UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Departamento de Sistemas de Computação

Controle de Robô Móvel Embarcado

Daniel Elias Machado Junho

São Carlos - SP

Controle de Robô Móvel Embarcado

Daniel Elias Machado Junho

Orientador: Eduardo do Valle Simões

Monografia referente ao projeto de conclusão de curso de Engenharia de Computação dentro do escopo da disciplina SSC0670 - Projeto de Formatura I do Departamento de Sistemas de Computação do Instituto de Ciências Matemáticas e de Computação – ICMC-USP.

Área de Concentração: Robótica, Sistemas Embarcados, Eletroeletrônica.

USP – São Carlos

10 de Junho de 2013

Melhor ser pirata, do que marinheiro...

Steve Jobs

Dedicatória

Dedico esse trabalho aos meus pais, Gabriel e Eulália, que sempre acreditaram em meus sonhos e me forneceram sustento, inspirações e ferramentas para realiza-los.

Aos meus irmãos, Samantha e Lucas, que com paciência me ajudaram em diversas maneiras para a construção deste trabalho.

Dedico também a minha tia Vanda, a quem sempre recorri nas férias para renovar os ares e me aventurar nas belezas e histórias de Minas Gerais.

Agradecimentos

Agradeço, antes de tudo, a Deus todo poderoso, alfa e ômega de todas as coisas.

Agradeço a minha família pelo apoio e disposição em me ajudar a conquistar meus sonhos e explorar meus potenciais durante todos esses anos.

Ao professor Eduardo do Valle Simões do ICMC pelo entusiasmo e inspiração que exaltou com esse projeto, e ao Prof. Dr. Evandro Luís Linhares Rodrigues da EESC pelos ensinamentos passados, tanto profissionais como cotidianos.

Também ao aluno André Grana do curso de Engenharia Elétrica pelo auxilio no desenvolvimento dos esquemáticos deste projeto e a empresa HI Tecnologia e seus funcionários pelo auxilio e motivação para realização desse trabalho.

E também aos meus amigos pelo auxilio e incentivo neste trabalho.

Resumo

Com a popularização da robótica e baixo custo das plataformas de sistemas embarcados tem aumentado muito o número de usuários que se aventuram nessa área. A grande maioria dos sistemas embarcados com processadores são baseados na arquitetura ARM. Dentre essas plataformas existentes a Beaglebone e a Raspberry Pi. Este trabalho apresenta o uso dessas plataformas de forma a evidenciar as diferença dessas na construção e aplicação de um robô móvel. O sistema completo utiliza as plataformas para controle do robô pelo usuário através de uma interface web onde o usuário poderá controlar a locomoção, observar a câmera acoplada ao robô e a proximidade de obstáculos.

Sumário

Lista de Abreviaturas

AJAX - JavaScript Assíncrono com XML - Asynchronous JavaScript and XML

CSS - Folhas de Estilo em Cascata - Cascading Style Sheets

GPL - Licença Pública GNU - GNU Public License

HDMI- Interface Multimídia de Alta-Definição - High-Definition Multimedia Interface

HTML - Linguagem de Marcação Hiper-Texto - HyperText Markup Language

I/O - Entrada e Saída - In and Out

JPEG- Junta de Grupos de Especialidas em Fotografia - Joint Photographic Experts Group

M-JPEG- JPEG em Movimento - Motion JPEG.

RAM - Memória de Acesso Aleatório - Random Access Memory

SoC - Sistema em um Chip - System on a Chip

USB- Barramento Serial Universal - Universal Serial Bus

UVC- Classe de Vídeo USB - USB Video Class

SO - Sistema operacional

GND – Ground, referente ao sinal terra de circuitos elétricos.

GPIO – Portas programáveis de entrada e saída de sinais. Oferecem uma interface entre processador e periféricos.

YUV - Luminância, intensidade de vermelho, intensidade de azul.

LED – Diodo emissor de luz.

GPU – Unidade de processamento gráfico.

ARM – Arquitetura de processador.

SPI – Protocolo de barramento serial

I2C - Protocolo de barramento serial

PWM – Modulação por largura de pulso

IP - Protocolo de comunicação de rede.

IP – *Intelectual property*.

DC – Corrente contínua.

Lista de Figuras

CAPÍTULO 1: INTRODUÇÃO

1.1. Contextualização e Motivação

O surgimento dos computadores possibilitou ao homem grande liberdade das tarefas cansativas, tais como simples cálculos repetitivos. Hoje, não apenas simples cálculos, mas muitas tarefas foram adaptadas para utilizar a tecnologia e, assim, facilitar e automatizar a vida dos trabalhadores.

O primeiro computador, considerado por muitos como sendo o ENIAC [1] ocupava o espaço de um andar e pesava 30 toneladas, consumindo uma potência 160kW. Na época seu processamento era de até 1000 vezes mais rápido que qualquer maquina de cálculos da época. Hoje seu processamento pode ser equiparado a uma calculadora atual. A Figura 1 mostra o ENIAC sendo operado pelo seu co-inventor John Mauchly.

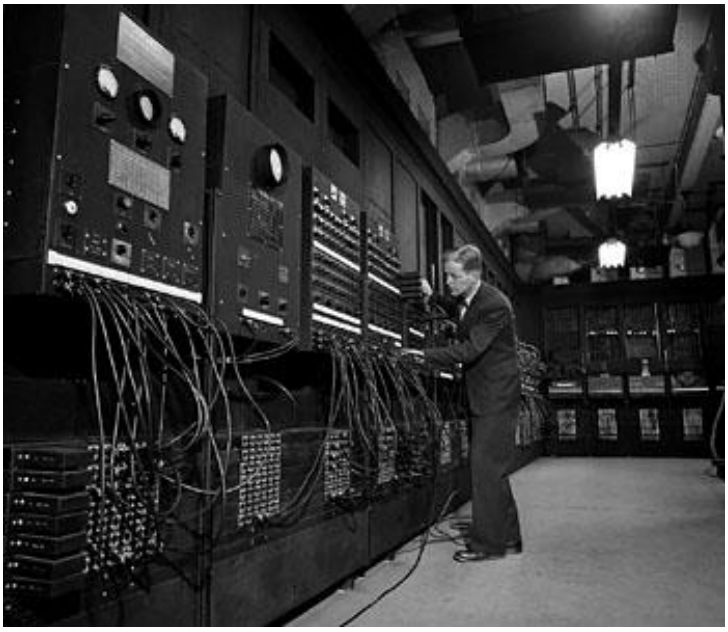

Figura 1 - ENIAC com seu co-inventor John Mauchly [2]

Uma revolução importante para os sistemas computacionais foi a descoberta do transistor por John Bardeen, William Shockley e Walter Brattain no *Bell Laboratories* em 1947 [3]. Após essa descoberta, o desenvolvimento da fabricação desses dispositivos em

chips[1] de silício, e o desenvolvimento de arquiteturas de processadores, os computadores começaram a ocupar áreas menores, consumir menos energia e a possuir um maior poder de processamento. Isso gerou uma revolução na computação, passando a ser mais acessível a grande parte da população entusiasta da computação. Dessa forma um único *chip* podia conter elementos básicos da computação, tais como memórias, periféricos, elementos de processamento, entre outros elementos. Estes *chips* em questão são chamados microprocessadores.

O primeiro microprocessador foi o Intel 4004 [4], construído em 1971, que fora muito utilizado em calculadoras, realizando até 92 mil operações por segundo [5]. A Figura 2 o apresenta.

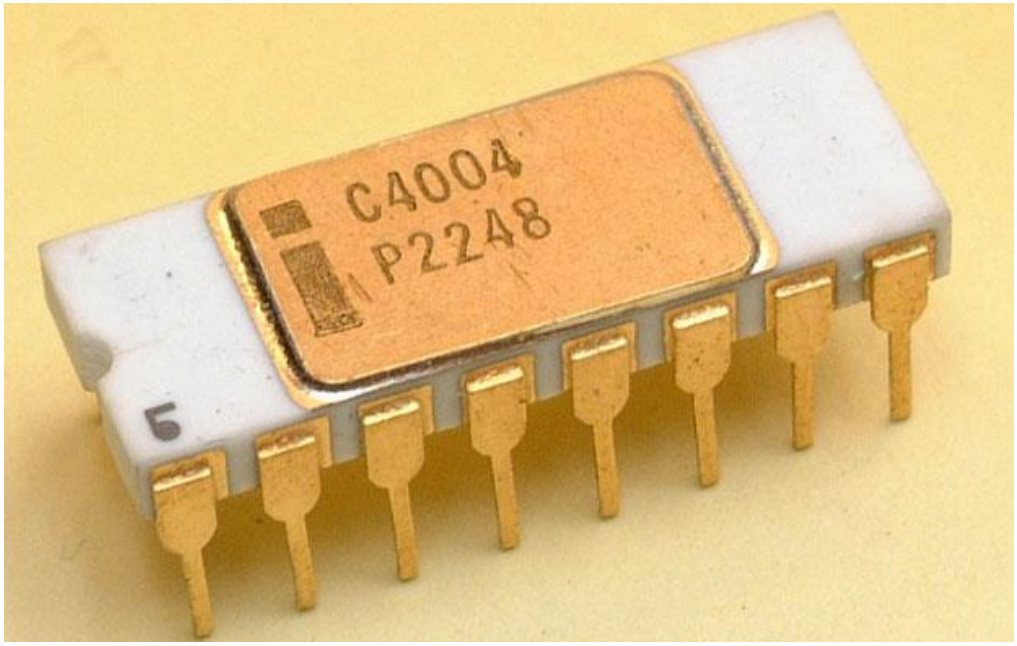

Figura 2 - Intel 4004 [6]

1.1.1 Robótica

Assim como o relógio, o homem tenta criar e recriar ferramentas que lhe auxiliassem no cotidiano. Dessa necessidade de ferramentas começa a surgir a robótica.

[1] Do inglês c*hips* significam “lascas”. O termo é usado como sinônimo para pastilhas de silício que contenham transistores gravados em sua estrutura.

Atualmente existem robôs[2] que são utilizados para acessar lugares perigosos ou de difícil acesso para humanos. Também é possível observar que são empregados em diversos ambientes industriais. Um exemplo notável da utilização de robôs em ambientes hostis é o caso do robô *Curiosity*, enviado a Marte para exploração [7]. A Figura 3 mostra uma representação 3D da *Curiosity* em Marte.

Figura 3 - Representação 3D da Curiosity em Marte [7].

Todo esse desenvolvimento da robótica foi possível graças aos *chips*. Com a possibilidade de construção de sistemas que usam circuitos integrados de baixo consumo e alto processamento. Isso possibilitou uma maior autonomia a dispositivos controlados através de uma fonte de alimentação móvel tal como uma bateria.

Sistemas robóticos funcionam basicamente através de 3 processos [34]: Percepção, Processamento e Atuação.

[2] *Robots,* vem da palavra tcheca *Robota* que significa escravo. Palavra originária do livro "Opilek" e "R.U.R. Rossum's Universal Robots" de Karel Capek. Nos livros um homem fabricava máquinas com aparencia humana para que servissem de escravos, e as denominava *robota*.

A aquisição consiste em sentir o ambiente, obtendo informações através de sensores. O processamento, na utilização dessas informações para análise e tomada de decisão. A atuação, posterior ao processamento, consiste na tomada de ações do robô junto ao ambiente.

1.1.2 Sistemas Embarcados

Sistemas embarcados consistem em sistemas microprocessados no qual todo o poder de processamento é voltado à realização de uma atividade mais específica ou um sistema que ele controla. O sistema é construído de forma a realizar sua atividade mais rapidamente, ocupando um menor espaço, consumindo menos energia e reduzindo-se gastos, se comparado com um computador normal [39]. Existem dispositivos com sistemas embarcados com diversos propósitos, alguns com um propósito mais dedicado e outros com um propósito mais geral.

Sistemas embarcados de propósito dedicado são aplicações em que seu uso é mais restrito. Geladeiras, máquinas de lavar, sistemas de freios ABS, copiadoras, entre outros, são sistemas que fazem parte de nosso dia-a-dia. Esses sistemas são dimensionados em seus componentes computacionais de acordo com suas tarefas, por exemplo, possuindo um processador de menor velocidade, mas que consiga responder em tempo hábil para desempenho da aplicação. A Figura 4 demonstra o funcionamento do sistema ABS, em que é realizado um processamento em tempo real para atuação em automóveis, mantendo a estabilidade em situações que podem ocasionar a perda do controle do veículo.

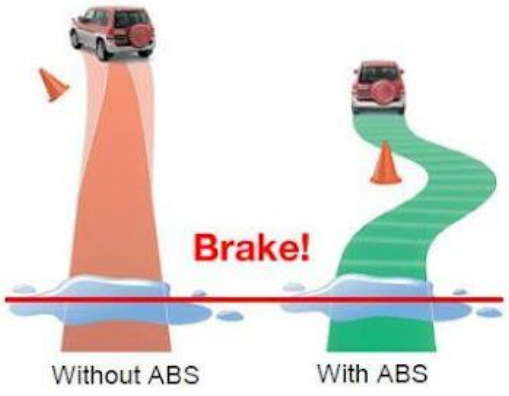

Figura 4 - Situação de uso do sistema de freios ABS [8]

Sistemas embarcados com propósito mais geral possuem geralmente processadores e memórias mais potentes e de maior desempenho se comparados aos dedicados. Pois seu uso pode várias de acordo com sua finalidade e podem ser inclusive multitarefa. Um exemplo comum desses sistemas são os telefones celulares *smartphones* e os *tablets.* Existem vários modelos e cada qual é criado visando um mercado consumidor específico. Atualmente, existem *smartphones* [9] que possuem processadores equivalentes a computadores [10] com processadores lançados de 8 à 4 anos atrás. A Figura 5 mostra um smartphone da marca Samsung, modelo *Galaxy* SIII [11], cujo processador Exynos 4 é um Dual ARM Cortex-A9 [12] de última geração.

Figura 5 - Celular Samsung Galaxy SIII [13]

Processadores ARM tem se tornado referência em sistemas embarcados [40]. O uso de processadores de 32 bits, dependendo da aplicação, pode ser mais vantajoso se comparado a alguns microcontroladores, tais como os da família PIC. A aquisição de processadores ARM que trabalham em frequências mais elevadas não é simples, pois são vendidos para grandes corporações em quantidades muito altas, mas aqueles que operam em frequências menores (até 80 MHz) custam poucos dólares e são vendidos por unidade, como qualquer outro componente eletrônico. Seu preço, dependendo do modelo, pode se equiparar a seus concorrentes como pode ser pesquisado em uma loja (tal como a loja online Digi-Key [14]). O fato de serem mais acessíveis possibilita soluções simples oferecendo um desempenho maior para o usuário final.

