. Para um melhor sensoriamento de obstáculos pequenos, é aconselhável a instalação de mais dois sonares a 60 graus cada.

O módulo de comunicação wireless não foi satisfatório. Ele funcionou corretamente e permitiu a telemetria remota, porém ao custo de muitas perdas de conexão. Era constante a necessidade de reiniciar a sessão SSH de controle do terminal da RaspberryPi, pois a conexão wireless havia sido perdida. Uma sugestão para trabalhos futuros é melhorar esta parte do software, incluindo rotinas capazes de reiniciar automaticamente a conexão em caso de perda.

O sistema de navegação se comportou dentro do esperado, e ajustes do controle proporcional dos motores no módulo de desvio de obstáculos tiveram que ser feitos durante os testes. Foram mudados os limites de supressão de um módulo para o outro dentro da arquitetura de subsunção de forma a otimizar o comportamento para o teste sendo realizado em cada ambiente de trabalho.

Considera-se que este trabalho satisfez seu objetivo, uma vez que foi construído, testado e validado um robô móvel autônomo capaz de se movimentar em velocidades de até 86 Km/h, o que é suficiente para testar o desenvolvimento de novas estratégias de controle de navegação para os VANTs produzidos pelo Laboratório de Computação Reconfigurável do ICMC-USP. Uma continuação natural deste trabalho pode ser a conexão de uma unidade inercial, compasso e GPS na Raspberry PI, para que esta possa estimar a posição do robô e a direção de seu movimento, possibilitando a navegação por rotas pré-definidas. Também se pode instalar câmeras de vídeo na Raspberry PI, para que softwares de reconhecimento de imagem possam reconhecer landmarks ou objetos de interesse, possibilitando o cumprimento de missões mais complexas. Também é sugerido que sensores como o Kinect da Microsoft sejam avaliados no robô.

O desenvolvimento deste projeto envolveu uma abordagem abrangente dos conteúdos estudados no curso de engenharia elétrica, em especial os abordados nas disciplinas específicas, Aplicações de Microprocessadores, Circuitos Eletrônicos e Máquinas Elétricas. Logo, o trabalho desenvolvido como um todo se mostrou de grande valor para a formação, pois permitiu o desenvolvimento de um sistema móvel autônomo completo, o que requer lidar com problemas muito variados, seja ruído, mau posicionamento do motor, programação do controlador, e até a montagem do chassi.

7. Referências bibliográficas

2213N 800Kv Brushless Motor – Hobby King.Disponível em http://hobbyking.com/hobbyking/store/__8622__2213n_800kv_brushless_motor.html

40A Eletronic Speed Controller – Hobby King.Disponível em http://www.hobbyking.com/hobbyking/store/__24562__HobbyKing_40A_ESC_4A_UBEC.html

4N25 | VishaySemiconductors, documento número 83725. Abril 2004. Disponível em www.vishay.com/docs/83725/4n25.pdf

Ararinha – GISA ICMC –USP, Agosto 2013. Disponível em http://gisa.icmc.usp.br/site/wp-content/uploads/2013/03/Ararinha-Folder-v1.1.pdf

BCM2835 ARM Peripherals | Broadcom Corporation, 2012.Disponível em www.farnell.com/datasheets/1521578.pdf

BROOKS, R. A Robust Layered Control System for a Mobile Robot, Setembro 1985.Disponívelem AI Memo 864.

CHIARETTA, S. How to control a brushless motor through a ESC with Arduino | Drones and ROVs, Novembro, 2012. Disponível em https://dronesandrovs.wordpress.com/2012/11/24/how-to-control-a-brushless-motor-esc-with-arduino/

Circular de Informações Aéreas AIC N 21/10. Disponível em http://servicos.decea.gov.br/arquivos/publicacoes/bf624198-2f5c-4dd6-93569e5d5fcb4f4c.pdf?CFID=9c1dd2b1-ab24-4d71-ad8b-fb9bfd13ce49&CFTOKEN=0

DCA. 63-4 DIRETRIZ PARA IMPLEMENTAÇÃO DOS COMITÊS REGIONAIS RESPONSÁVEIS PELOS ASSUNTOS RELACIONADOS AOS SISTEMAS DE AERONAVES REMOTAMENTE PILOTADAS (RPAS). Disponível em http://servicos.decea.gov.br/arquivos/publicacoes/29a2fe60-8406-40df-bc093a4219116f70.pdf?CFID=9c1dd2b1-ab24-4d71-ad8b-fb9bfd13ce49&CFTOKEN=0

Elecfreaks | Ultrasonic Ranging Module HC-SR04 Datasheet.Disponível em http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf

ESC 40 A Manual | Hobby King.Disponível em https://www.hobbyking.com/hobbyking/store/uploads/HK_ESC_Manual%281%29.doc

Fairchild | Optocoupler Solutions, 2015.Disponívelemhttps://www.fairchildsemi.com/collateral/Optocoupler-Solutions.pdf

GERKEY, VAUGHAN e HOWARD, 2003. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. Disponível em: ProceedingsoftheInternationalConferenceonAdvancedRobotics (ICAR 2003) pages 317-323, Coimbra, Portugal, June 30 - July 3, 2003.

