Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Jacinto Luis Azevedo Carvalho

Wheelie

Plataforma Auto balanceada em duas rodas

Dissertação de Mestrado

Orientado por:
Doutora Ana Cristina Lopes – IPT/ESTT

Dissertação apresentada ao Instituto Politécnico de Tomar
para o cumprimento dos requisitos necessários
à obtenção do grau de Mestre em
Controlo e Eletrónica Industrial

Agradecimentos

A minha orientadora, Prof. Doutora Ana Cristina Lopes, por ter acreditado em mim e no projeto proposto, bem como ao Prof. Doutor Paulo Coelho e ao Prof. Doutor Gabriel Pires. Por nunca ter duvidado das minhas competências, mesmo quando as coisas não corriam pelo melhor, e por me ter dado liberdade total na execução deste projeto.

Gostaria ainda de agradecer ao Eng.º Pedro Neves por todo o apoio prestado nomeadamente na elaboração das placas de circuito impresso e cedência de material.

Um agradecimento especial aos meus colegas de curso por me terem ajudado a ultrapassar muitas das barreiras encontradas ao longo deste percurso.

Gostaria também de agradecer a todas as outras pessoas que de alguma forma ou por algum outro meio me ajudaram na resolução do projeto.

Por fim, mas não menos importante, aos meus pais (e ao meu irmão), que sempre me apoiaram, proporcionando me todas as condições para que atingisse o sucesso.

Resumo

O “Wheelie”, é uma plataforma inspirada na conhecida plataforma Segway e consiste num meio de transporte de duas rodas inventado por Dean Kamen em Dezembro de 2001 [1].

O controlador funciona a partir do equilíbrio do indivíduo que o utiliza, tendo por base o sistema de controlo do pêndulo invertido. O “Wheelie” integra um conjunto de sensores e atuadores sendo estes últimos atuados por um sistema de controlo baseado no microcontrolador ATMEGA32 que permite ao “Wheelie” o auto-equilíbrio e a deslocação em duas rodas. Os sensores utilizados são micro-giroscópios e acelerómetros. Estes sensores serão responsáveis por observar as mudanças de terreno, a posição do corpo do condutor e a velocidade e posição do sistema, respetivamente. Para que o “Wheelie” se movimente para a frente, o indivíduo tem de se inclinar para a frente e para que recue é necessária uma inclinação para trás. Para virar, basta oscilar o braço central do “Wheelie” para o lado pretendido. A utilização deste tipo de plataformas é vocacionada para a mobilidade urbana, onde poderá contribuir para uma redução de emissões de gases de efeito de estufa significativa, especialmente quando comparado com qualquer outro veículo motorizado, ou híbrido.

Nota: Toda esta dissertação de projeto foi escrita ao abrigo do Novo Acordo Ortográfico

Abstract

Wheelie is a two-wheel transportation platform inspired in the well-known platform designated by Segway. It was invented by Dean Kamen in December 2001 [1]. The Segway is based on a self-balance controller, and its dynamics is similar to the classic control problem of the inverted pendulum. The hardware architecture of the wheelie platform includes a set of sensors, responsible for internal and environment perception, and actuators. The control system based on ATMEGA32 microcontroller is the system core that allows the "Wheelie" self-balance and moving on two wheels. Micro-gyroscopes, accelerometers and magnetic sensors were the sensors chosen for perception. These sensors are respectively responsible for detecting changes in terrain, the position of the driver's body and the speed and position of the system. For the "Wheelie" to move forward, the individual has to lean forward, and pull back when a backward movement is required. To turn, the operator must simply swing the central arm of the "Wheelie" in the direction of the desired side. The use of such platforms is oriented to urban mobility, which could represent a significant reduction of greenhouse gases emissions, especially when compared to any other powered transportation system.

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas

Lista de abreviaturas (ordenadas por ordem alfabética)

ATMega32	Microcontrolador com 32k de memoria *flash*.
ATMega8	Microcontrolador com 8k de memoria *flash*.
ADC	Analog to Digital Converter
DARPA	Defence Advanced Research Projects Agency.
GM	General Motors.
IHMC	Institute for Human & Machine Cognition.
Open-Source	Que não é pago, aberto para outros utilizadores.
PC	Computador.
PT	Personal Transporter (Segway).
PWM	Pulse With Modulation.
RMP	Robotic Mobility Platform (Segway).
Segway	Meio de transporte elétrico que se auto-equilibra em duas rodas.
SMD	Surface Mounting Devices. Um componente SMD é geralmente menor do que seu equivalente convencional.

Capítulo 1

Introdução

1.1 - Enquadramento da dissertação

Atualmente mobilidade é uma condição necessária ao nosso quotidiano. As novas tecnologias têm vindo a desempenhar um papel importante, no sentido de se atingir uma mobilidade mais rápida e mais ecológica. Com a crise económica e o consecutivo aumento do petróleo é imperativo a busca de novas e mais económicas formas de mobilidade.

Recentemente surgiu uma tecnologia que visa dar resposta aos desafios anteriores (economia e ecologia) designadamente o veículo de duas rodas auto-balanceado, normalmente chamado de "Wheelie" (*Segway)*. Esta é uma tecnologia cada vez mais utilizada quer na mobilidade de pessoas quer de objetos em plataformas robotizadas, uma vez que estes veículos podem apresentar uma mobilidade rápida e económica num meio urbano. Este sistema tem como principal vantagem, face a outros tipos de sistemas de locomoção móvel mais tradicionais, o facto de serem sistemas com uma boa autonomia e boa velocidade de deslocamento, de referir que são sistemas capazes de manter o equilíbrio mesmo sujeitos a grandes perturbações, fator este fundamental quando se está a falar na mobilidade de uma pessoa.

Logo para concluir podemos dizer que as principais vantagens deste tipo de veiculo são:

- Consumos - ao contrário dos veículos motorizados, este não necessita de ser alimentado pelos recursos energéticos normais, como a gasolina ou o gás. É uma vantagem importante, pois implica diretamente uma vantagem a nível do utilizador, que implica numa grande redução de

custos; e indiretamente a nível global, dado que não existem emissões diretas poluentes.