A ARM [15] é uma empresa, sediada na França, que trabalha com *Intelectual property*[3](IP). Seu foco não é a fabricação e sim o desenvolvimento de projetos de processadores para uso de terceiros. Diversas empresas do mercado da eletrônica, como Atmel, Samsung, Motorola, utilizam os projetos da ARM, desenvolvendo processadores, microcontroladores, e *SoCs*[4] agregando-os em seus produtos.

Esse mercado proporcionou a criação de plataformas de desenvolvimento que utilizam arquitetura ARM. Essas soluções foram realizadas tanto por desenvolvedores independentes como ligados a corporações. Essa tendência alavancou o mercado de sistemas embarcados, permitindo o desenvolvimento de GPS, modens, celulares, televisores, entre outros muitos dispositivos eletrônicos.

A facilidade de desenvolvimento para processadores ARM logo possibilitou a adaptação do Linux, um sistema operacional, e toda sua estrutura às arquiteturas desse processador. Isso possibilitou o surgimento do Android[5], uma plataforma popular de celulares com processadores dessa tecnologia.

1.2. Objetivos

Esse trabalho tem por objetivo realizar um comparativo e mostrar a possibilidade de utilização de plataformas de desenvolvimento ARM, tal como a BeagleBone [17], e a Raspberry Pi [18], como uma plataforma de controle de robôs móveis. Esta plataforma constitui um robô simples com sistemas de locomoção, um sensor para medição de proximidade de obstáculos, e um sistema de câmera de vídeo. O sistema é totalmente

[3] Referente a propriedade intelectual. No ramo da eletrônica, significa que o projeto é uma concepção, tal como uma arquitetura, e não um produto físico.

[4] Sigla de *System on a Chip*. Corresponde a um circuito integrado que incorpora vários componentes de computadores em um único *chip*. Muito utilizado em sistemas embarcados.

[5] Sistema operacional baseado no kernel do Linux para dispositivos móveis, desenvolvido pela Open Handset Alliance, e liderado pela Google.

controlado por um usuário através de uma interface web, não tendo, o robô, nenhuma inteligência ou controle sobre suas ações.

Pretende-se obter com esse trabalho um comparativo entre as plataformas aqui trabalhadas, para escolha de desenvolvimento de um projeto robótico e uma comparação do desenvolvimento dos componentes do sistema deste projeto para as plataformas ARM utilizadas. A comparação entre as duas plataformas é em termos de desempenho para o usuário, com a percepção de tempo e resposta entre ações, e em termos de desenvolvimento para o projetista, facilidade de geração de código.

Todo o desenvolvimento dos códigos é fornecido nos anexos deste trabalho.

1.3. Organização do Trabalho

Este trabalho apresentará a uma introdução sobre as plataformas utilizadas, de forma a mostrar as diferenças principais entre elas para o desenvolvimento do robô. As seções seguintes abordaram sobre a forma como foi realizado o desenvolvimento do robô para ambas as plataformas. E por fim, apresentará uma análise sobre as maiores diferenças na implementação deste projeto.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

2.1. PWM – Pulse Width Modulation

PWM é a abreviatura de *Pulse Width Modulation*, em português Modulação de largura de pulso [37]. Neste projeto, esse tipo de sinal é usado para controle da velocidade do motor. Este sinal é utilizado para transmissão de potência para uma carga. Imagine um sinal que hora esteja desligado, hora ligado. Este sinal quando desligado não passa corrente à carga, e quando ligado passa, transmitindo energia à carga. Quando esse movimento do sinal é realizado a uma frequência alta e 50% do tempo o sinal esteja ligado transmitirá à carga uma potência de 50% do total que pode ser oferecido pelo sinal. Portanto a potência média vista pela carga é a própria tensão média aplicada.

Agora, os tempos em que o sinal está fechado ou aberto podem variar. Esses tempos somados fornecem o período desse sinal. A razão entre o tempo do pulso, tempo em que o sinal está ativo, e o período do sinal fornece o ciclo ativo do sinal, chamado de *duty cycle*.

Largura do pulso é o tempo em que o sinal é ativo, portanto um sinal que tenha largura 50%, em seu período fica 50% do tempo ativo, um sinal que tenha largura 5% fica 5% do período ativo, e assim por diante. A Figura 6 mostra sinais PWM.

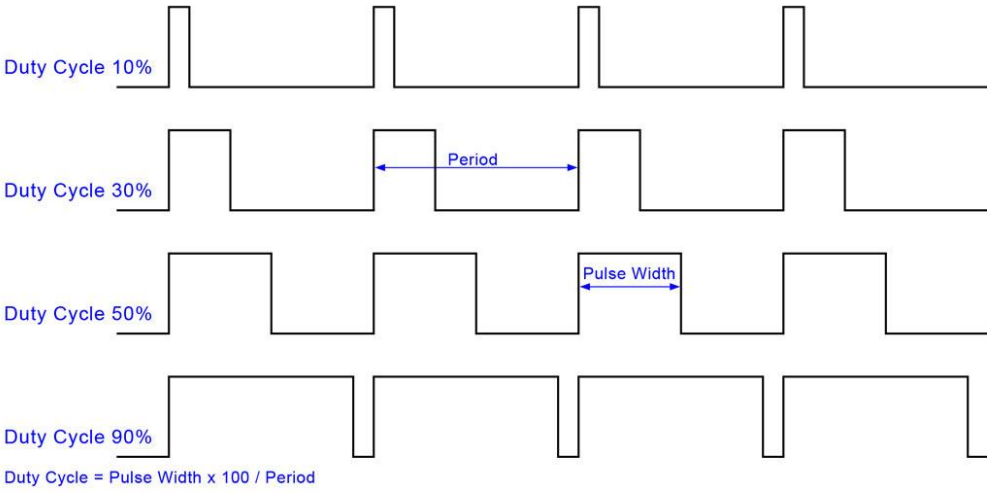

Figura 6 - 4 sinais PWM. Fonte [37]

2.2. Ponte H

Ponte H é um circuito elétrico que tem por objetivo permitir a passagem de corrente, hora em um sentido hora em outro [38]. Amplamente utilizado em controle de motores de corrente continua (DC) em diversas aplicações, neste projeto também é utilizado. Seu circuito consiste numa estrutura de 4 transistores em 2 colunas, 2 transistores em cada coluna, de tal forma que a carga esteja na horizontal entre os 4. Os transistores são saturados dois a dois, sempre na diagonal, de tal forma que a corrente na carga irá circular em um sentido ou noutro. Assim, quando a ponte está ligada, 2 transistores estão saturados enquanto os outros 2 estão em corte. Quando a ponte está desligada os 4 transistores estão em corte.

A construção da ponte H pode ser feita com a utilização de transistores, ou com o uso de circuitos integrados que já fornecem uma ponte H. A Figura 7 apresenta um esquema simplificado de uma ponte H.

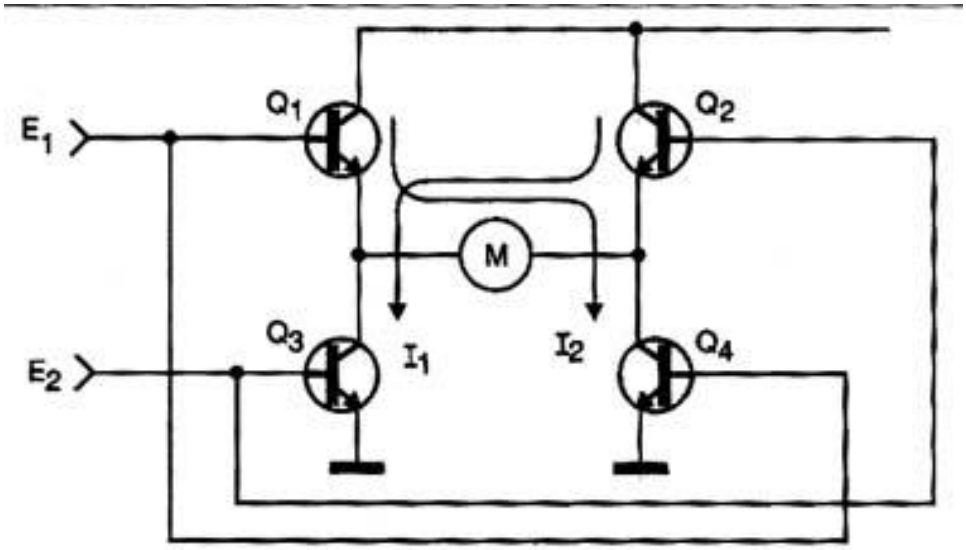

Figura 7 - Ponte H. Fonte [38]

CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO

3.1. Considerações Iniciais

Os sistemas utilizados nesse projeto correspondem ao sistema operacional GNU/Linux, rodando sobre uma plataforma ARM de 32 bits [15]. A facilidade de utilização do Linux permite maior entendimento por parte do desenvolvedor sobre o funcionamento e integração do sistema desenvolvido com sistema operacional, pois este é de código aberto (do inglês *opensource*) que permite total acesso e modificação de sua estrutura. Os contratos vinculados a esse sistema e seus *softwares* são em sua grande maioria GPL [16]. A utilização desse S.O. vem crescendo tanto por usuários, como por desenvolvedores, e atrelado a isso a documentação e material de auxílio para aprendizagem.

Para a Raspberry Pi utilizou-se o S.O. Occidentalis [34] e na Beaglebone o S.O. Angstrom [35]. Ambos as fontes desses sistemas ensinam passo a passo a instalação para utilização do mesmo. Para tal é necessário apenas a plataforma e um cartão SD[6], no caso da Raspberry Pi, e um microSD, no caso da Beaglebone.

3.2. O Projeto

Com o objetivo de realizar um comparativo entre a Beaglebone e a Raspberry Pi como uma plataforma robótica, determinou-se um conjunto robótico básico para aquisição de dados para uma comparação. O robô em suma tem 5 módulos básicos para controle e atuação, são eles: locomoção, sensor de distância, visualização, processamento e interface com o usuário realizada através da comunicação via rede TCP/IP.

O módulo de deslocamento é constituído de 2 motores conectados a esteiras de forma semelhante a um tanque de guerra [41]. Os motores são controlados através de um

[6] Cartão de memória flash.

PWM, para cada motor, gerados pelas placas de processamento. A Figura 8 apresenta a estrutura deste módulo. Para o sensor de distância, utilizou-se um sonar, desenvolvido para uso do Arduino[7], modelo HC-SCR04 [24]. A Figura 20 apresenta esse sensor. Como visualização do robô, uma webcam genérica USB foi conectada para geração e transmissão das imagens. Para o processamento utilizou-se as plataformas ARM, Raspberry Pi modelo B de 512MB ou o Beaglebone revisão A6. A interface como usuário foi desenvolvida através de uma página Web, que envia informações para o robô.

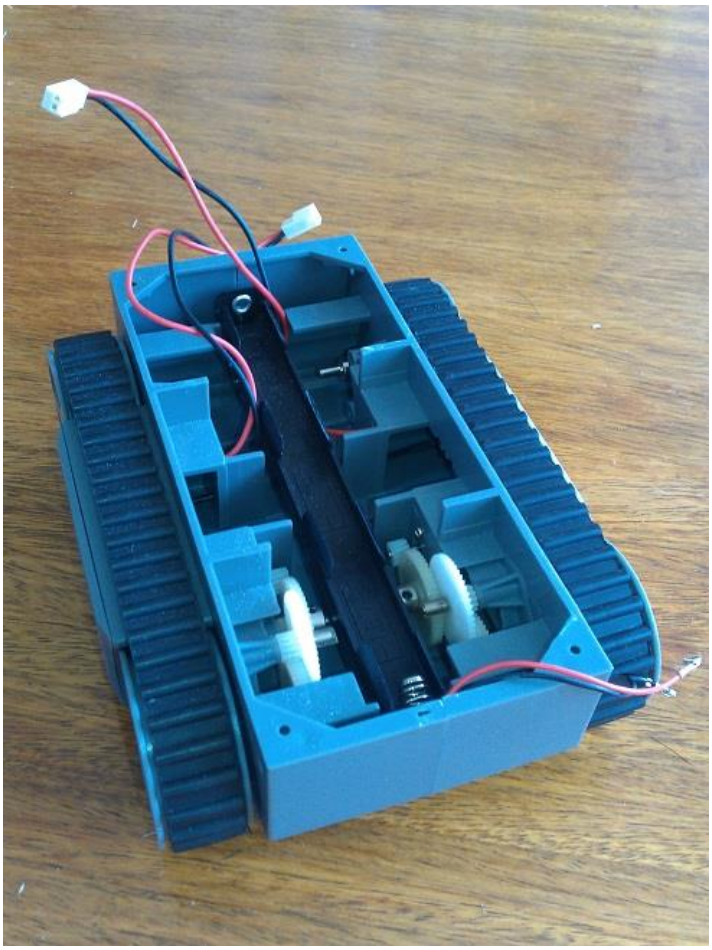

Figura 8 - Base robótica de locomoção [41]

[7] Ferramenta de prototipagem eletrônica aberta.

3.2.1. Plataformas ARM

Nessa seção serão apresentadas as plataformas Beaglobone e Raspberry Pi utilizadas. Ambas são exibidas na Figura 9.

Figura 9 - Beaglebone e Raspberry Pi utilizadas nesse projetos.

3.2.1.1 Raspberry Pi

O projeto da Raspberry Pi, lado direito na Figura 9, tem por objetivo disponibilizar computadores simples e baratos ao mercado consumidor especialmente jovens e crianças para que tenham acesso a ferramentas básicas para aprendizado de programação. Sua concepção surgiu nos laboratórios da Universidade de Cambridge no Reino Unido [42].

O principal componente é SoC da Broadcom [28], que agrega o processador ARM 11 de 700 MHz ARM1176JZF-S, uma GPU VideoCore IV e 512MB memória RAM. Além deste, há também um saída de vídeo HDMI e RCA, saída de áudio P2, interface de rede ethernet[8], 2 portas USB, e um conector micro USB para alimentação de 5V e 700mA.

[8] Arquitetura de interconexão de redes locais baseada no envio de pacotes.

Existem 26 pinos disponíveis, dentre eles 1 PWM. A tabela abaixo apresenta as informações técnicas da Raspberry Pi.