GRANA, A. Sistema de controle de navegação embarcado para um robô movél autônomo baseado no computador de baixo custo Raspberry PI. Dezembro 2013.

HENDERSON, G.About | WiringPi. Disponível em http://wiringpi.com/

HENDERSON, G. Software PWM Library | WiringPi.Setembro, 2012. Disponível em https://projects.drogon.net/raspberry-pi/wiringpi/software-pwm-library/

HOFFMANN, F. Evolutionary Algorithms for Fuzzy Control.Disponívelem Proceedings of the IEEE, v. 89, n. 9, Setembro 2001

Kar, R. PWM on Raspberry Pi | Raspberry Pi Blog.Disponível em http://www.rpiblog.com/2012/11/pwm-on-raspberry-pi.html

Moorhead, J. Raspberry Pi device will 'reboot computing in schools' - The Guardian. Disponível em http://www.theguardian.com/education/2012/jan/09/raspberry-pi-computer-revolutionise-computing-schools

Nolan, D. Sensorless six-step BLDC commutation | ST Microelectronics, Application Note 4220, Janeiro 2013.

PBL.Raspberry Pi Schematics, Abril 2012.Disponível em http://www.raspberrypi.org/archives/1090

Picademy | RaspberryPi, 2014. Disponível em https://www.raspberrypi.org/picademy/

Raspberry Pi | About Us, 2012.Disponívelemhttps://www.raspberrypi.org/about/

Raspbian | Embedded Linux Wiki, 2014. Disponível em http://elinux.org/Raspbian

Robobees – Harvard, 2015. Disponivel em <http://robobees.seas.harvard.edu/>

RPi Buying Guide – Embedded Linux Wiki, Março 2012. Disponível em http://elinux.org/RPi_Buying_Guide

RPi Distributions – Embedded Linux Wiki, Março 2015. Disponível em http://elinux.org/RPi_Hub

RPi Verified Peripherals | Embedded Linux Wiki, 2012. Disponível em http://elinux.org/RPi_VerifiedPeripherals

SCHEVE, T. “How the MQ-9 Reaper Works” Julho, 2012.Disponível em http://www.howstuffworks.com/reaper1.htm

Servo Library | ArduinoReference. Disponível em http://www.arduino.cc/en/Reference/Servo

Sobre Nós | Yadaa, 2014. Disponível em http://www.yadaa.com.br/#!faq/cl4b

SolarEagle – Boeing, Setembro 2014. Disponível em http://boeing.mediaroom.com/index.php?s=20295&item=1425

TOWNSEND, K. Adafruit | Learn, Agosto 2012. Disponível em https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/

Warner, E. S., Graham, R. W., Read, R. E., 1996. Small format AERIAL PHOTOGRAPHY.Whittles Publishing.Malta.

Wester-Ebbinghaus, W., 1980. Aerial Photography by radio controlled model helicopter. London. England. The PhotogrammetricRecord. Vol. X No. 55.

WOLF, D.; SIMÕES, E. Robótica Móvel Inteligente: Da Simulação às Aplicações no Mundo Real, 2009. Disponível em: XXVIII Jornadas de Atualização em Informática.

Xmobots – www.xmobots.com Disponível em http://www.xmobots.com/Versions/BR/Imprensa/Noticias/20131115_XMobots_Nauru_CAVE_ANAC.html?pagina=noticias.php&idmateria=60

Yedamale, P. Brushless DC (BLDC) Motor Fundamentals | Microship Technology Inc, Application Note 885, 2003.

Apêndice A

Nesse apêndice está apresentado o código usado nos testes de validação dos sonares.