- Recarregar - Para recarregar as suas baterias não é preciso ir a nenhum local nem ter nenhuma máquina específica para a função, basta ligar a uma tomada caseira e deixar que este se carregue totalmente.
- Barulho – como veículo elétrico que é, o ruído que produz é muito baixo, o que se traduz numa vantagem para a população em geral, devido à redução significativa de poluição sonora.

1.2 - Objetivo da dissertação

A presente dissertação tem como finalidade o desenvolvimento de uma plataforma auto-balanceda designada "Wheelie" em que o objetivo deste é aumentar a mobilidade de pessoas em ambientes urbanos. O "Wheelie" deverá ter uma autonomia considerável, mais ou menos 20Km dependendo do modo de utilização, velocidade e inclinação do percurso a efetuar. A dissertação tem como base um circuito já editado e disponível comercialmente. Numa primeira fase do projeto será realizada toda a pesquisa do hardware necessário ao desenvolvimento do sistema. Numa segunda fase será feita a análise do controlador e desenvolvimento do software de controlo. A terceira e ultima fase consiste no desenvolvimento de um sistema de navegação semi-autónoma reativa, baseada em sensores de ultrassons e integração no sistema atual:

Fase 1 - desenvolvimento do hardware necessário:

- Análise e compilação do material necessário para implementação do projeto proposto;

- Análise do sistema de atuação e andares de potência (Motores + Pontes em H);

- Análise do sistema de perceção: acelerómetro, giroscópio, sensores de corrente para efeitos de limitação de corrente e possivelmente codificadores óticos (a parte dos codificadores óticos é uma tarefa opcional do projeto);

- Integração com o microcontrolador ATMEGA32;

Fase 2 - desenvolvimento do software necessário:

- Definição de parâmetros de entrada e saída;

- Estudo do microcontrolador ATMEGA32;

- Desenvolvimento de um controlador simples do sistema;

- Programação do microcontrolador;

- Testes experimentais;

Fase 3 - desenvolvimento do sistema de navegação semi-autónoma:

- Implantação dos sensores magnéticos;
- Teste com os sensores magnéticos;
- Implantação dos sensores de ultrassons;
- Teste com os sensores de ultrassons;
- Teste finais com todos os sistemas implementados;

1.3 - Estratégia

O desenvolvimento deste trabalho teve por base o projeto de Licenciatura [2]. Essa estrutura era relativamente mais reduzida (com o objetivo de redução de custos), mas em que o controlador é o mesmo, visto que este foi desenvolvido de modo a poder ser futuramente adaptado ao projeto com motores de 500W, permitindo deste modo suportar a estrutura e consequentemente o transporte de uma pessoa. Foi adquirido um kit, para desenvolvimento do projeto, visto já existir no mercado uma plataforma comercial, e este ficar mais barato do que o desenvolvimento de raiz.

No entanto, foi necessário a montagem e alterações substanciais da plataforma adquirida, visto que esta revelou diversos problemas na sua montagem, sendo necessário o desenvolvimento de novas soluções de hardware para aumentar a robustez do sistema adquirido.

1.4 - Organização do documento

O documento está organizado em nove capítulos onde se descreve todo o trabalho realizado. Os conteúdos serão descritos de forma sucinta na presente secção. Todos os capítulos estão estruturados com uma introdução ao tema a apresentar.

O primeiro capítulo é o capítulo introdutório, o qual se destina a fazer o enquadramento da dissertação, assim como definir quais os objetivos pretendidos com a realização desta dissertação.

No capítulo dois é feito o estudo do estado da arte no que diz respeito à tecnologia de auto-equilíbrio em duas rodas, assim como, de robôs de telepresença. Serão ainda analisados os projetos mais relevantes da atualidade.

O capítulo três descreve a modelação do sistema, em que é feito em primeiro lugar a análise do problema, de seguida realiza-se a análise de forças e sistema de equações, da qual se retira após a linearização do sistema as equações de espaços de estados e a função de transferência. Após definidos os parâmetros do sistema é feita a análise do sistema, sem e com compensação e com controlador PID.

No capítulo quatro é feita a descrição do Funcionamento do "Wheelie", sendo explicado o modo como os diversos componentes interagem para o seu bom funcionamento. Também é feita a descrição das características técnicas dos diversos componentes (sensores, atuadores, e microcontroladores).

No capítulo cinco é descrito a montagem, bem como todos os sistemas desenvolvidos para o melhoramento da plataforma, de modo a termos um sistema de navegação semi-autónoma.

No capítulo seis é feita a descrição do software de controlo, onde se descreve as funções do software utilizado no microcontrolador ATmega32.

No capítulo sete é feito o teste à plataforma de modo a testar o funcionamento da mesma e de todos o melhoramentos introduzidos.

No capítulo oito são apresentadas as conclusões da dissertação e futuros desenvolvimentos.

Capítulo 2

Estado da Arte

Neste capítulo analisamos sistemas já existentes envolvendo plataforma auto-balanceadas e sistemas robóticos com auto-equilíbrio em duas rodas.
A presente dissertação tem como base a construção de um *"Wheelie"* com *esquemáticos* já existentes, de forma a desenvolver uma plataforma auto-balanceada. Como tal, pretendia-se com está análise identificar as possíveis vantagens e utilização de um veículo com a capacidade de auto-equilíbrio. Após a análise feita, pode-se concluir que a utilização de uma plataforma auto-balanceada *"Wheelie"*, apresenta vantagens em relação aos tradicionais meios de locomoção, pois podem circular a uma velocidade razoavel em vários tipos de terrenos e são bastante manobráveis.
A análise do veículo Segway® Personal Transporter (PT), serviu a inspiração para o desenvolvimento de um controlador que permita o controlo do "Wheelie". Esta abordagem permite também que o "Wheelie" possa circular a maiores velocidades com segurança.

2.1 - Enquadramento da dissertação

São muitos os conceitos que suportam a tecnologia em que se baseia o funcionamento de um meio de locomoção como o "Wheelie". Uma vez que essa tecnologia permite a construção de uma estrutura, com maior mobilidade, estabilidade e menor dimensão em comparação com outros veículos. O número de plataformas auto-balanceadas com esta tecnologia cresceu principalmente com o aparecimento de uma nova gama de equipamentos, designada Segway® Personal Transporter (PT) [3], e o

Segway® Robotic Mobility Platform (R.M.P.) [4], desenvolvido pela Agência de Projetos de Pesquisa Avançada de Defesa (DARPA – Defence Advanced Research Projects Agency). O "Wheelie" é uma plataforma móvel, que se auto-equilibra em duas rodas, servindo de base móvel para o deslocamento e mobilidade de pessoas e objetos de reduzidas dimensões.