Preço	U$35
SoC	SoC da Broadcom BCM2835
CPU	700 MHz ARM1176JZF-S core
GPU	Broadcom VideoCore IV
Memória	512 MB SDRAM
Porta USB	2 portas
Saída de vídeo	RCA composto e HDMI
Saída de áudio	Conector de 3,5mm (P10), HDMI
Rede	10/100 Ethernet
Periféricos	GPIO (8), UART, I²C, SPI
Consumo	700mA (3,5W)

3.2.1.2 BeagleBone

A Beaglebone, na esquerda na Figura 9, é um projeto e hardware aberto com design simples. O objetivo do projeto era a construção de uma plataforma de prototipação. Desenvolvido pela Beagleboard.org [17].

Seu principal componente é um ARM Cortex-A8 da Texas Instruments, mas também possui um acelerador gráfico 3D, 256MB de SDRAM, interface USB Host e Ethernet. Outras interfaces podem ser acessadas via 2 conectores de expansão para acesso a 46 pinos, entre essas interfaces estão SPI, HDMI, VGA e conversor A/D. Sua alimentação pode ser via micro USB, também usada para acesso serial, ou uma fonte de 5 V. Existem 46 pinos disponíveis, dentre eles 8 podem ser usados como PWM. A tabela abaixo apresenta as informações técnicas da beaglebone.

Preço	U$89
CPU	AM3359 ARM Cortex-A8 720MHz
Memória	256 MB DDR2

Porta USB	1 portas
Saída de vídeo	Nenhuma
Saída de áudio	Nenhuma
Rede	10/100 Ethernet
Periféricos	SPI, I2C, GPIO (65), LCD, HDMI, VGA, MMC, RS232, CAN e A/D (12 bits)
Consumo	500mA (2,5W)

3.3. Descrição das Atividades Realizadas

3.3.1. Locomoção

Para o sistema de locomoção do robô, foi adquirido uma base robótica, com duas esteiras e um motor associado a cada esteira. Ela permite que com a utilização de um suporte sustentar todos os componentes do projeto [41]. A Figura 8, apresenta essa base.

Para utilização dos motores, optou-se pela construção de uma interface entre esses módulos. Essa escolha foi tomada devido ao grande ruído gerado por motores em geral e pela possibilidade de geração de corrente indesejada pelos motores. Corrente que poderia resultar na queima de uma das portas de interface do processador das placas, visto que a ligação dessas interfaces aos conectores de acesso aos periféricos é feita diretamente, sem um circuito de proteção, como é apresentado pelos esquemáticos das placas [21] e [22]. Para geração da tensão lógica para controle do motor, utilizou-se um regulador de tensão LM7805 [23].

3.3.1.1. Construção do Driver

A construção do driver para controle do motor baseou-se no sistema de controles ponte H e na possibilidade de drenagem de corrente dos pinos do processador. Então foram utilizados componentes disponíveis e de fácil acesso para construção do mesmo, tanto para a utilização da ponte-H, quanto para segurança da corrente utilizada pelos pinos.

A corrente máxima de saída que pode passar pelos pinos é identificada nos manuais e esquemas associados às plataformas ARM. Para a Raspberry Pi essa corrente é 16mA por pino, mas com o máximo 51mA somando-se todos os pinos e para a Beaglebone é de 4 a 6 mA por pino.

Para assegurar sobre as correntes passadas pelos pinos optou-se por isolar os sinais provenientes dos pinos, sobre os sinais provenientes do motor. Assim, no esquemático pode ser facilmente observada a isolação dessas partes, e interfaceando essas partes a utilização dos optoacopladores. Estes dispositivos, explicando de forma bem simplificada, funcionam de forma que em seu interior um LED controla a corrente que passa através de um transistor do outro lado. Isso permite que se possam isolar de forma simples os sinais do motor dos sinais que advém do processador.

A Figura 10 apresenta, na vertical, o esquemático do *driver* desenvolvido e construído nesse projeto, esse esquemático será apresentado em maiores detalhes e explicado parte por parte mais a frente.

Existem 6 sinais de entrada para o driver. Sendo eles os sinais dos motores 1 e 2, sentidos dos motores 1 e 2, 5V do processador e o sinal GND do processador. Estes sinais são ligados do *driver* aos suportes dos pinos das plataformas através de fios.

Pode-se descrever o driver como contendo 6 módulos. Sendo eles Amplificador Operacional, Optoacopladores, Inversor, L298 componente de controle do motor através de uma ponte-H, e o motor.

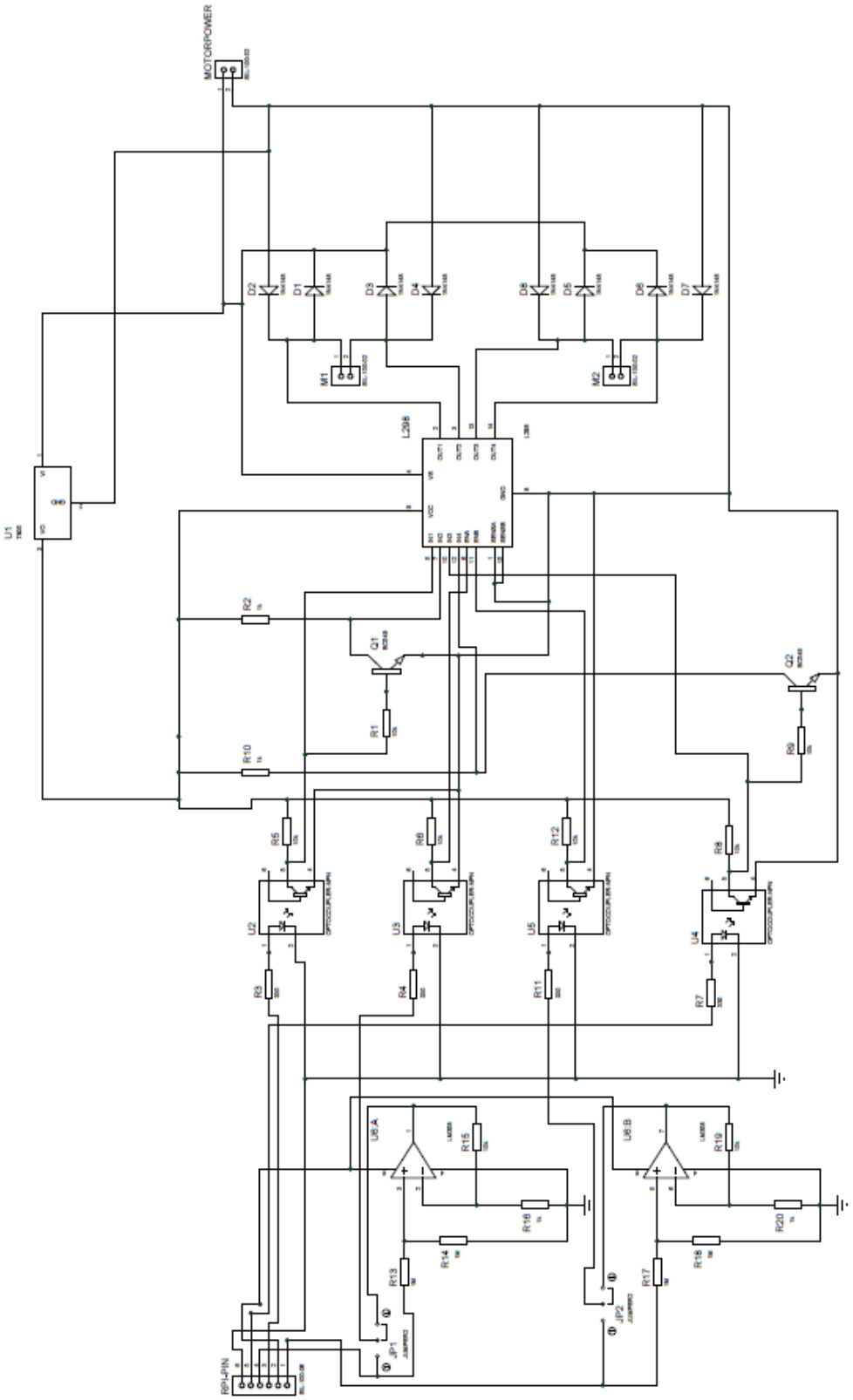

Figura 10 - Esquemático do driver para controle do motor.

Amplificador Operacional

A Raspberry Pi para esse projeto tem uma limitação. Em sua estrutura apenas um PWM é acessível através das GPIO disponíveis. Contudo, podemos observar em seu esquemático a existência de 2 PWM, além do disponível, que são utilizados no áudio. Esses sinais serão utilizados para o controle do Motor. A Figura 11 e a Figura 12 mostram partes do esquemático [22] em que é possível ver o uso desses sinais PWM no áudio. Esses passam por um circuito de forma a gerar o sinal um sinal analógico para uso de sistemas de áudio.

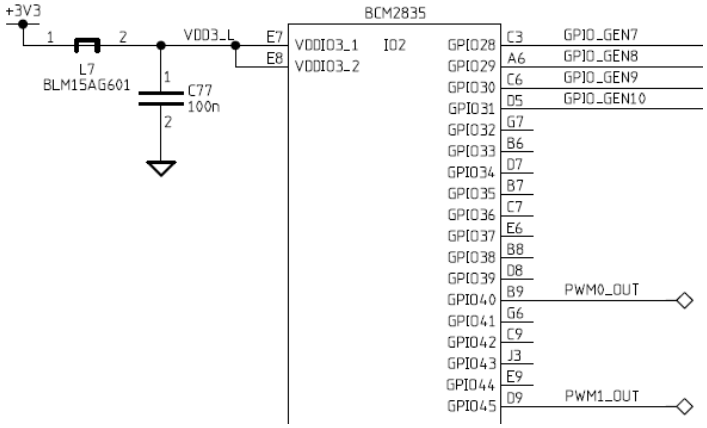

Figura 11 - GPIO da BCM2835 definidas como sinais PWM. Fonte [22]

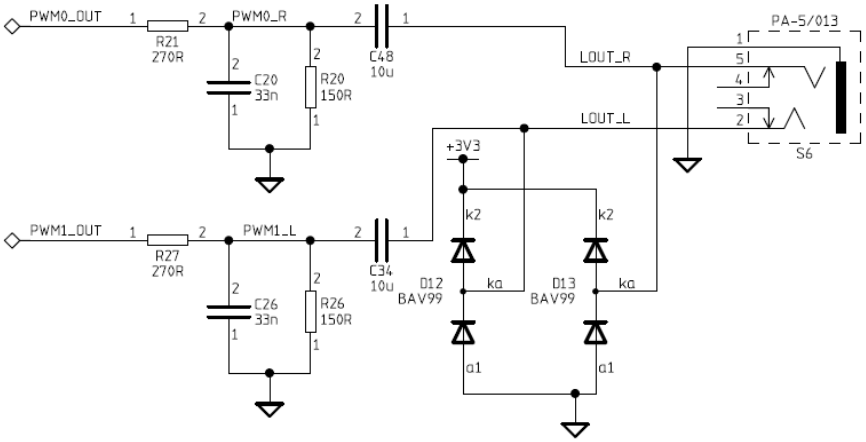

Figura 12 - Circuito em que as PWM da Raspberry Pi são utilizados como áudio. Fonte [22]

Dessa forma, devido à existência de um capacitor de 10µF na saída do circuito de áudio, o uso de um amplificador operacional foi necessário para não sobrecarregar esta saída e fornecer uma isolação adequada à este circuito.

O circuito de entrada do driver é mostrado na Figura 13, nele pode-se observar o valor dos resistores utilizados. O circuito realiza a amplificação de forma a não inverter o sinal, utilizando a configuração não inversora. Pode-se também observar a inserção de um *jumper* de forma a ter a opção de utilização ou não do módulo do amplificador operacional. Isto permite a utilização do driver também com a Beaglebone.

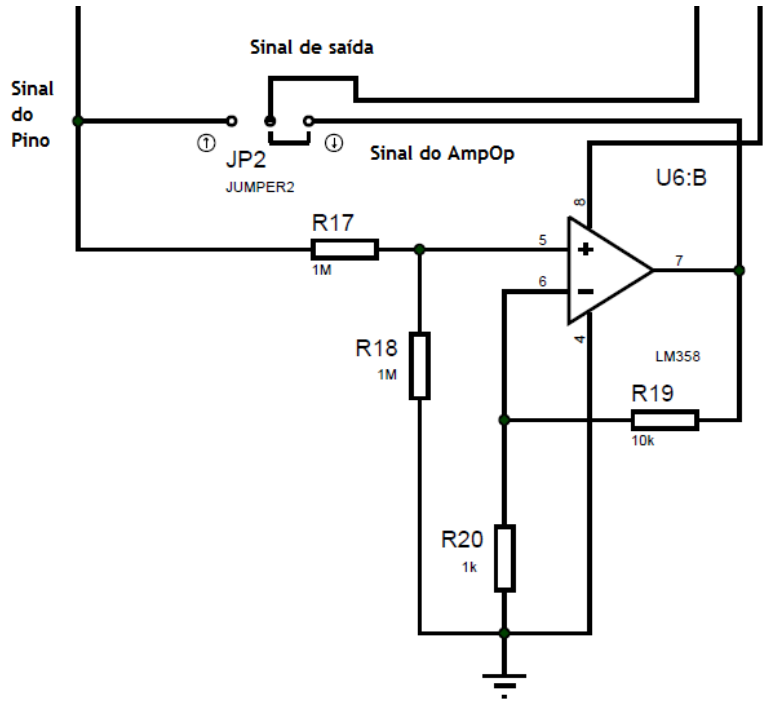

Figura 13 - Módulo de amplificação do sinal do driver.

Optoacopladores

O sistema possui quatro optoacopladores, apresentados na Figura 14, um para cada sinal, liga/desliga e sentido, dos motores. Estes dispositivos utilizados são do tipo NPN. Posteriormente a construção do driver, percebeu-se um erro de projeto que gerou um erro de segurança do sistema. Os sinais do motor, que fornece o sinal PWM, quando desligados,

acabam por ligar o motor, devido a escolha dos optoacopladores NPN. Esse erro pode ser corrigido posteriormente pela adição de um inversor ao sinal. Essa correção será feita numa próxima revisão.