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <string.h>


#define TRIG 17
#define ECHO 18

#define MOTOR0 40
#define MOTOR1 45

void setup() {
wiringPiSetupGpio();
pinMode(TRIG, OUTPUT);
pinMode(ECHO, INPUT);

    //TRIG pin must start LOW
digitalWrite(TRIG, LOW);
delay(30);

                    //seta motor
                    pwmSetMode(PWM_MODE_MS);
                    pwmSetRange(1024);
                    pwmSetClock(375);
                    pwmWrite(MOTOR0,975);
                    pwmWrite(MOTOR1,975);
}

intgetCM() {
                    //Send trig pulse
digitalWrite(TRIG, HIGH);
delayMicroseconds(20);
digitalWrite(TRIG, LOW);

    //Wait for echo start
while(digitalRead(ECHO) == LOW);

    //Wait for echo end
longstartTime = micros();
while(digitalRead(ECHO) == HIGH);
longtravelTime = micros() - startTime;

    //Get distance in cm
int distance = travelTime / 58;

return distance;
}

int main(void) {

                    intdistancia=100;
                    int count=0;

                    charstr[20];
                    printf("Nome do arquivo: ");
```

```
                fgets(str,20,stdin);
                if(str[strlen(str)-1]=='\n'){str[strlen(str)-1]='\0';}
                strcat(str,".txt");
                puts(str);

                FILE *fp;
                fp = fopen(str,"w");

                setup();

                pwmWrite(MOTOR0,965);
                pwmWrite(MOTOR1,964);

                while(count < 101){
                        delay(100);
                        distancia=getCM();
                        printf("Distance: %dcm\n", distancia);
                        fprintf(fp,"%d\n",distancia);
                        count++;
                }
                fclose(fp);
return 0;
}
```

Apêndice B

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <string.h>

//FILE *fp;

#define TRIG0 17
#define ECHO0 18

#define TRIG1 27
#define ECHO1 22

#define TRIG2 23
#define ECHO2 24

#define MOTOR0 40
#define MOTOR1 45

void setup() {
wiringPiSetupGpio();
pinMode(TRIG0, OUTPUT);
pinMode(ECHO0, INPUT);
                        pinMode(TRIG1, OUTPUT);
pinMode(ECHO1, INPUT);
                        pinMode(TRIG2, OUTPUT);
pinMode(ECHO2, INPUT);

      //TRIG pin must start LOW
digitalWrite(TRIG0, LOW);
                        digitalWrite(TRIG1, LOW);
                        digitalWrite(TRIG2, LOW);
delay(30);

                        //seta motor
                        pwmSetMode(PWM_MODE_MS);
                        pwmSetRange(1024);
                        pwmSetClock(375);
                        pwmWrite(MOTOR0,975);
                        pwmWrite(MOTOR1,975);
}

intgetCM(int TRIG, int ECHO) {
                        int count = 0;

                        //Send trig pulse
digitalWrite(TRIG, HIGH);
delayMicroseconds(20);
digitalWrite(TRIG, LOW);

      //Wait for echo start
                        printf("ta baixo    ");
while(digitalRead(ECHO) == LOW);
                        longstartTime = micros();
                        longtravelTime;
                        printf("ta alto   ");

            loop: while(digitalRead(ECHO) == HIGH){
                        travelTime = micros() - startTime;
                        if (travelTime>= 4640 ) gotosai;
```

```
	}
		count=0;
		printf("passou   ");
		while(count<30){
			count++;
			if (digitalRead(ECHO) == HIGH) goto loop;
		}
	sai:;

		//Get distance in cm
int distance = travelTime / 58;

printf("%d",distance);
		//printf("\n");
		//fprintf(fp,"%d\n",distance);
		return (distance);
}

int main(void) {
		int d0,d1,d2,m0,m1;
		setup();
		delayMicroseconds(2000000);

		pwmWrite(MOTOR0,965);
		pwmWrite(MOTOR1,965);

		//char str[20];
		//printf("Nome do arquivo: ");
		//fgets(str,20,stdin);
		//if(str[strlen(str)-1]=='\n'){str[strlen(str)-1]='\0';}
		//strcat(str,".txt");
		//puts(str);

		//fp = fopen(str,"w");

		while(1){

			printf("\nSENSOR0: ");
			d0 = getCM(TRIG0, ECHO0);
			printf("\nSENSOR1: ");
			d1 = getCM(TRIG1, ECHO1);
			printf("\nSENSOR2: ");
			d2 = getCM(TRIG2, ECHO2);
			m0 = 965 ;m1 = 965;
			//if(d0 < 60) m0 = 975;
			if (d0 > 30){
				m0 = 970 - ((d0 - 30)/10);
			}else{
				m0=970;
			}
			//if(d2 < 60) m1 = 975;
			if (d2 > 30){
				m1 = 970 - ((d2 - 30)/10);
			}else{
				m1=970;
			}
			if(d1 < 5) break;
			else if(d1 < 40) {m0=m1=975;};
			pwmWrite(MOTOR0,m0);
			pwmWrite(MOTOR1,m1);

			delayMicroseconds(250000);
		}
```

```
                pwmWrite(MOTOR0,975);
                pwmWrite(MOTOR1,975);
return 0;
}
```