Existem também já no mercado vários modelos de plataformas auto-balanceadas, como é o caso do mais conhecido Segway® e outros de outros tipos de veículos mais sofisticados que utilizam o mesmo conceito de locomoção em duas rodas, como é o caso do AnyBot® [5] ou mesmo do PUMA EN-V Project [6].

Neste capítulo apresenta-se o estado da arte em matéria de desenvolvimento de plataformas auto-balanceadas, não só para as plataformas auto-balanceadas cuja função é o aumento da mobilidade de pessoas, mas também relativamente ao desenvolvimento de R.M.P. de outro tipo (robôs).

2.2 - Plataformas Auto balanceadas em duas rodas

Plataformas auto-balanceadas em duas rodas são veículos que utilizam um complexo sistema de sensores que providenciam a informação necessária para o sistema de controlo mantenha o seu estado de equilíbrio. Quando ocorre uma perturbação, movimento detetado na plataforma, a plataforma através do processamento dos valores recebidos dos sensores, vai analisar e compensar o desequilíbrio através do acionamento dos motores. A movimentação da plataforma utiliza um sistema de equilíbrio que permite que ao desequilibrar-mos a plataforma para a frente, esta vai repor o estado de equilíbrio fazendo-a deslocar para a frente para compensar o desequilíbrio

provocado, o mesmo acontece quando desequilibramos a plataforma para trás.

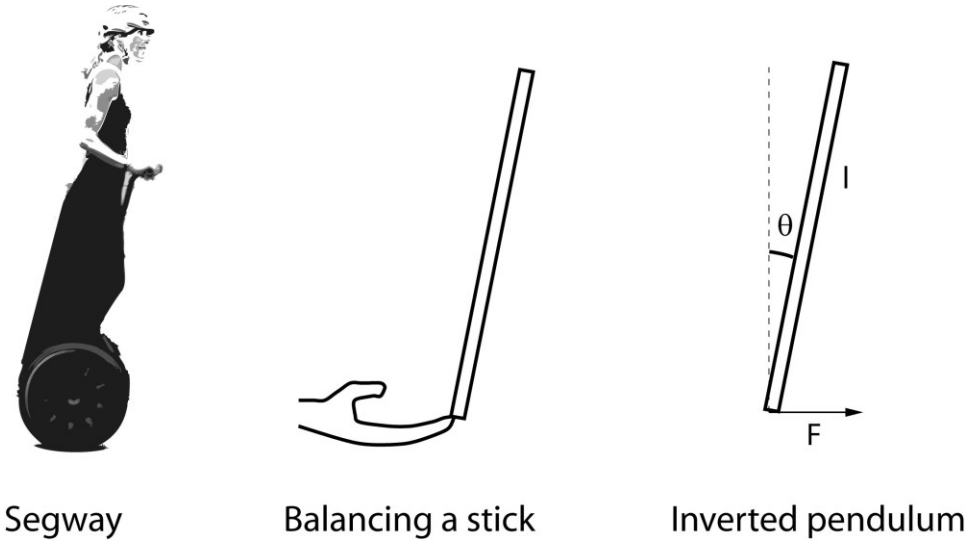


Figura 2.1 – Sistema de pêndulo invertido.

O nosso sistema não é mais do que um sistema de pêndulo invertido, para explicar o descrito acima, temos a Fig. 2.1, no qual verificamos que quando desequilibrado temos de aplicar uma força de compensação no mesmo sentido do desequilíbrio, mantendo assim o sistema em equilíbrio. Como o nosso sistema só tem dois ângulos de desequilíbrio, para a frente e para trás, logo as forças que terão de ser aplicadas, são o movimento para a frente e para trás, respetivamente.

2.3 - Segway® Personal Transporter (PT)

O primeiro veículo auto-balanceado, desenvolvido para transporte de pessoas, foi o Segway® Personal Transporter (PT), inventado por Dean Kamen em 2001 [1]. A sua conceção revolucionária permite que o condutor com uma simples inclinação do corpo, se movimente para a frente e para trás com muita segurança. Para compreender como o sistema funciona, basta ter

em consideração o modelo que o inventor usou para o desenvolvimento da plataforma: o corpo humano. Quando caminhamos, qualquer inclinação realizada pelo nosso corpo, é detetada pelo cérebro devido a uma alteração no fluido do ouvido interno, fazendo com que uma perna se movimente para frente, impedindo a queda. Se continuarmos a inclinar-nos para frente, o cérebro continuará a colocar uma perna à frente, mantendo-nos em pé. Ao invés de cairmos, andamos para a frente, um passo de cada vez. O Segway PT faz praticamente a mesma coisa, mas utiliza as rodas no lugar das pernas, um motor no lugar dos músculos, um grupo de microprocessadores no lugar do cérebro e um conjunto de sensores de inclinação no lugar do sistema de equilíbrio do ouvido interno. Tal como o cérebro, o *Segway* sabe quando o condutor se inclina para frente. Para manter o equilíbrio e impedir a queda, movimenta os motores na velocidade certa fazendo assim o *Segway* andar [7]. Na Figura 2. pode ser visto um dos modelos mais recentes, o Segway PT i2.

Figura 2.2 – Segway PT i2 [3].

2.4 - Segway® Robotic Mobility Platform (R.M.P)

Com o aparecimento do Segway PT muitos investigadores começaram a desenvolver os seus próprios veículos de duas rodas auto-balanceados. Actualmente sistemas de locomoção idênticos ao do *Segway* são muito utilizados em robôs, uma vez que permitem que robôs com grandes estaturas se tornem bastante estáveis, ágeis, rápidos e robustos. Esta tecnologia cresceu principalmente com o aparecimento do Segway® RMP, desenvolvido pela DARPA. O Segway RMP é uma plataforma auto-balanceada, que é utilizada como base para o desenvolvimento de robôs, facilitando assim o desenvolvimento de robôs complexos, uma vez que toda a parte de locomoção pode ser comprada à parte. Existem actualmente vários modelos de Segways RMP. Podem ser vistos na Figura 2.3, os modelos RMP 100 e RMP 200, modelos muito utilizados em projectos de robótica [4].