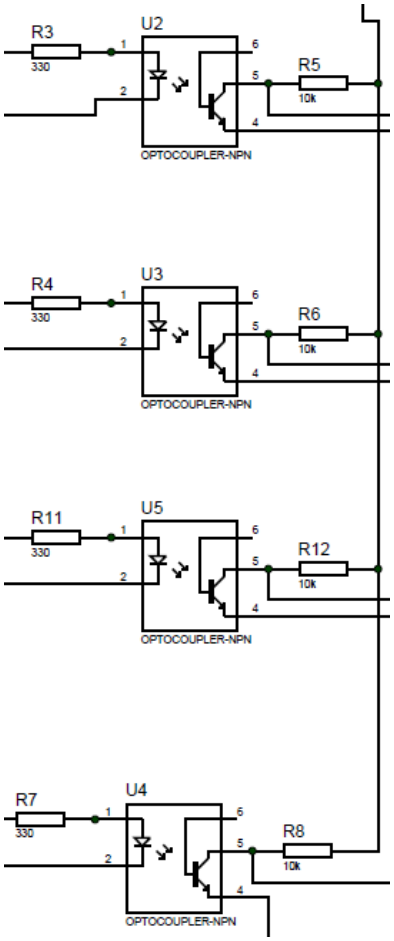

Figura 14 - Esquemático dos optoacopladores.

Inversor

O modulo de inversão, apresentado na Figura 15, foi produzido devido a uma questão de economia de sinais e processamento. O CI[9] L298 [20], apresentado na próxima seção, para controle de sentido do motor utiliza 2 sinais. Quando estes sinais são iguais, o motor para, quando diferentes o motor gira para um lado ou para o outro, dependendo da ordem dos sinais lógicos.

[9] Sigla para Circuito Integrado.

Assim, para economia de sinais, e consequentemente de pinos disponíveis, optou-se por utilizar apenas um sinal lógico para sentido, e inverte-lo para utilizar o par de sinais para o L298.

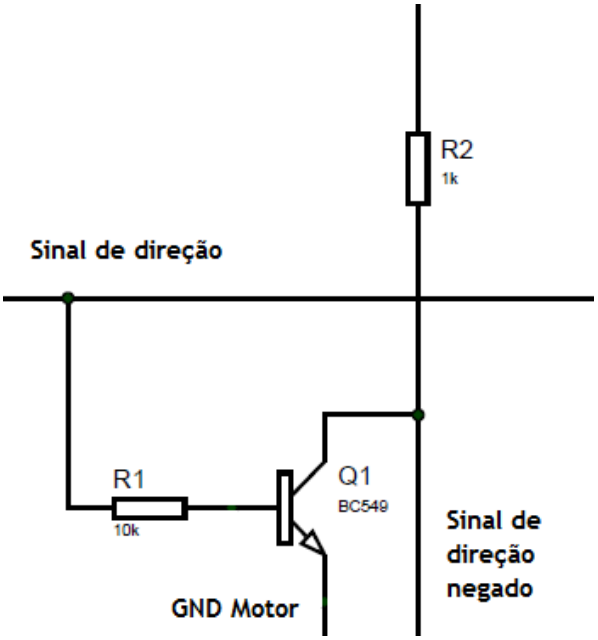

Figura 15 - Inversor

L298

O L298 é o CI responsável pela administração da ponte-H. É representado no esquemático como na Figura 16. Todo seu funcionamento pode verificado em seu *datasheet* [20].

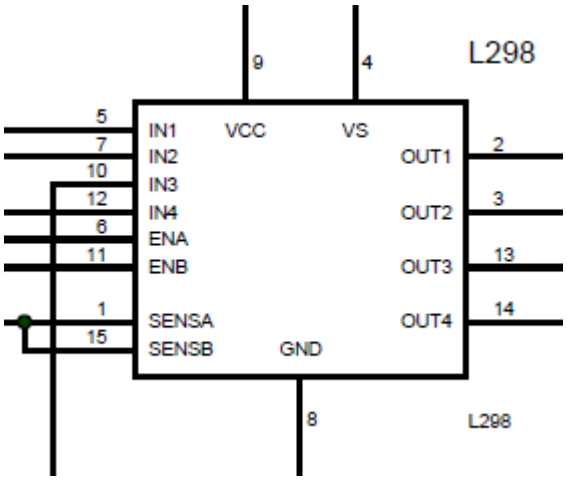

Figura 16 - CI L298

Os sinais IN 1 e 2 correspondem aos sinais de sentido do motor 1, e IN 3 e 4 ao motor 2, o sinal EN A o sinal de liga/desliga do motor 1 e B, liga/desliga do motor 2. SENSA e SENSB são utilizados para verificação da corrente que passa através da ponte H, que neste projeto não são utilizados e assim ambos são ligados ao GND. A tabela lógica a seguir demonstra a lógica dos sinais para o motor 1.

ENA	IN1	IN2	Resultado
0	X	X	Motor parado
1	0	0	Motor parado
1	0	1	Motor gira no sentido horário
1	1	0	Motor gira no sentido anti-horário
1	1	1	Motor parado

Motor

Os motores são conectados nas saídas OUT do L298 e associados a diodos para proteção da força contra eletromotriz induzidas nas bobinas do motor quando este é ligado e desligado. Tensões, estas que podem danificar o circuito de saída do L298. Conforme sugerido pelo fabricante [20], foi feita a disposição dos diodos entre os motores VS e GND.

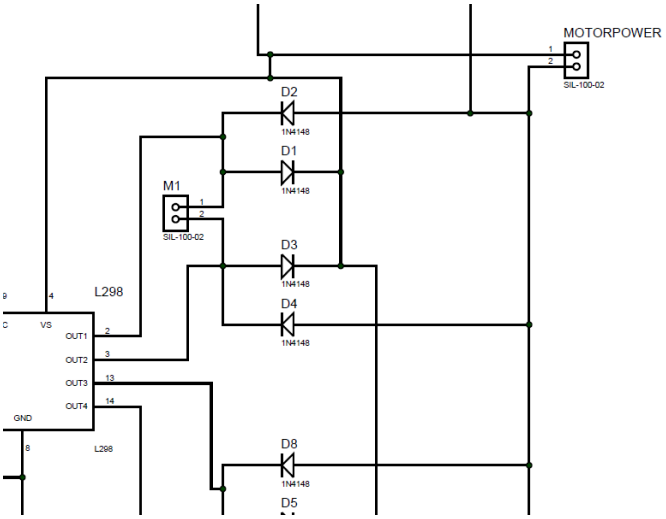

Figura 17 - Esquemático do motor.

Layout

Após feita a lógica de funcionamento do drive, as ligações dos componentes e simulação computacional com utilização do programa Proteus [43] e a montagem do circuito em um protótipo foi desenhado o *layout*[10] da placa. Este é apresentado na Figura 18.

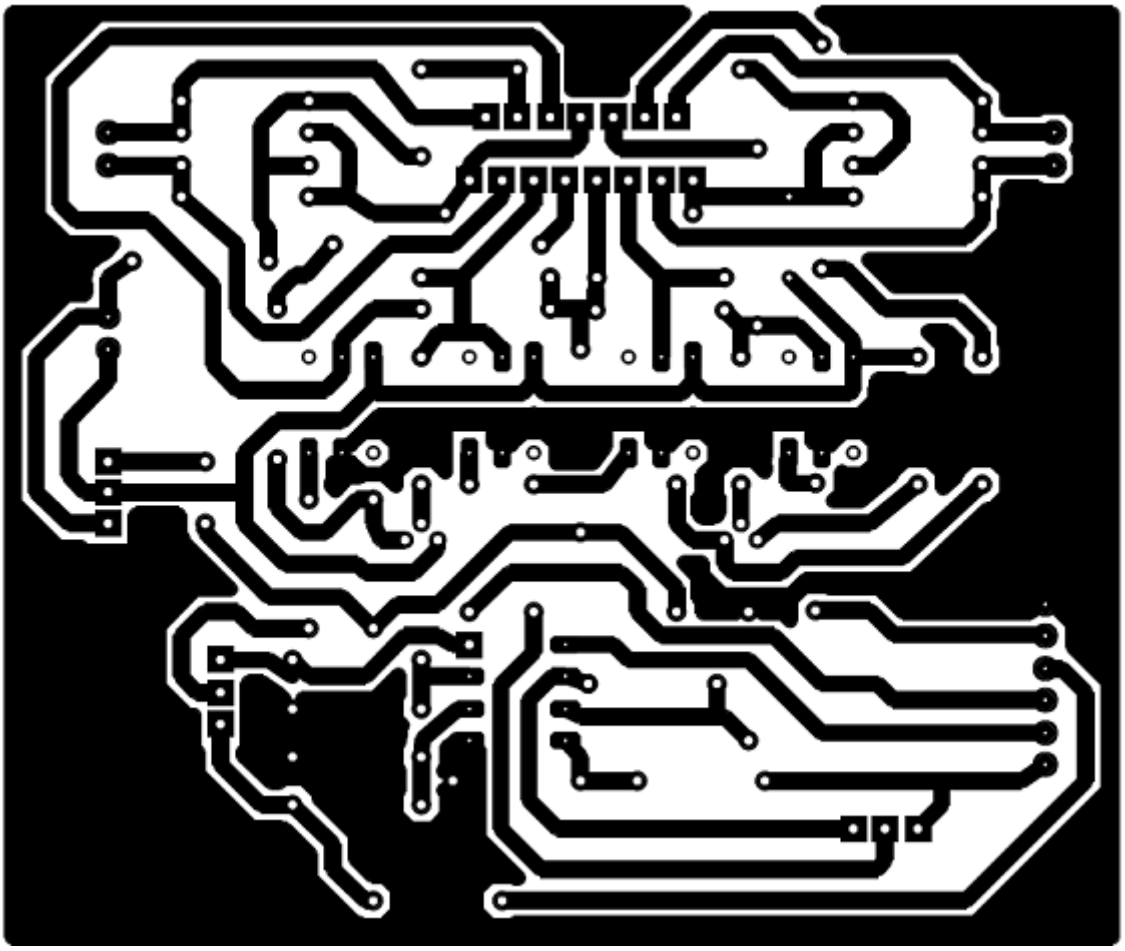

Figura 18 - Desenho do *layout* das trilhas do driver. Lado cobreado.

[10] Associado ao desenho de cobre que ficam nas placa eletrônicas devido a disposição dos componentes. Esses desenhos em cobre servem para a condução da corrente elétrica.

A imagem da placa finalizada, já com a corrosão, encaixe e solda dos componentes é apresentada na Figura 19. Todos os componentes utilizados nesta placa são listados no Apêndice B deste documento.

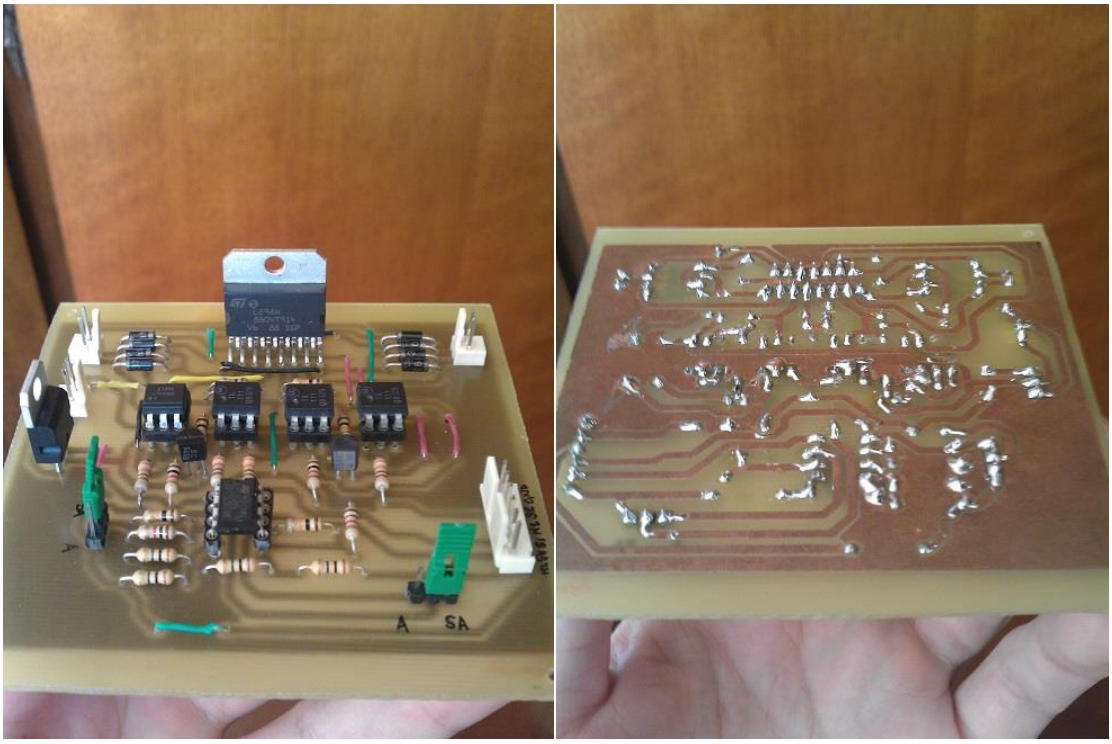

Figura 19 - Driver finalizado.

Optou-se pelo desenvolvimento do driver ao invés da adoção de soluções prontas para colocar em práticas ações para desenvolvimento de um protótipo de circuito eletrônico.

3.3.2 Programação

A programação foi feita em C, evitando-se utilizar linguagem de mais “alto” nível ou interpretadas. Isto aumenta o desempenho do sistema como um todo visto que algumas partes possuem tempo como seu fator crítico. Os programas rodam sobre um sistema operacional e portanto competem pelo uso do processador, funcionando mais lentamente, do que um programa não executado sobre um S.O.

Os controles dos motores e do sonar foi desenvolvido, utilizando-se no caso da Raspberry Pi, uma biblioteca chamada WiringPi [29] para acesso aos GPIO, e no caso da Beaglebone, uma biblioteca chamada BBLib [30], que foi modificada para uso.

A principal diferença entre as duas plataformas, foi o acesso e controles dos pinos. A Beaglebone e a Raspberry permitem um acesso das GPIO via leitura e escrita de arquivos que são monitorados pelo Kernel. Contudo na Raspberry Pi, devido ao único canal de PWM disponível pelo pinos, utilizou uma biblioteca [29] que realiza um mapeamento de memória permitindo o acesso a todos os pinos do SoC Broadcom, incluído os pinos utilizados pelo áudio.

Nos códigos a frequência aplicada ao sinal PWM é 150Hz, este valor foi escolhido de forma empírica, de forma que apresentou uma resposta mais satisfatória do motor de todos os valores testados.

O controle das variáveis do sistema é feito pela leitura e escrita dos arquivos. O servidor web executa programas que leem e escrevem em arquivos, controlando as variáveis velocidade e ângulo. Estes programas são “inc_velocity.c”, “dec_velocity.c”, “inc_angulo.c” e “dec_angulo.c”. Um programa deve ser executado na plataforma robótica para coleta desses dados. Esse programa controla o motor e é definido em “Motor.c” que lê os arquivos “velocity.db” e “angulo.db” e, com esses dados, aplica a ação sobre os sinais do motor. A variável "Velocidade", vinculada a "velocity.db", é utilizada para calcular o duty cycle dos sinais PWM e o sentido de rotação do motor, caso seja negativa. A variável "Ângulo", vinculada a "angulo.db", é usada para manipular os valores de duty cicle de forma a fazer as rodas girarem em velocidades diferentes, fazendo com que o robô faça uma curva. Note que ângulo não está relacionado ao ângulo da curva, mas sim a velocidade angular em que a curva é realizada.

Os programas desenvolvidos nesse projeto são executados nas plataformas ARM e controlados através da comunicação estabelecida via rede TCP/IP. O protótipo desenvolvido utilizou um cabo ethernet para estabelecimento da rede mas, foram realizados alguns testes com um adaptador de rede sem fio e um roteador Wi-Fi [44]. Contudo, a arquitetura de rede do robô não é o foco desse projeto e, portanto, foi adotado o uso do cabo ethernet para a comunicação do robô.

Os programas são controlados pelo usuário através da interface web disponível, em que parâmetros armazenados em arquivos são modificados e utilizados pelos programas. Os códigos desenvolvidos estão alocados no Apêndice A deste projeto.

3.3.3. Distância

A distância é medida através da técnica de sonar ultrassônico. Para tal, utilizou-se um módulo, comumente utilizado em projetos com Arduino, modelo HC-SR04 [24], como o da Figura 20.

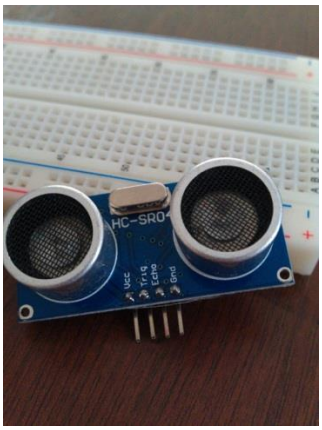

Figura 20 - Módulo do sonar ultrassônico.

Este módulo funciona com tensão lógica de 5 volts. O sinal de entrada de trigger, que gera o pulso de transmissão, foi conectado diretamente ao pino de saída do processador, que embora tendo tensão lógica "1" igual a 3,3V, está dentro do limite aceitável pelo circuito do modelo como nível lógico "1". Como suas saídas lógicas fornecem 5V como lógica "1", não foi possível liga-la diretamente a placa de processamento, já que as suas tensões lógicas são de 3,3V e não possuem circuito de proteção de sobretensão. Assim, para as saídas lógicas foi utilizado um divisor de tensão simples, descrito na Figura 21.

O cálculo da distância é realizado através de uma fórmula, em que se utiliza o tempo de espera entre um pulso dado em *trigger*, e a resposta em *echo*. Dado que a velocidade do som é de $340\ m/s$, temos:

$d = \frac{t}{58}$ Centímetros, sendo t em microsegundos.

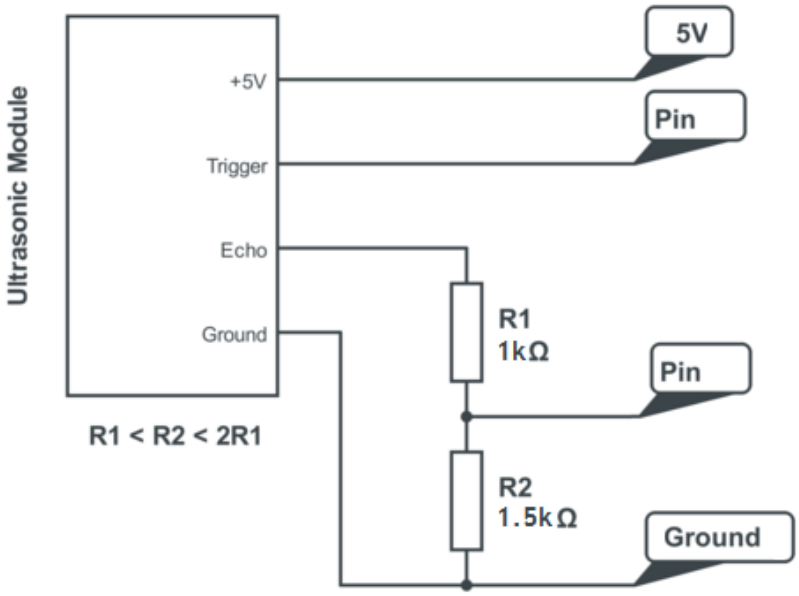

Figura 21 - Divisor de tensão para funcionamento do módulo ultrassônico. Imagem adaptada de [25].

3.3.4. Visualização

Para o sistema de visualização da câmera webcam acoplada ao robô utilizou-se a aplicação de *streaming* [26] chamada MJPG-Streamer [27]. Essa aplicação funciona com câmeras USB compatíveis com o UVC[11]. Essa aplicação já fornece suporte a HTML, podendo exportar os vídeos anexando-os em um código HTML simples. Seu manuseio e configurações são simples.

A instalação do streamer de vídeo é ao todo muito simples. Para sua utilização é necessário ter uma webcam conectada que seja reconhecida pelo linux. Para tanto, faremos aquisição do código fonte e a compilaremos dentro da própria plataforma para facilitar.

Para tanto é necessário instalar o subversion, um software de controle de versão. Utilizaremos este para baixar a versão mais corrente do mjpg-streamer.

[11] Abreviação do inglês *USB Video Controller*, Controlador de Vídeo USB, em português. Mais referências sobre esse projeto podem ser encontradas em http://www.ideasonboard.org/uvc/.

```
opkg install subversion (Beaglebone)
sudo apt-get install subversion (Raspberry Pi)
```

Após a instalação, o utilizaremos para baixar o código fonte. Pode-se baixa-lo em qualquer pasta.

O comando para download do código:

```
svn co https://mjpg-streamer.svn.sourceforge.net/svnroot/mjpg-streamer mjpg-streamer
```

Após faça a compilação do mesmo.

```
make all
```

Depois de feita a compilação, pode-se utiliza-lo executando a seguinte linha de comando:

```
./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -r 160x120 -f 10" -o "./output_http.so -p 8080 -w /home/root/mjpg-streamer/mjpg-streamer/www -n"
```

Os argumentos para configuração da aplicação são os seguintes:

Parâmetros da Webcam:

-d	Dispositivo de video a ser aberto
-r	Resolução da imagem
-f	frames[12] por segundo

[12] Do inglês, quadro ou moldura. Se refere as imagem que é adquirida pela câmera.

-y	habilita formato YUYV e desabilita modo MJPEG
-q	Compressão de qualidade JPEG em porcentagem (ativa formato YUYV e desabilita MJPEG)
-m	Não adquire frames menores que esse limite
-n	Não inicializa dynctrls do Linux-UVC driver
-l	Muda o LED liga/desliga, o deixa piscando ou deixa para ser setado pelo driver

Parâmetros do HTML:

-w	Caminho da pasta que contem os arquivos da pagina web da aplicação MJPG-Streamer
-p	Porta TCP para o servidor HTTP
-c	Pergunta por “usuário:senha” ao abrir conexão
-n	Desabilita execução de comandos

Neste caso, a aplicação executa o servidor de streaming sob a porta 8080, gerando uma imagem de 160 por 120 *pixels* a 10 frames por segundo. Com o MJPG-Streamer em execução na plataforma ARM, acesse um navegador web do computador conectado a plataforma e acesse o endereço http://[IP da plataforma ARM]:8080, a Figura 22 deverá ser apresentada.

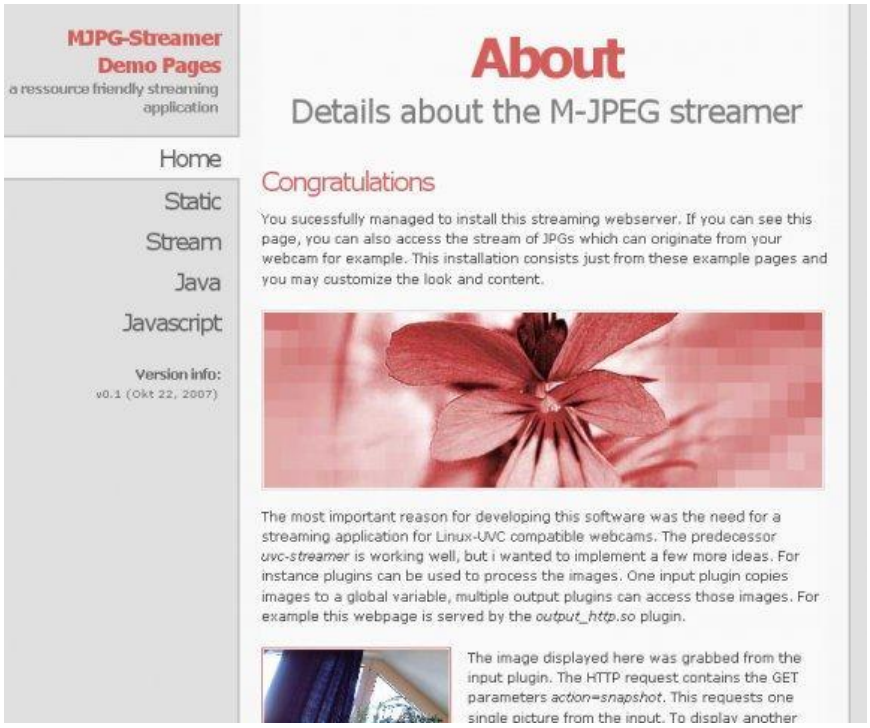

Figura 22 - Interface web do MJPG-Streamer

3.3.5. Interface com o usuário

Para realizar uma interface do usuário com a plataforma de controle do robô, foi desenvolvido uma página web, em que o usuário tinha controle sobre a movimentação do mesmo. Para isso utilizou-se de HTML, PHP e CSS para criar uma interface simples. A Figura 23 mostra a interface desenvolvida onde ao centro temos a exibição da imagem da web, a direita a cima a distância obtida do sonar, a direita ao centro 4 botões direcionais para controle de locomoção do robô e a direita a baixo um botão para parar a movimentação do robô.

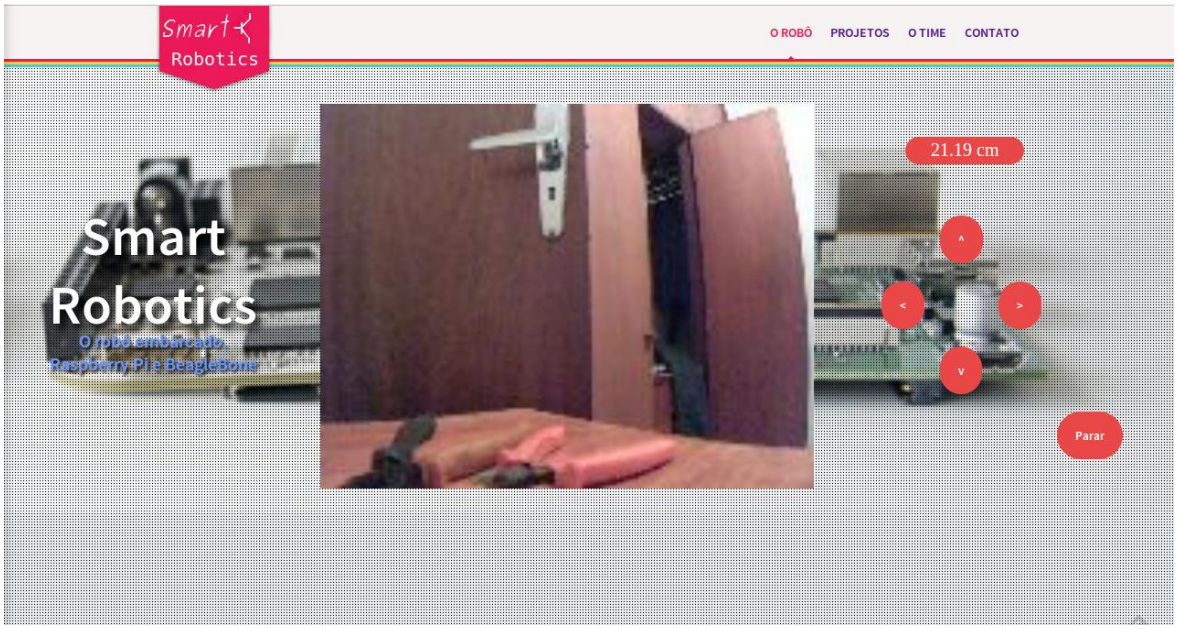

Figura 23 - Página desenvolvida para controle do robô.

O servidor web utilizado foi o Lighttpd [31], um servidor leve e de fácil configuração que consome em média 5 MB de memória RAM, onde para uma aplicação embarcada o uso de memória é um fator importante.

Tanto na Raspberry Pi, quanto na Beaglebone a instalação do servidor web lighttpd é realizada de forma similar, as diferenças são em relação ao Beaglebone em que a imagem instalada do Angstrom constem alguns softwares que utilizam a porta de comunicação web, 80. Desta forma é necessário, antes da instalação, proceder com a desativação desses softwares que rodam como serviço.

Os sistemas que rodam na beaglebone funcionando similarmente a um servidor web, são o Cloud9, servidor de edição via navegador de scripts para programação, bone101, página web acessada que possui informações sobre a beaglebone, gateone, sistema de ssh via navegador, e o bonescript que executa os script gerados pela cloud9. Para desativá-los execute os seguintes comandos:

```
systemctl disable cloud9.service
systemctl disable bone101.service
systemctl disable gateone.service
systemctl disable bonescript.service
systemctl disable bonescript.socket
systemctl disable bonescript-autorun.service
```

Após esse procedimento, é extremamente recomendado reiniciar o sistema. Para isso execute o comando:

```
shutdown -r now
```

Retiradas essas aplicações da Beaglebone, pode-se instalar o lighttpd nas placas, para isso conecte-se via ssh à placa. Agora é necessário instalar os pacotes necessários para funcionamento do servidor web. São eles: php, php-cgi, php-cli, lighttpd, lighttpd-module-fastcgi, lighttpd-module-rewrite. Os pacotes de php são necessários para que o sistema possa interpretar arquivos e comandos da linguagem php. Os pacotes referentes ao lighttpd são necessários para rodar o servidor web. Os módulos a ele associados são os fastcgi e o rewrite, o primeiro é o módulo que permite ao servidor rodar aplicações web em php, este módulo para funcionamento necessita do segundo módulo.