Figura 2.3 - Segway RMP 100 e RMP 200 [4].

O Segway® RMP tira partido do desempenho demonstrado no Segway® PT. Este oferece um sistema compacto e robusto de locomoção, para aplicações em robótica móvel, está preparado para mover cargas pesadas em pequenos espaços e em vários tipos de terreno. Apesar de parecido com o Segway® PT, o sistema de controlo do Segway® RMP é mais sofisticado, uma vez que já vem equipado com um sistema para controlo de posiçionamento (para evitar colisões com objectos, pessoas ou outros RMP). Já no Segway® PT é o ocupante do veículo que controla a posição do mesmo, através da inclinação do seu corpo.

2.5 - PUMA - EN-V Project

Um protótipo desenvolvido para um projeto denominado *EN-V* ao qual se deu o nome de PUMA, e que surgiu de uma parceria entre a Segway® e a General Motors (GM). Trata-se de um projeto futurista, que tem como base um novo conceito de veículos de transporte urbano para duas pessoas, como substituto para os carros. O veículo baseia-se no funcionamento do *Segway,* equilibrando-se em duas rodas, no entanto tem uma cabine fechada com dois lugares sentados oferecendo o conforto de um carro. Um dos três protótipos criados pode ser visto na Figura 2.3.

O projeto EN-V prevê que em 2030 as pessoas irão utilizar este novo conceito de transporte. Tudo isto devido ao seu tamanho reduzido, à agilidade e sendo também amigo do ambiente [6].

Figura 2.4 – PUMA – EN-V Project [6].

No entanto, o controlo e interface do utilizador com este veículo é diferente do Segway. O condutor possui um volante para controlar o veículo, já não sendo necessário inclinar-se para frente ou para trás para o movimentar, como acontece no Segway PT. No volante existe um sistema semelhante a um *joystick* que quando acionado movimenta um motor linear, existente na base do veículo, que desloca toda a carroçaria para frente e para trás consoante os comandos do condutor. Acionando a movimentação da carroçaria, o condutor consegue mudar o centro de massa, e assim, mover o veículo para frente e para trás. Na Figura 2.4 pode ser visto o motor linear existente na base do veículo.

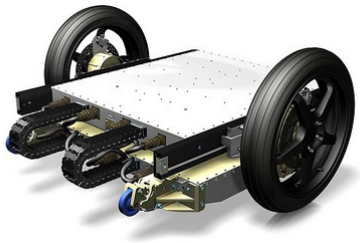


Figura 2.5 – Principio de funcionamento do PUMA [6].

2.6 - Robôs de Telepresença

Os robôs de telepresença são cada vez mais uma realidade hoje em dia, e uma grande parte destes também utilizam o sistema de plataforma auto-balanceada para sua movimentação. O crescimento da tecnologia *wireless* torna possível o controlo remoto deste tipo de robôs através da internet, com a vantagem de ser possível aceder à rede, onde se encontra conectado o robô, em qualquer lugar do mundo. A videoconferência, já é amplamente utilizada por empresas, possibilitando que pessoas de países ou cidades diferentes possam interagir em reuniões de trabalho sem a necessidade de se deslocarem fisicamente ao mesmo local. No entanto a utilização de um robô de telepresença permite um novo tipo de interação com as pessoas e com o lugar envolvente, fazendo com que a pessoa que opera o robô se sinta realmente presente no local, dando-lhe mais liberdade, uma vez que se pode deslocar para onde quiser e conversar com quem entender, mantendo contacto visual com a pessoa em questão. Este novo conceito de telepresença é um real substituto dos atuais sistemas de telepresença por videoconferência. Robôs de telepresença já foram desenvolvidos e testados como auxiliares em hospitais, permitindo que os médicos possam visitar doentes sem estarem presentes no hospital, também têm aplicação como sistemas de vigilância, sistemas de diagnóstico de falhas em subestações isoladas, em prevenção de incêndios florestais e até como ajuda da ida à escola de crianças incapacitadas, em que através de um robô de telepresença podem assistir às aulas e conviver com os colegas a partir de casa.

2.7 - Anybots QB/QA– Your Personal Avatar

O robô de telepresença, Anybots QB, foi desenvolvido pela Anybots, uma empresa de robótica da Califórnia. Desloca-se como um *Segway*, equilibrando-se em duas rodas, o que o torna mais eficiente, manobrável e rápido. A cabeça do robô é apoiada numa barra fina e comprida, que pode ser ajustada conforme a altura desejada (Figura 2.6). Para além da câmara na cabeça de 5-megapixel, possui também uma câmara de baixa resolução apontada para o chão, para facilitar as manobras e impedir colisões. Possui também um laser que permite ao operador do robô apontar para onde quiser, três microfones e colunas de som de alta qualidade. Pode ser controlado remotamente pela internet a partir de um browser em qualquer PC. Possui um sistema de deteção de colisões, motores de potência elevada e baterias de lítio que possibilitam o funcionamento do robô por 8 horas, o suficiente para um dia completo no trabalho [5], [7].

Figura 2.6 – QB Anybots Telepresence Robot [5].

Com o Anybots QB já em comercialização a empresa Anybots encontra-se já a desenvolver uma nova versão desde robô, o Anybots QA, mostrado na (Figura 2.7). O Anybots QA tem praticamente as mesmas funcionalidades que o Anybots QB, no entanto apresenta um design mais atractivo e profissional e tem também a capacidade de se sentar sozinho, para poupar bateria, e voltar a levantar-se de seguida, ficando equilibrado sob as duas rodas [9].

Figura 2.7 – QA Anybots Telepresence Robot [9].

2.8 - Rover: The Mobile Robotic Target System

Na Austrália, o corpo de fuzileiros navais está a testar um novo conceito de treino para os atiradores furtivos, utilizando um robô desenvolvido pela *Marathon Robotics [10]*. O Rover é um sistema de alvo móvel robótico concebido para executar um conjunto pré-programado de atividades, de

forma a criar cenários para o treino dos fuzileiros. A introdução de comandos pré-programados elimina a necessidade de controlar individualmente cada robô, reduzindo o número de operadores necessários para conduzir um cenário. O manequim colocado por cima da plataforma é inclinado para a frente e para trás para o robô se deslocar da mesma forma que uma pessoa faria. A plataforma é blindada e o manequim é feito de um plástico resistente. Quando o robô é atingido, o manequim cai para trás, deslocando-se 90º em relação à plataforma, para dar uma representação visual de que o alvo foi atingido. O manequim pode ser depois colocado na sua posição original automaticamente, com auxílio de um motor que desloca o manequim novamente para a posição vertical.

É fácil perceber a importância do uso de uma plataforma baseada num *Segway* neste tipo de aplicações. Permite a construção de robôs com a mesma estatura de um homem, permitindo simular o deslocamento humano e garantindo o seu equilíbrio. O auto-equilíbrio é no fundo, o ponto-chave para esta aplicação, uma vez que o uso de qualquer outro tipo de plataforma faria com que o robô tombasse ao ser atingido por um projétil [10].

Figura 2.8 – Rover: The Mobile Robotic Target System [8].

Capítulo 3

Modelação do Sistema

3.1 - Demonstração do Problema, Requerimentos de Desenho

O robô de equilíbrio, cujo esquema simplificado é mostrado na Fig. 3.1, e um sistema instável. Teremos que identificar as equações dinâmicas que descrevem o sistema e, considerando que o robô quase não se inclina, linearizá-las para um angulo θ que tende para zero. Em seguida teremos que determinar um controlador que satisfaça os nossos requisitos de equilíbrio.

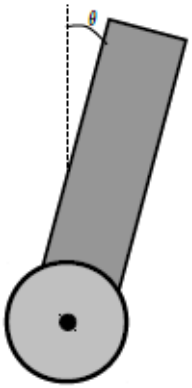


Figura 3.1 – Representação do robô: corpo e eixo de rodas.

Para os nossos cálculos, vamos assumir:
M → massa da roda
m → massa da parte superior, que chamaremos de pêndulo
b → coeficiente de fricção da roda
l → distancia do eixo ao centro de massa
I → momento de inércia do pêndulo
F → força que deve ser aplicada às rodas por um motor
x → posição da roda
θ→ ângulo do pêndulo com a vertical

Os requisitos para esse sistema são:

- tempo de estabelecimento de, no máximo, 1 segundo
- a variação do ângulo não deve exceder 30º, sendo 15º para cada lado.

3.2 - Análise de forças e sistema de equações

Podemos considerar o sistema de equilíbrio do robô como um sistema de dois corpos. Veja na Fig. 3.2 os diagramas dos dois corpos que regem o sistema separadamente.

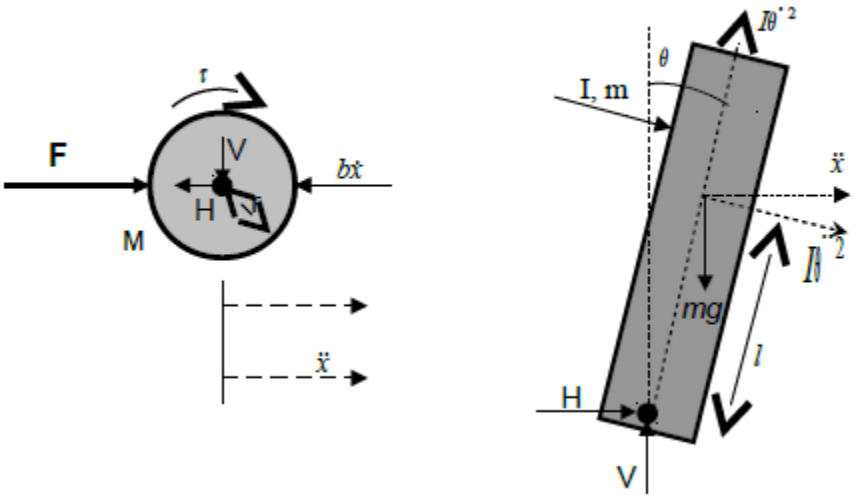


Figura 3.2 – Representação em dois sistemas separados.

Assumindo as forças na horizontal no diagrama da roda, encontramos a seguinte equação:

$$M\ddot{x} + b\dot{x} + H = F \quad (3.1)$$

Note que poderíamos, também, ter somado as forças na vertical, mas não obteríamos informações úteis para o nosso desenvolvimento, pois não há trabalho nesse eixo.

Lembramos também que no nosso caso, não estamos apenas interessados na força que iremos aplicar ao sistema, mas principalmente no binário que devemos aplicar para exercer essa força. Então temos que:

$$\tau = r\,.F = r\,.F\,.\text{sen}\alpha$$

onde r é o raio da roda (3.2)

No nosso caso temos que α será sempre 90º, portanto, o seu seno é unitário. Temos então,

$$\tau = r.F$$

$$\Rightarrow F = \frac{\tau}{r} \qquad (3.3)$$

Vamos continuar a analisar F, pois ainda não decidimos o binário do motor que vamos utilizar, nem o raio da roda.

Somando as forças na horizontal no diagrama do pêndulo da Fig. 3.2, encontra-se a seguinte equação:

$$H = m\ddot{x} - ml\ddot{\theta}.cos\theta - ml\dot{\theta}^2.sen\theta \qquad (3.4)$$

Substituindo a equação (3.4) na equação (3.1), obtemos o seguinte resultado:

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\theta}.cos\theta - ml\dot{\theta}.sen\theta = F \qquad (3.5)$$

Para obter a segunda equação de movimento, somam-se as forças perpendiculares ao pêndulo. Resolvendo o sistema por esse eixo, obtemos a equação (3.6).

$$V.sen\theta + H.cos\theta - mg.sen\theta = -ml\ddot{\theta} + m\ddot{x}.cos\theta \qquad (3.6)$$

Para nos desfazermos dos termos em V e H na equação (3.6), somamos os momentos em torno do centróide do pêndulo.