Raspberry Pi:

```
sudo apt-get install lighttpd php5-cgi
sudo lighttpd-enable-mod fastcgi fastcgi-php
```

. Beaglebone:

```
opkg install php php-cgi php-cli
opkg install lighttpd lighttpd-module-fastcgi lighttpd-module-rewrite
```

Configuração do servidor web

É importante verificar a instalação do servidor, e do sistema php. Para isso podemos executar o comando **php -v** que verifica a versão do software php instalado, e a função **php-cgi -v** que verifica se o php-cgi está devidamente instalado.

As saídas desses comandos é mostrada abaixo.

```
# php -v
PHP 5.3.6 (cli) (built: May 15 2012 11:29:56)
Copyright (c) 1997-2011 The PHP Group
Zend Engine v2.3.0, Copyright (c) 1998-2011 Zend Technologies
```

```
# php-cgi -v
PHP 5.3.6 (cgi-fcgi) (built: May 15 2012 11:29:56)
Copyright (c) 1997-2011 The PHP Group
Zend Engine v2.3.0, Copyright (c) 1998-2011 Zend Technologies
```

A configuração do servidor exige o conhecimento sobre onde foi instalado o php-cgi. Para obter o diretório onde este se encontra a aplicação basta executar o seguinte comando:

```
$ which php-cgi
/usr/bin/php-cgi
```

A configuração do lighttpd é realizada pela modificação do arquivo lighttpd.conf que se encontra na pasta /etc. No arquivo é necessário habilitar o modulo fastcgi, instalado ao lighttpd. Para isso descomente as seguintes linhas. Comentários são determinados pelo sustenido (“#”).

```
server.modules              = (
                                    "mod_rewrite",
#                                   "mod_redirect",
#                                   "mod_alias",
                                    "mod_access",
#                                   "mod_cml",
#                                   "mod_trigger_b4_dl",
#                                   "mod_auth",
#                                   "mod_status",
#                                   "mod_setenv",
                                    "mod_fastcgi",
#                                   "mod_proxy",
#                                   "mod_simple_vhost",
)
```

Mais adiante no arquivo é necessário descomentar mais uma parte para funcionamento do módulo. Nessa parte devemos indicar o diretório do php-cgi.

```
fastcgi.server             = ( ".php" =>
                               ( "localhost" =>
                                 (
                                   "socket" => "/tmp/php-fastcgi.socket",
                                   "bin-path" => "/usr/bin/php-cgi"
                                 )
                               )
                             )
```

A linha “bin-path” deve conter o diretório da php-cgi, obtido anteriormente. Caso não esteja correto é necessário modificá-lo.

Desta forma, configuramos o servidor para funcionamento. Após as configurações deve-se reiniciar o servidor para perpetuação das configurações. Para tal, execute:

```
/etc/init.d/lighttpd restart
```

O sistema agora deve estar online. Para verificar basta abrir o navegador e inserir nele o IP da placa.

Deverá aparecer a página com os dizeres **“It Woks!”.**

Para testar o funcionamento do PHP, pode-se executar um arquivo php para teste.

As páginas do lighttpd, por padrão ficam localizadas no diretório /www/pages, mas pode ser alterado pela edição do lighttpd.conf.

Para criar uma página php de teste execute o seguinte comando.

```
$ echo "<?php phpinfo(); ?>" > /www/pages/phpinfo.php
```

Para acessar a página de teste, basta digitar no navegador de qualquer computador que esteja conectado na rede onde a beaglebone está conectada e acessar o arquivo phpinfo.php. Ao acessar o arquivo através do link http://<IP da plataforma ARM>/phpinfo.php o navegador exibirá uma página com informações da versão do PHP instalada. Se esta página for exibida significa que tanto o servidor quanto o módulo PHP do servidor estão devidamente instalados e funcionando e poderão suportar a interface web desenvolvida.

CAPÍTULO 4: RESULTADOS

4.1. Resultados Obtidos

Este projeto resultou em um robô que pode ser controlado, via web por duas plataformas diferentes. Seu sistema web desenvolvido com PHP, HTML e CSS, permite ao usuário controlar os movimentos do robô, tal como direção, e permite ao usuário a visualização pelo robô através de streaming de vídeo. Um sensor de distância mostra a cada atualização de página a distância do robô a um obstáculo.

O resultado do desenvolvimento do sistema foi muito similar em ambas as plataformas. A instalação do servidor web, lighttpd, e do servidor de *streaming, mjpg-streamer,* foi idêntica em ambos, apesar de ter de desativar algumas aplicações já utilizadas na beaglebone para funcionamento. Esse fato facilita a integração de aplicações para utilização em conjunto.

Os códigos gerados para as plataformas foram diferentes em implementação, mas idênticos em concepção e lógica. A Beaglebone facilita em temos de portabilidade para a linguagem de programação, pois o acesso aos pinos pode ser feito através da leitura e escrita em arquivos. A Raspberry Pi, para este projeto, nesse mesmo quesito, não é fácil, pois os acessos aos pinos é feito através do mapeamento de memória, o que torna o entendimento dessa biblioteca mais complexo e dependente do processador. Bibliotecas existentes facilitam o trabalho para utilização, mas máscara ao usuário seu funcionamento, dificultando a compreensão.

4.2. Dificuldades e Limitações

Uma das maiores dificuldades na realização deste projeto foi o desenvolvimento e aplicação dos conhecimentos em eletrônica para a construção do projeto. A leitura dos *datasheets* e compreensão dos mesmos para utilização e entendimento do funcionamento para o driver não foi simples. Isso ocorreu devido a pouca prática e contato do desenvolvedor deste projeto nessa área.

A dificuldade no desenvolvimento do esquemático e layout do sistema impossibilitou uma comparação mais detalhada das plataformas utilizando, por exemplo, métodos e conceitos aprendidos em Avaliação e Desempenho de Sistemas Computacionais.

4.3. Considerações Finais

Ambas as plataformas tiveram um desempenho bom para o sistema robótico desenvolvido. A Tabela **1** apresenta algumas diferenças entre as plataformas que podem ser relevantes para as considerações de desenvolvimento de algum sistema que venham a utilizar uma dessas. Baseadas nas pesquisas feitas durante o projeto e de informações dos próprios sites dos fabricantes [17] e [18].

Tabela 1 - Comparativo entre as duas plataformas

	Raspberry Pi Modelo B	**Beaglebone Rev A6**
Preço	U$35	U$89
Processador	ARM 11 (Texas Instruments)	ARM Cortex A8 (Broadcom)
Clock do Processador	700MHz	700MHz
RAM	512MB	256MB
GPIO Digital	8	65
Entrada Analógica	Nenhuma	7 de 12bits cada
PWM	1 (Acessível)	8
I2C	1	2
SPI	1	1
UART	1	5
Ethernet	1/100	1/100
USB	2 USB 2.0	1 USB 2.0

Saída de Vídeo	HDMI, RCA	Nenhuma
Saída de Áudio	HDMI e Analógica	Analógica

Estas comparações podem auxiliar a decisão da escolha baseada na utilização da aplicação a ser construída. Para esse projeto, desconsiderando-se o preço a Beaglebone seria a melhor escolha devido ao número de sinais PWM. Contudo, levando-se em consideração o preço, o custo envolvido para adaptação do circuito do *driver* do motor é muito menor que a diferença do custo entre as duas plataformas, o que torna a Raspberry Pi uma melhor escolha. A mostra o robô montado com todos seus componentes.

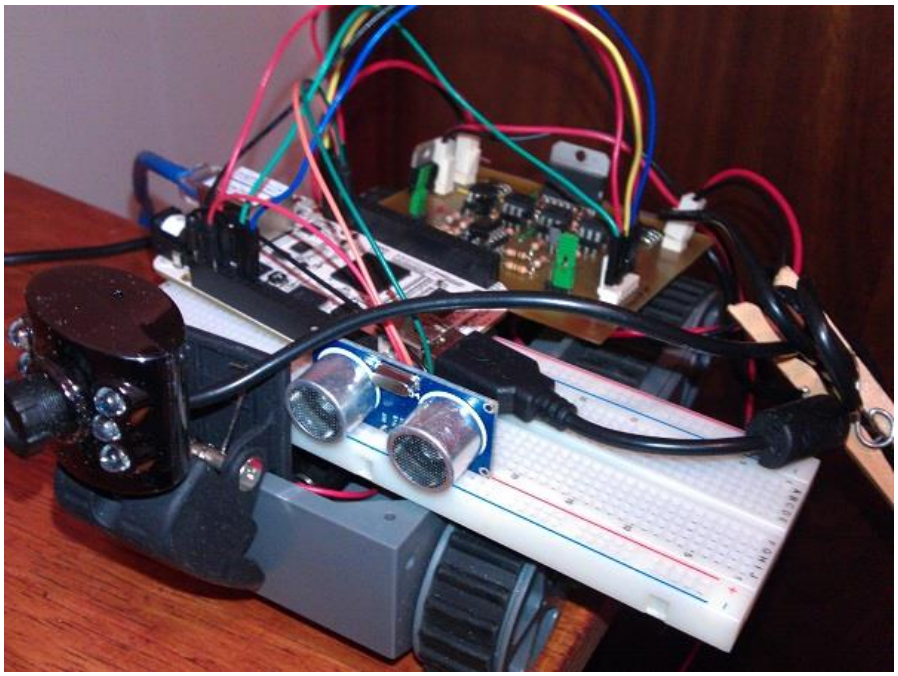

Figura 24 - Robô concluído.

CAPÍTULO 5: CONCLUSÃO

5.1. Contribuições

Neste trabalho foi apresentada uma aplicação construída para duas plataformas ARM. Como observado na realização do trabalho, as diferenças no desenvolvimento para ambas as plataformas diferem na forma de acesso aos pinos. Contudo, a forma de acesso e o ambiente são muito similares, permitindo descrever todo o processo de instalação em um mesmo conjunto, destacando apenas algumas diferenças, quando necessário.

O projeto permitiu mostrar a importância do correto acoplamento entre diferentes módulos em um sistema robótico, pois um comando que mande, por exemplo, o robô para frente, deve realmente mandar e fazê-lo ir para frente. Além disso, todos os requisitos são importantes e envolvem desde a elaboração do projeto em termos de hardware quanto a utilização e desenvolvimento dos softwares.

5.2. Relacionamento entre o Curso e o Projeto

A realização desse projeto possibilitou a utilização de diversos conceitos, ferramentas e técnicas aprendidas durante o curso de Engenharia de Computação para a formação do sistema final. Os conhecimentos adquiridos em disciplinas de programação, redes de computadores, sistemas embarcados, multimídia e hipermídia, circuitos elétricos e circuitos eletrônicos foram fundamentais para a realização deste projeto, permitindo realizar tarefas de forma mais rápida e prática.

Devido ao tempo escasso, não foi possível aplicar em sua plenitude os conhecimentos de Avaliação e Desempenho de Sistemas Computacionais, o que permitiria fazer uma comparação mais significativa para o projeto.

O curso oferece poucas práticas em relação a projeto e implementação de circuitos elétricos e eletrônicos. O desenvolvimento de esquemáticos, leitura de *datasheets*, análise de sinais e desenvolvimento de *layout* são práticas pouco trabalhadas no cursos e essenciais para a formação de um Engenheiro de Computação.

5.4. Trabalhos Futuros

Projetos futuros podem ser realizados utilizando-se a nova beaglebone, nomeada de beaglebone black [36], mais potente e mais barata do que o modelo utilizado neste projeto. Ela é vendida à U$45,00, e possui um processador da Texas Instruments, Sitara AM335x ARM Cortex-A8 1GHz, 512MB de RAM, 2GB de Flash interna e Saída de vídeo e áudio HDMI.

Pode-se também incorporar ao projeto o uso de outras tecnologias web. Tecnologias como AJAX e Java script que permitiriam uma navegação mais fluida e sem interrupções, diminuindo a carga sobre transferência de arquivos do servidor web.

REFERÊNCIAS

[1] About.com. *The History of the ENIAC Computer*. Disponível em: <http://inventors.about.com/od/estartinventions/a/Eniac.htm>. Acesso em 07 de Maio de 2013.

[2] Britannica.com. *Mauchly, John W.: ENIAC computer and its coinventor, John W. Mauchly*. <http://global.britannica.com/EBchecked/topic/183842/ENIAC>. Acesso em 07 de Maio de 2013.

[3] Instituto Newton C. Braga. *A História do Transistor*. <http://www.newtoncbraga.com.br/index.php/artigos/67-historia-da-eletronica/407-a-historia-do-transistor.html> Acesso em 07 de Maio de 2013.

[4] Intel.com [2011]. *Primeiro processador do mundo, o Intel® 4004, comemora seu 40º aniversário*. <http://newsroom.intel.com/community/pt_br/blog/2011/11/19/primeiro-processador-do-mundo-o-intel-4004-comemora-seu-40º-aniversário>. Acesso em 7 de Maio de 2013.

[5] Intel.com. *4004 single chip 4-bit Datasheet*. <http://www.intel.com/Assets/PDF/DataSheet/4004_datasheet.pdf>. Acesso em 07 de Maio de 2013.

[6] Mac Life. *The Lifer: How Far CPUs Have Come -- And Where They're Going*. < http://www.maclife.com/article/columns/lifer_how_far_cpus_have_come_and_where_they're_going>. Acesso em 07 de Maio de 2013.

[7] Nasa.gov. *Discovery Guide: Mars Rover Curiosity*. <http://www.jpl.nasa.gov/education/marsrover.cfm>. Acesso em 07 de Maio de 2013.

[8] Technology for you. How does ABS brake system work. <http://technologyforyou.org/blog/2012/07/01/how-does-abs-break-system-work/>. Acessado em 07 de Maio de 2013.

[9] GreeneComputing. *Linpack for Android Top 10.* < http://www.greenecomputing.com/apps/linpack/linpack-top-10/>. Acesso em 07 de Maio de 2013.

[10] roylongbottom.org. *Linpack Benchmark Results On PCs.* <http://www.roylongbottom.org.uk/linpack%20results.htm>. Acesso em 07 de Maio de 2013.

[11] Samsung. Samsung Galaxy SIII. < http://www.samsung.com/global/galaxys3/>. Acesso em 07 de Maio de 2013.

[12] Samsung. Samsung Exynos 4 Dual 45nm. < https://www.samsung.com/global/business/semiconductor/minisite/Exynos/products4210.html>. Acesso em 07 de Maio de 2013.

[13] tapscape. Disponível em <http://www.tapscape.com/wp-content/uploads/2012/07/SAMSUNG-GALAXY-S3-4.jpg>. Acesso em 07 de Maio de 2013.