$$-Vl.sen\theta - Hl.cos\theta = -I\ddot{\theta} \qquad (3.7)$$

Lembramos que o momento de inercia I de um pêndulo invertido ideal pode ser calculado com a fórmula apresentada em (3.8).

$$I = \frac{1}{3}ml^2 \qquad (3.8)$$

Combinando as equações (3.6) e (3.7), conseguimos a segunda equação dinâmica do pêndulo.

$$(I + ml^2)\ddot{\theta} - mgl.sen\theta = ml\ddot{x}.cos\theta \qquad (3.9)$$

Com isso, obtivemos as nossas duas equações dinâmicas que descrevem o sistema. Elas são reescritas em baixo para maior clareza.

$$\begin{bmatrix} (M+m)\ddot{x} + b\dot{x} - ml\ddot{\theta}.cos\theta - ml\dot{\theta}.sen\theta = F \\ (I + ml^2)\ddot{\theta} - mgl.sen\theta = ml\ddot{x}.cos\theta \end{bmatrix} \qquad (3.10)$$

Podemos notar que alguns parâmetros aparecem repetidamente nas expressões acima. Vamos, então, fazer algumas substituições para simplificar as expressões e facilitar a análise.

$$\begin{cases} J = I + ml^2 = \frac{1}{3}ml^2 + ml^2 = \frac{4}{3}ml^2 \\ K = mgl \\ L = ml \\ P = M + m \end{cases} \qquad (3.11)$$

O objetivo, agora, é fazer com que cada equação contenha apenas uma derivada de segunda ordem. Com alguns passos, podemos obter o resultado descrito nas equações (3.12) e (3.13).

$$\begin{cases} \ddot{x} = \frac{-Jb\dot{x} + KL.cos\theta + LJ\dot{\theta}^2.sen\theta + JF}{PJ - L^2.cos^2\theta} \\ \ddot{\theta} = \frac{-Lb\dot{x}.cos\theta + KP.sen\theta + L^2\dot{\theta}^2.cos\theta.sen\theta + LF.cos\theta}{PJ - L^2.cos^2\theta} \end{cases} \qquad (3.12)\ (3.13)$$

3.3 - Linearização do Sistema

Para podermos trabalhar com este sistema de equações e construirmos um modelo de controlo, temos que linearizá-las para tal considera-se.

$$\Theta = 0^o$$

Como pretendemos equilibrar o pêndulo na vertical, o ponto de linearização mais propício corresponde a θ próximo de 0º. Assumimos, então $\Theta = 0^o + \phi$ (onde ϕ representa um ângulo muito pequeno). Assim temos que:

$$\begin{cases} \cos(\theta) = 1 \\ sen(\theta) = \theta \\ \dot{\theta}^2 = 0 \end{cases} \qquad (3.14)$$

Depois de efetuarmos a linearização no conjunto de equações (3.12) e (3.13), as equações de movimento ficam da forma:

$$\begin{cases} \ddot{x} = \frac{-Jb\dot{x}+KL\theta+JF}{PJ-L^2} \\ \ddot{\theta} = \frac{-Lb\dot{x}+KP\theta+LF}{PJ-L^2} \end{cases} \qquad (3.15)\ (3.16)$$

3.4 - Equações de Espaço de Estados

Escrevendo o sistema na forma matricial, obtém-se:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-Jb}{PJ-L^2} & \frac{KL}{PJ-L^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-Lb}{PJ-L^2} & \frac{KP}{PJ-L^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{J}{PJ-L^2} \\ 0 \\ \frac{L}{PJ-L^2} \end{bmatrix} F(t) \qquad (3.17)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} F(t)$$

A matriz C da equação (3.17) é 2 por 4, pois tanto a posição do robô como a inclinação fazem parte da saída. Para o cálculo da equação de espaço de estados, vamos utilizar um sistema multi-saídas, com a posição do robô na primeira linha, e a inclinação na segunda linha.

3.5 - Função de Transferência

Para obter a função de transferência das nossas equações linearizadas, vamos aplicar a transformada de Laplace às equações.

$$\begin{cases} X(s)s^2 = \frac{-JbX(s)s+KL\Theta(s)+JF(s)}{PJ-L^2} \\ \Theta(s)s^2 = \frac{-LbX(s)s+KP\Theta(s)+LF(s)}{PJ-L^2} \end{cases} \quad (3.18)\ (3.19)$$

Isolando X(s) na equação (3.18), obtemos:

$$X(s) = \frac{KL\Theta(s)+JF(s)}{(PJ-L^2)s^2+Jbs} \quad (3.20)$$

Isolando X(s) na equação (3.19), permanecemos com:

$$X(s) = \frac{-(PJ-L^2)\Theta(s)s^2+KP\Theta(s)+LF(s)}{Lbs} \quad (3.21)$$

Igualando as últimas duas equações, teremos:

$$\frac{\Theta(s)}{F(s)} = \frac{L(PJ-L^2)s^2}{(PJ-L^2)^2s^4+Jb(PJ-L^2)s^3-KP(PJ-L^2)s^2-KPJbs+KL^2bs} \quad (3.22)$$

Simplificando e substituindo os valores de (3.11), obtemos:

$$\frac{\Theta(s)}{F(s)} = \frac{\frac{ml}{q}s}{s^3+\frac{(I+ml^2)b}{q}s^2-\frac{(M+m)mgl}{q}s-\frac{mlgb}{q}} \quad (3.23)$$

$$\text{onde} \quad q = (M+m)(I+ml^2) - m^2l^2 \quad (3.24)$$

Se não considerarmos o coeficiente de fricção, isto é, considerando b=0, podemos simplificar nossa função de transferência para:

$$\frac{\Theta(s)}{F(s)} = \frac{\frac{ml}{q}}{s^2 - \frac{(M+m)mgl}{q}} \qquad (3.25)$$

Fazendo uma análise desta função de transferência, percebemos que ela não possui nenhum zero, mas possui dois pólos, sendo estes dados por:

$$s = \pm\sqrt{\frac{(M+m)mgl}{q}} \qquad (3.26)$$

Como temos um dos pólos no lado direito do eixo s, o sistema é claramente instável.

3.6 - Parâmetros do Sistema

Nos próximos passos, serão efetuadas diversas simulações e cálculos de controladores. Por este motivo necessitaremos dos dados reais de nossa estrutura mecânica. Vamos apresentar aqui os valores utilizados para os cálculos.

Item	M	m	l	g	I	b
Valor Real	0,3 kg	4,3 kg	0,153 m	9,8 N	0,00047 kgm2	0,1

Tabela 1 - Valores reais

Agora, podemos calcular o valor de q:

$$q = (M + m)(I + ml^2) - m^2 l^2 = 0.0325 \quad (3.27)$$

Após a substituição dos valores, função de transferência representada na equação (3.23) fica da forma:

$$\frac{\theta(s)}{F(s)} = \frac{20{,}2964}{s^3 + 0{,}3126s^2 - 914{,}9624s - 19{,}8905} \quad (3.28)$$

Considerando o coeficiente de atrito (b=0), temos a equação (3.25) reescrita da seguinte forma:

$$\frac{\theta(s)}{F(s)} = \frac{20{,}2964}{s^2 - 914{,}9624s} \quad (3.29)$$

3.7 - Análise do Sistema sem Compensação

Sem compensação podemos concluir que o nosso sistema é instável. Tal pode ser comprovado pela simulação realizada em ambiente MatLab, a qual se apresenta na Fig. 3.4.

3.7.1 - Considerando coeficiente de atrito

Na Fig. 3.3, apresentamos o nosso sistema completo e considerando uma entrada a impulso.

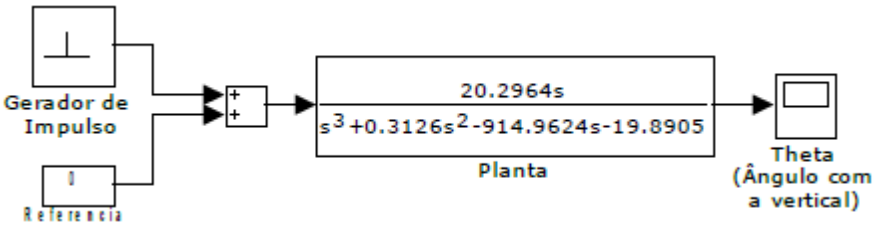


Figura 3.3 – Função de transferência em circuito aberto.

Como já referimos antes, Fig. 3.4 mostra que o sistema sem compensação é instável, por possuir pólos no lado direito do eixo *s*.

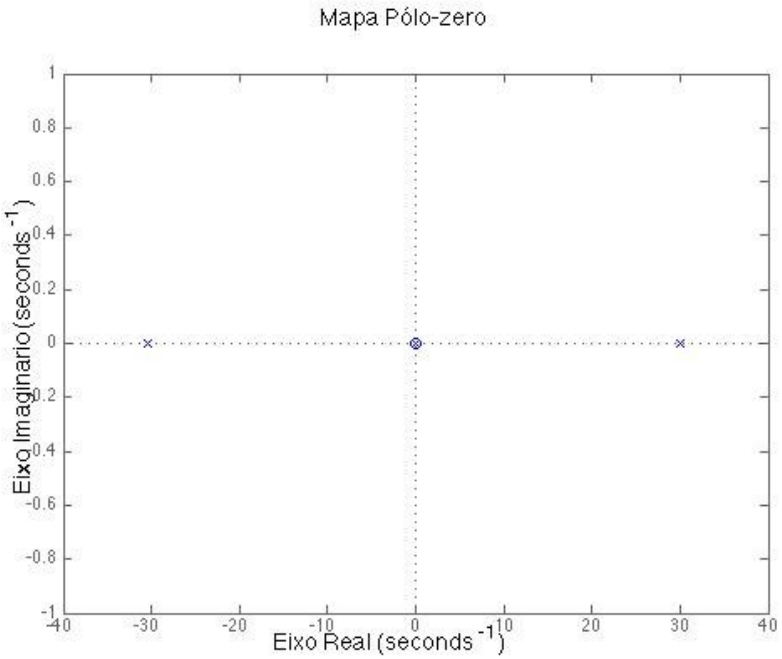


Figura 3.4 – Mapa Pólo-Zero do sistema sem compensação.

Também podemos ver claramente que o sistema é altamente instável através do gráfico da resposta do sistema a um impulso (Fig. 3.5) e a um degrau (Fig. 3.6). Nessas respostas é possível observar que o ângulo diverge rapidamente.

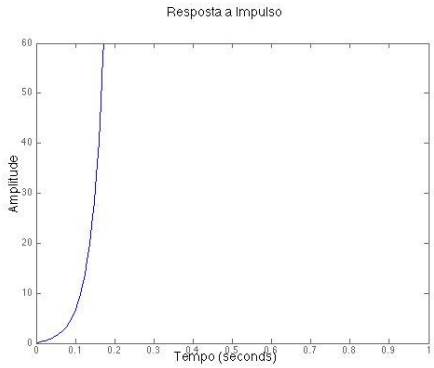


Figura 3.5 – Resposta a um impulso com a função de transferência em circuito aberto.

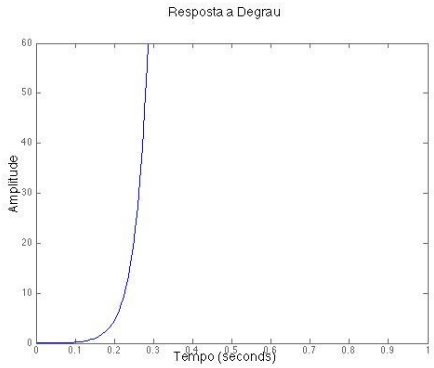


Figura 3.6 – Resposta a um degrau com a função de transferência em circuito aberto.

Existem casos em que um sistema instável em circuito aberto passa a ser estável. Isso ocorre apenas quando o circuito está fechado. O sistema em circuito fechado sem compensação pode ser estudado, olhando o gráfico do Lugar das Raízes aplicado ao sistema. Pode observar esse gráfico na Fig. 3.7.