[14] Digi-Key. *Digi-Key coporation.* < http://www.digikey.com/>. Acesso em 09 de Maio de 2013.

[15] ARM. ARM The architecture for the Digital World. < http://arm.com/>. Acesso em 09 de Maio de 2013.

[16] GNU.ORG. [2012]. *GNU General Public License*. Disponível em: <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 15 de Maio de 2012.

[17] BeagleBoard.org. *Beaglebone.* <http://beagleboard.org/Products/BeagleBone>. Acesso em 14 de Abril de 2013.

[18] RaspberryPi. *Raspberry Pi.* <http://www.raspberrypi.org/>. Acesso em 14 de Abril de 2013.

[19] Bits & Pieces from the Embedded Design World. *Arduino Uno vs BeagleBone vs Raspberry Pi.* <http://atmelcorporation.wordpress.com/2013/04/26/arduino-uno-vs-beaglebone-vs-raspberry-pi/>. Acesso em 20 de Abril de 2013.

[20] ST [2012]. *L298 Dual full-brigde driver*. Disponível em < http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00000240.pdf>. Acesso em 25 de Abril de 2013.

[21] Beaglebone.org [2012]. *Beaglebone Schematics*. Disponível em <http://beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SCH.pdf>. Acesso em 01 de Maio de 2012.

[22] RaspberryPi.org [2012]. *Raspberry Pi R2.0 Schematics*. Disponível em <http://www.raspberrypi.org/wp-content/uploads/2012/10/Raspberry-Pi-R2.0-Schematics-Issue2.2_027.pdf>. Acesso em 14 de Abril de 2012.

[23] Farchild Semiconductor. *LM78XX/LM78XXA 3-Terminal 1A Positive Voltage Regulator*.Disponível em <http://www.fairchildsemi.com/ds/LM/LM7805.pdf>. Acesso em 20 de Abril de 2012.

[24] Elecfreaks. *Ultrasonic Ranging Module HC-SR04.* Disponível em <http://elecfreaks.com/store/download/HC-SR04.pdf>. Acesso em 28 de Abril de 2012.

[25] Raspberry Pi Spy. *Ultrasonic Distance Measurement Using Python – Part 1*. Disponível em <http://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/>. Acesso em 30 de Abril de 2012.

[26] STREAMINGMEDIA.COM. [2012]. What is streaming? Disponível em: <http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=74052>. Acesso em 10 de Maio de 2013.

[27] STREAMER, Mjpg. [2012]. . Disponível em: <http://sourceforge.net/apps/mediawiki/mjpg-streamer/index.php?title=Main_Page>. Acesso em 12 de Maio de 2013.

[28] ELinux.org [2012]. *Broadcom BCM2835 ARM Peripherals.* Disponível em <http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf>. Acesso em 11 de Abril de 2012.

[29] Projects, Gordon. *WiringPi.* Disponível em <https://projects.drogon.net/raspberry-pi/wiringpi/>. Acesso em 20 de Janeiro de 2013.

[30] GitHub. *BBLIB.* Disponível em <https://github.com/erhs-robotics/BBLIB>. Acesso em 16 de Maio de 2013.

[31] LIGHTTPD. [2012]. *Lighttpd*. Disponível em: <http://www.lighttpd.net/>. Acesso em: 20 de Maio de 2013.

[32] Keay, Andra [2011]. *A Robot, Slave or Companion Species?* Disponível em <http://www.academia.edu/809282/A_Robot_Slave_or_Companion_Species#>. Acesso em 02 de Fevereiro de 2013.

[33] Newitz, Annalee [2010]. *The kinds of slaves that robots will become.* Disponível em <http://io9.com/5667380/the-kinds-of-slaves-that-robots-will-become>. Acesso em 05 de Abril de 2013.

[34] Siegwart, Roland. *Introduction to autonomous mobile robots.* A Bradford Book

2004. ISBN 0-262-19502-X

[34] Adaruit Learning System. Occidentalis v0.2. Disponível em <http://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro/occidentalis-v0-dot-2>. Acesso em 13 de Maio de 2013.

[35] Agnstrom.org. *Beaglebone demo files*. Disponível em <http://downloads.angstrom-distribution.org/demo/beaglebone/>. Acesso em 20 de Maio de 2013.

[36] Beagleboard.org. *Beaglebone Black.* Disponível em <http://beagleboard.org/Products/BeagleBone%20Black>. Acesso em 28 de Maio de 2013.

[37] Aberto até de madrugada. *O que é PWM?*. Disponível em <http://abertoatedemadrugada.com/2013/04/o-que-e-o-pwm.html>. Acesso em 03 de Junho de 2013.

[38] Braga, Newton C. *Ponte-H com controle PWM (mec009).* Disponível em <http://www.newtoncbraga.com.br/index.php/robotica/1213-ponte-h-com-pwm>. Acesso em 03 de Junho de 2013.

[39] Embedded Architects. *O que é um sistema embarcado*. Disponível em <http://www.embarc.com.br/p1600.aspx>. Acesso em 10 de Junho de 2013.

[40] Prado, Sergio [2011]. *A onipresente arquitetura ARM.* Disponível em <http://sergioprado.org/a-onipresente-arquitetura-arm/>. Acesso em 10 de Junho de 2013.

[41] Robot Gear. *RP5 Tracked Chassis – Gray*. Disponóvel em < http://www.robotgear.com.au/Product.aspx/Details/366-RP5-Tracked-Chassis-Gray>. Acesso em 10 de Junho de 2013.

[42] RaspberryPi.org. *About us.* Disponível em <http://www.raspberrypi.org/about>. Acesso em 10 de Junho de 2013.

[43] Labcerter. *Labcenter eletronics*. Disponível em <http://www.labcenter.com/index.cfm>. Acesso em 25 de Junho de 2013.

[44] WiFi.org. *Wi-Fi Allince*. Disponível em <http://www.wi-fi.org/>. Acesso em 25 de Junho de 2013.

APÊNDICE A – Códigos.

Codigos da interface Web

Index.php

```
<!DOCTYPE html>

<?php

///// functions /////

function inc_velocity(){
   $v = shell_exec("./Velocity/inc_velocity.o");
   echo "Velocity = ".$v;
}

function dec_velocity(){
   $v = shell_exec("./Velocity/dec_velocity.o");
   echo 'Velocity = '.$v;
}

function inc_angulo(){
   $v = shell_exec("./Angulo/inc_angulo.o");
   echo 'Angulo = '.$v;
}

function dec_angulo(){
   $v = shell_exec("./Angulo/dec_angulo.o");
   echo 'Angulo = '.$v;
}

function zero_velocity(){
   $v = shell_exec("./Velocity/zero_velocity.o");
   echo 'Velocity = 0';
}


///// START /////

?>

<html>
        <head>
                <meta http-equiv="content-type" content="text/html; charset=UTF-8">
                <title>Treinamento</title>
                <link rel="stylesheet" type="text/css" href="css/template.css">
                <link href='http://fonts.googleapis.com/css?family=Source+Sans+Pro:600'
rel='stylesheet' type='text/css'>
```

```
        </head>
        <body style="background-color = rgb(230, 234, 237);">

                <header id="topo">
                        <div class="container">
                                <a id="logo" href="http://www.google.com/">Robot
Embedded</a>

                                <ul id="menu">
                                        <li class="item active"><a href="">O Robô</a></li>
                                        <li class="item"><a href="">Projetos</a></li>
                                        <li class="item"><a href="">O Time</a></li>
                                        <li class="item"><a href="">Contato</a></li>
                                </ul>
                        </div>
                </header>
                <div id="conteudo-geral">
                        <div id="banner">
                                <div id="banner2"></div>
                        </div>
                        <div id="banner-txt">
                                <div class="container">
                                        <div class="text1">Robot Embedded<br/>
                                                <div class="text2">O robô
embarcado.<br/>Raspberry Pi e BeagleBone</div></div>

<?php

   if (isset($_GET['run']))
      $linkchoice=$_GET['run'];
   else $linkchoice='';

   switch($linkchoice){
      case 'incv' :
         inc_velocity();
         break;

      case 'decv' :
         dec_velocity();
         break;

      case 'inca' :
         inc_angulo();
         break;

      case 'deca' :
         dec_angulo();
```

```
            break;

        case 'zerov' :
            zero_velocity();
            break;

        default :
            /*echo 'no run'; */
    }

?>
                                        <div id="dist"><?php echo shell_exec("cat
distancia.db")." metros" ?></div>
                                        <a class="botao" href="?run=incv" style="top: 30%;
left: 80%;" href="">^</a>
                                        <a class="botao" href="?run=decv" style="top: 50%;
left: 80%;" href="">v</a>
                                        <a class="botao" href="?run=inca" style="top: 40%;
left: 75%;" href=""><</a>
                                        <a class="botao" href="?run=deca" style="top: 40%;
left: 85%;" href="">></a>
                                        <a class="botao" href="?run=zerov" style="top: 60%;
left: 90%;" href="">Parar</a>
                                    </div>
                            </div>
                            <div id="stream"><img        style="height: 420px; width: 540px;"
src="http://beaglebone.local:8080/?action=stream" /></div>
                            <div id="conteudo">
                                    <div class="container">
                                    </div>
                            </div>
                    </div>
            </body>
</html>
```

Template.css

```
body{
        background-color: rgb(230, 234, 237);
        padding: 0px;
        margin: 0px;
}

.container{
        width: 960px;
        margin: auto;
}
```

```
#topo {
	height: 67px;
	position: fixed;
	width: 100%;
	z-index: 101;
	background: url(../img/linha-colorida.gif) bottom repeat-x #f6f3f3;
}

#topo #logo{
	margin-left: 6px;
	display: inline-block;
	width: 128px;
	height: 96px;
	text-indent: -99999px;
	float: left;                                    /*Coloca o logo para flutuar
para a esquerda*/
	background: url(../img/logo.png);
}

#topo #menu{
	float: right;
	margin-top: 10px;
	font-family: "Source Sans Pro";
	padding-right: 0px;
}

#topo #menu .item{
	list-style: none;	/*Retira as bolinhas do list*/
	float: left;		/*Coloca a lista na horizintal*/
}

#topo #menu .item a{
	display: inline-block;
	text-transform: uppercase;
	text-decoration: none;
	font-weight: 600;
	padding: 10px;			/*10 de altura*/
	font-size: 14px;
	height: 29px;
}

#topo #menu .active a{
	color: #ed1a59;
	background: url(../img/menu.gif) center bottom no-repeat;
}

#conteudo-geral{
```

```
        padding-top: 65px;
        /*position: absolute;*/
}

#conteudo-geral #banner{
        height: 610px;
        /*background: url(../img/banner.jpg) no-repeat;*/
        background: url(../img/Rpi-BB.jpg) no-repeat;
        background-size: 100% auto;
        position: relative;

}

#conteudo-geral #banner #banner2{
        /*height: 610px;*/
        width: 100%;
        background: url(../img/ponto.png);
        position: absolute;
        left: 0px;
        top: 0px;
        right: 0px;
        bottom: 0px;
}

#conteudo-geral #banner-txt{
        /*position: relative;*/
}

#conteudo-geral #banner-txt .text1 {
        position: absolute;
        color: #fff;
        padding: 0px;
        height: 140px;
        width: 300px;
        font-size: 60px;
        padding-top: 70px;
        font-family: 'Source Sans Pro', sans-serif;
        left: 1%;
        top: 20%;
        text-align: center;
        text-shadow: black 0.1em 0.1em 0.2em;
        z-index: 96;
}

#conteudo-geral #banner-txt .text1 .text2 {
        position: absolute;
        color: rgb(107, 154, 255);
```

```
        padding: 0px;
        text-align: center;
        height: 60px;
        width: 300px;
        font-size: 20px;
        padding-top: 70px;
        font-family: 'Source Sans Pro', sans-serif;
        left: 0px;
        top: 140px;
        text-shadow: black 0.1em 0.1em 0.2em;
        z-index: 96;
}

#conteudo-geral #banner-txt .botao{
        display: inline-block;
        background-color: #ea4848;
        color: #fff;
        margin-top: 15px;
        padding: 18px 20px;
        font-size: 13px;
        font-family: "Source Sans Pro", arial, sans-serif;
        font-weight: bold;
        text-decoration: none;
        border-radius: 100px;
        -webkit-transition: background 0.2s;
        position: absolute;
        z-index: 96;
}

#conteudo-geral #banner-txt .botao:hover{
        background-color: #000;
        -webkit-transition: background 0.5s;
}

#conteudo-geral #stream{
        height: 420px;/*420px;*/
        width: 540px;
        /*background: url(../img/preto.png) no-repeat;*/
        background: url(http://beaglebone.local:8080/?action=stream) no-repeat;
        background-size: 100% auto;
        position: absolute;
        left: 27%;
        top: 15%;
        z-index: 95;
}

#conteudo-geral #dist{
```

```
        height: 30px;/*420px;*/
        width: 130px;
        background-color: #ea4848;
        color: #fff;
        border-radius: 15px;
        line-height: 30px;
        font-size: 20px;
        background-size: 100% auto;
        text-align: center;
        position: absolute;
        left: 77%;
        top: 20%;
        z-index: 95;
}
```

Códigos comum a ambas as plataformas

inc_angulo.c

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

#define AMAX 10
#define AMIN -10

int main(int argc, char *argv[]){

   FILE *p;
   char as[5];
   int ai;

   //tenta abrir o bd de velocidade
   if ((p = fopen("angulo.db", "r")) == NULL){
      printf("ERRO: Não conseguiu ler!\n");
      return -1;
   }

   fscanf(p, "%s", as);
   ai = atoi(as);

   fclose(p);

   //Velocidade varia de 0 à 100
   if ((ai < AMAX) && (ai >= AMIN)){
      //Escreve o novo valor da velocidade
      if ((p = fopen("angulo.db", "w")) == NULL){
         printf("ERRO: Não conseguiu escrever!\n");
         return -2;
```

```
        }
        fprintf(p, "%d", ++ai);
        fclose(p);
        printf("%d", ai);
        return 0;
    }

    printf("Erro\n");
    return -2;
}
```

dec_angulo.c

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

#define AMAX 10
#define AMIN -10

int main(int argc, char *argv[]){

    FILE *p;
    char as[5];
    int ai;

    //tenta abrir o bd de velocidade
    if ((p = fopen("angulo.db", "r")) == NULL){
        printf("ERRO: Não conseguiu ler!\n");
        return -1;
    }

    fscanf(p, "%s", as);
    ai = atoi(as);

    fclose(p);

    //Velocidade varia de 0 à 100
    if ((ai <= AMAX) && (ai > AMIN)){
        //Escreve o novo valor da velocidade
        if ((p = fopen("angulo.db", "w")) == NULL){
            printf("ERRO: Não conseguiu escrever!\n");
            return -2;
        }
        fprintf(p, "%d", --ai);
        fclose(p);
        printf("%d", ai);
        return 0;
    }
```

```
    printf("Erro\n");
    return -2;
}
```

zero_velocity.c

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

#define VMAX 100
#define VMIN -100

int main(int argc, char *argv[]){

    FILE *p;
    char vs[3];
    int vi;

    //tenta abrir o bd de velocidade
    if ((p = fopen("velocity.db", "w")) == NULL){
        return -1;
    }

    fprintf(p, "0");

    fclose(p);

    if ((p = fopen("angulo.db", "w")) == NULL){
        return -2;
    }

    fprintf(p, "0");

    fclose(p);

    printf("%d", vi);
    return 0;
}
```

inc_velocity.c

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

#define VMAX 100
#define VMIN -100
#define PASSO 5
```

```
int main(int argc, char *argv[]){

   FILE *p;
   char vs[3];
   int vi;

   //tenta abrir o bd de velocidade
   if ((p = fopen("velocity.db", "r")) == NULL){
      printf("ERRO: Não conseguiu ler!\n");
      return -1;
   }

   fscanf(p, "%s", vs);
   vi = atoi(vs);

   fclose(p);

   //Velocidade varia de 0 à 100
   if ((vi < VMAX) && (vi >= VMIN)){
      //Escreve o novo valor da velocidade
      if ((p = fopen("velocity.db", "w")) == NULL){
         printf("ERRO: Não conseguiu escrever!\n");
         return -2;
      }

      fprintf(p, "%d", vi+PASSO);
      fclose(p);
      printf("%d", vi+PASSO);
      return 0;
   }

   printf("Erro\n");

   return -2;
}
```

Dec_velocity.c

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

#define VMAX 100
#define VMIN -100
#define PASSO 5

int main(int argc, char *argv[]){

   FILE *p;
```

```
    char vs[3];
    int vi;

    //tenta abrir o bd de velocidade
    if ((p = fopen("velocity.db", "r")) == NULL){
        printf("ERRO: Não conseguiu ler!\n");
        return -1;
    }

    fscanf(p, "%s", vs);
    vi = atoi(vs);

    fclose(p);

    //Velocidade varia de 0 à 100
    if ((vi <= VMAX) && (vi > VMIN)){
        //Escreve o novo valor da velocidade
        if ((p = fopen("velocity.db", "w")) == NULL){
            printf("ERRO: Não conseguiu escrever!\n");
            return -2;
        }

        fprintf(p, "%d", vi-PASSO);
        fclose(p);
        printf("%d", vi+PASSO);
        return 0;
    }

    printf("ERRO\n");
    return -2;
}
```

Beaglebone

Sonar.c

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include <sched.h>
#include "bbio.c"


int waitforpin(const char *pin, int level)
```

```
{
  struct timeval now, start;
  int done = 0;
  long micros = 0;

  gettimeofday(&start, NULL);

  while (!done)
  {
    if (digitalRead(pin) == level){

      done = 1;

      gettimeofday(&now, NULL);
      if (now.tv_sec > start.tv_sec)
        micros = 1000000L;
      else
        micros = 0;

      micros = micros + (now.tv_usec - start.tv_usec);
    }
  }
  return micros;
}

int wait(int time)
{
  struct timeval now, start;
  int done = 0;
  long micros = 0;

  gettimeofday(&start, NULL);

  while (!done)
  {

    gettimeofday(&now, NULL);
    if (now.tv_sec > start.tv_sec)
      micros = 1000000L;
    else
      micros = 0;

    micros = micros + (now.tv_usec - start.tv_usec);

    if (micros >= time)
      done = 1;
  }
```

```
  //printf("wait: %d us\n", micros);
  return micros;
}

int piHiPri (int pri)
{
  struct sched_param sched ;

  memset (&sched, 0, sizeof(sched)) ;

  if (pri > sched_get_priority_max (SCHED_RR))
    pri = sched_get_priority_max (SCHED_RR) ;

  sched.sched_priority = pri ;
  return sched_setscheduler (0, SCHED_RR, &sched) ;
}


int main (int argc, char *argv[])
{
  int i;
  int pulsewidth1 = 0, pulsewidth2 = 0;
  float distance;

  piHiPri(99);

  exportGpio("P8_5"); //Trigger
  exportGpio("P8_7"); //Echo

  muxPin("P8_5", 7);
  muxPin("P8_7", 27);

  digitalMode("P8_5", OUTPUT);
  digitalMode("P8_7", INPUT);

  for (i = 0; i < 10; i++){
    /* trigger reading */
    wait(1000000); // Espera 1 segundo de leitura em leitura
    digitalWrite("P8_5", 1); //trigger
    wait(10); /* espera 10 us*/
    digitalWrite("P8_5", 0); //trigger
    // Contagem inicia quando Trigger == LOW

    //printf("Pulso no Trigger\n");

    pulsewidth1 = waitforpin("P8_7", 1);
    pulsewidth2 = pulsewidth1 + waitforpin("P8_7", 0);
```

```
    // Leitura termina quando Echo == LOW
    distance = ((float)pulsewidth2 / 58) - 6; //6 parece ser um erro constante que deve ser
ignorado

    printf("echo at %.2f cm and pulsewidth is %d\n", distance, pulsewidth2);

    waitforpin("P8_7", 0);
  }

  unexportGpio("P8_5");
  unexportGpio("P8_7");
}
```

Motor

Motor.c

```
#include <stdlib.h>
#include <stdio.h>
#include "bbio.c"

#define FREQ 150
#define FRENTE 1
#define TRAS 0

int main(int argc, char **argv){
  int p = 0, f=0;
  FILE *fvel, *fang;
  char vs[5], as[5];
  int vi = 0, ai = 0;
  int dir, esq;


  /* Seta pinos como PWM */
  muxPin("P9_14", 6);
  muxPin("P9_16", 6);

  /* Seta pinos como GPIO */
  muxPin("P8_3", 7);
  muxPin("P8_4", 7);

  exportGpio("P8_3"); //sentido Motor 1
  exportGpio("P8_4"); //sentido Motor 2

  digitalMode("P8_3", OUTPUT);
  digitalMode("P8_4", OUTPUT);

  digitalWrite("P8_3", 0);
```

```
digitalWrite("P8_4", 0);

   //(pin,frequency,percent,isrun)
pwmWrite("P9_14", FREQ, 100, 1);
pwmWrite("P9_16", FREQ, 100, 1);

while(1){

 /* Obtenção da velocidade */
 if ((fvel = fopen("../velocity.db", "r")) != NULL){
  fgets(vs, 5, fvel);
  fclose(fvel);
  vi = atoi(vs);
  printf("Velocidade = %d\n", vi);
 }

 vi = 100 - vi;

 /* Obtenção da direção */
 if ((fang = fopen("../angulo.db", "r")) != NULL){
  fgets(as, 5, fang);
  fclose(fang);
  ai = atoi(as);
  printf("Angulo = %d\n", ai);
 }

 dir = (vi+(10*ai));
 printf("dir = %d\n", dir);
 esq = (vi-(10*ai));
 printf("esq = %d\n", esq);

 if (dir <= 100){
  digitalWrite("P8_3", FRENTE);
 } else {
  digitalWrite("P8_3", TRAS);
  dir = 200 - dir;
 }

 if (esq <= 100){
  digitalWrite("P8_4", FRENTE);
 } else {
  digitalWrite("P8_4", TRAS);
  esq = 200 - esq;
 }

 /* Aplicação dos dados obtidos */
 pwmWrite("P9_14", FREQ, dir, 1);
```

```
    pwmWrite("P9_16", FREQ, esq, 1);

    sleep(1);
    printf("Ciclo terminado!\n");

  }

  /* Desliga os motores */
  pwmWrite("P9_14", FREQ, 100, 1);
  pwmWrite("P9_16", FREQ, 100, 1);

  unexportGpio("P8_3");
  unexportGpio("P8_4");

  return 0;
}
```

Raspberry Pi

motor.c

```
#include <stdio.h>
#include <unistd.h>
#include <wiringPi.h>

#define PWM0_OUT 40
#define PWM1_OUT 45

#define Sentido1 1
#define Sentido0 0

int main()
{
  int i=0;

  if (wiringPiSetup () == -1){
    printf("Erro na inicialização do WiringPi\n");
    return 1 ;
  }

  //piHiPri (10) ; sleep (1) ; //Aumenta a prioridade

  pinModeGpio (PWM1_OUT, PWM_OUTPUT);
  pinModeGpio (PWM0_OUT, PWM_OUTPUT);
  pinMode(Sentido1, OUTPUT);
  pinMode(Sentido0, OUTPUT);

  pwmSetModeWPi(PWM_MODE_MS); //MS 32/200 = 3k
```

```
pwmSetRangeWPi(100); //32
pwmSetClockWPi(200); //200]

//pwmWriteGpio(PWM0_OUT, 16);
//pwmWriteGpio(PWM1_OsudoUT, 31);

digitalWrite(Sentido0, 0);
digitalWrite(Sentido1, 0);


while(1){

 /* Obtenção da velocidade */
 if ((fvel = fopen("../velocity.db", "r")) != NULL){
  fgets(vs, 5, fvel);
  fclose(fvel);
  vi = atoi(vs);
  printf("Velocidade = %d\n", vi);
 }

 vi = 100 - vi;

 /* Obtenção da direção */
 if ((fang = fopen("../angulo.db", "r")) != NULL){
  fgets(as, 5, fang);
  fclose(fang);
  ai = atoi(as);
  printf("Angulo = %d\n", ai);
 }

 dir = (vi+(10*ai));
 printf("dir = %d\n", dir);
 esq = (vi-(10*ai));
 printf("esq = %d\n", esq);

 if (dir <= 100){
  digitalWrite(Sentido, 1);
 } else {
  digitalWrite(Sentido, 0);
  dir = 200 - dir;
 }

 if (esq <= 100){
  digitalWrite("P8_4", FRENTE);
 } else {
  digitalWrite("P8_4", TRAS);
```

```
    esq = 200 - esq;
   }

   /* Aplicação dos dados obtidos */
   pwmWriteGpio(PWM1_OUT, dir);
   pwmWriteGpio(PWM0_OUT, esq);

   sleep(1);
   printf("Ciclo terminado!\n");

  }


  pwmWriteGpio(PWM0_OUT, 100);
  pwmWriteGpio(PWM1_OUT, 100);

  return 0 ;
}
```

Sonar.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/time.h>
#include <wiringPi.h>
#define TRIGGER_PIN 23
#define ECHO_PIN  24

int waitforpin(int pin, int level)
{
  struct timeval now, start;
  int done = 0;
  long micros = 0;

  gettimeofday(&start, NULL);

  while (!done)
  {
    if (digitalRead(pin) == level){
      done = 1;

      gettimeofday(&now, NULL);
      if (now.tv_sec > start.tv_sec)
        micros = 1000000L;
      else
        micros = 0;
```

```
      micros = micros + (now.tv_usec - start.tv_usec);
    }
  }
  //printf("waitforpin: %d us\n", micros);
  return micros;
}

int wait(int time)
{
  struct timeval now, start;
  int done = 0;
  long micros = 0;

  gettimeofday(&start, NULL);

  while (!done)
  {
    gettimeofday(&now, NULL);
    if (now.tv_sec > start.tv_sec)
      micros = 1000000L;
    else
      micros = 0;

    micros = micros + (now.tv_usec - start.tv_usec);

    if (micros >= time)
      done = 1;
  }
  //printf("wait: %d us\n", micros);
  return micros;
}

/*int waitforpin(int pin, int level, int timeout)
{
  struct timeval now, start;
  int done;
  long micros;
  gettimeofday(&start, NULL);
  micros = 0;
  done=0;
  while (!done)
  {
    gettimeofday(&now, NULL);
    if (now.tv_sec > start.tv_sec)
      micros = 1000000L;
    else
      micros = 0;
```

```
    micros = micros + (now.tv_usec - start.tv_usec);

    if (micros > timeout)
      done=1;
    if (digitalRead(pin) == level)
      done = 1;
  }
  return micros;
}*/

int main (int argc, char *argv[])
{
  int i;
  int pulsewidth1 = 0, pulsewidth2 = 0;
  float distance;

  piHiPri(99);

  if (wiringPiSetupGpio () == -1){
    fprintf (stderr, "Can't initialise wiringPi: %s\n", strerror (errno)) ;
    return 1 ;
  }

  pinModeGpio(TRIGGER_PIN, OUTPUT);
  pinModeGpio(ECHO_PIN, INPUT);

  for (i = 0; i < 10; i++){
    /* trigger reading */
    wait(1000000); // Espera 1 segundo de leitura em leitura
    digitalWriteGpio(TRIGGER_PIN, HIGH);
    wait(10); /* espera 10 us*/
    digitalWriteGpio(TRIGGER_PIN, LOW);
    // Contagem inicia quando Trigger == LOW

    pulsewidth1 = waitforpin(ECHO_PIN, HIGH);
    pulsewidth2 = pulsewidth1 + waitforpin(ECHO_PIN, LOW);
    // Leitura termina quando Echo == LOW
    distance = ((float)pulsewidth2 / 58) - 6; //6 parece ser um erro constante que deve ser
ignorado

    printf("echo at %.2f cm and pulsewidth is %d\n", distance, pulsewidth2);

    waitforpin(ECHO_PIN, LOW);
  }
}
```

APÊNDICE B – Lista de materiais do Driver do motor.

```
Bill Of Materials
=================
Design:   MotorAmpOp
Doc. no.: 1
Revision: 3
Author:   Daniel Junho
Created:  26/03/13
Modified: 01/06/13

QTY  PART-REFS           VALUE               CODE
---  ---------           -----               ----
Modules
-------
2    M1,M2               SIL-100-02

Resistors
---------
8    R1,R5,R6,R8,R9,R12, 10k                 M330R
     R15,R19
4    R2,R10,R16,R20      1k                  M330R
4    R3,R4,R7,R11        330                 M330R
4    R13,R14,R17,R18     1M                  M330R

Integrated Circuits
-------------------
1    U1                  7805
4    U2-U5               OPTOCOUPLER-NPN
1    U6                  LM358

Transistors
-----------
2    Q1,Q2               BC549

Diodes
------
8    D1-D8               1N4148

Miscellaneous
-------------
2    JP1,JP2             JUMPER2
1    L298                L298
1    MOTORPOWER          SIL-100-02
1    RPI-PIN             SIL-100-06
```