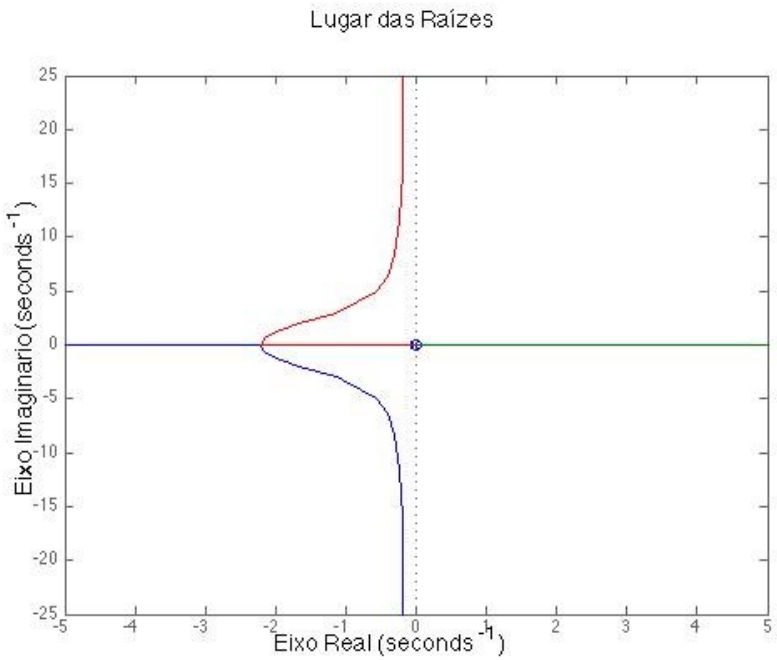


Figura 3.7 – Lugar das Raízes do sistema sem compensação.

O gráfico revela que o sistema não pode ser controlado simplesmente utilizando um circuito fechado de realimentação unitária. Independentemente do valor do ganho aplicado ao circuito em malha fechada, uma parte permanece na região instável do gráfico. Isso torna o sistema impossível de ser controlado utilizando um circuito em malha fechada unitário.

3.7.2 - Desprezando o coeficiente de atrito

Na Fig. 3.8, apresentamos o nosso sistema completo que tem em consideração a equação (3.25), e considerando uma resposta à entrada a impulso.

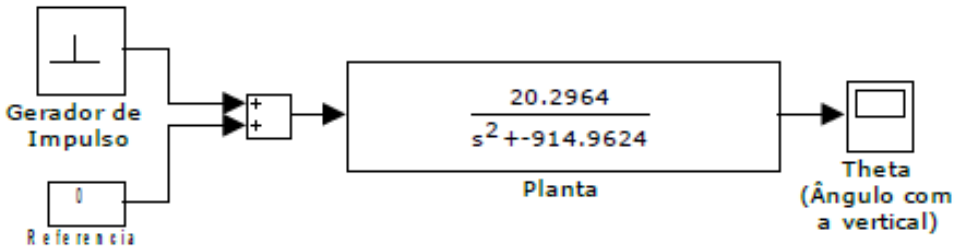


Figura 3.8 – Função de transferência em malha aberta.

A Fig. 3.9 mostra que o sistema sem compensação é instável, por possuir pólos no lado direito do plano no domínio de *s*.

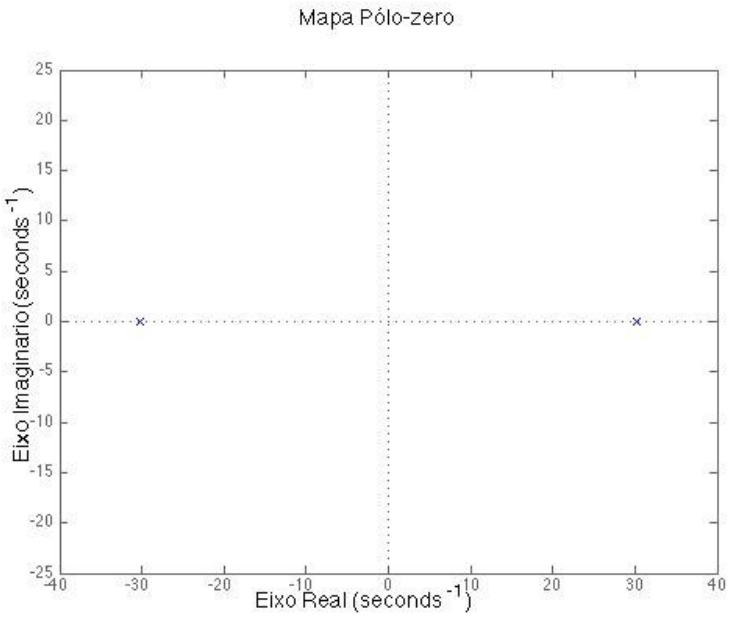


Figura 3.9 – Mapa Pólo-Zero do sistema sem compensação.

Também podemos ver claramente que o sistema é altamente instável através do gráfico da resposta do sistema a um impulso (Fig. 3.10)e a um degrau (Fig. 3.11). Por análise das figuras podemos observar que o ângulo Θ diverge rapidamente.

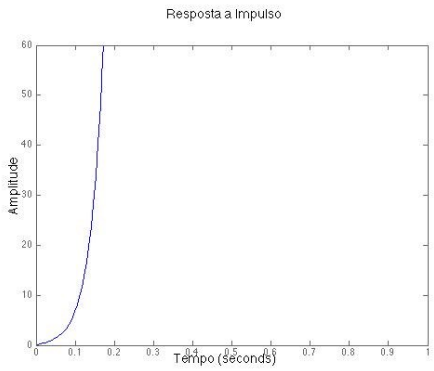


Figura 3.10 – Resposta a um impulso com a função de transferência em circuito aberto.

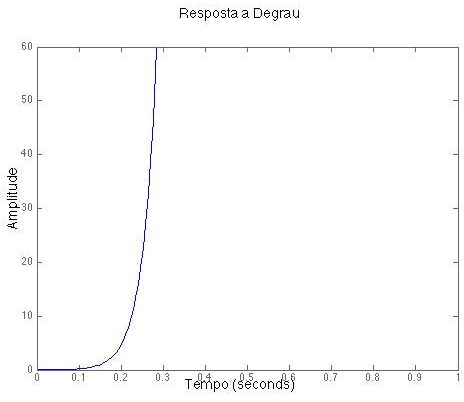


Figura 3.11 – Resposta a um degrau com a função de transferência em circuito aberto.

Existem casos em que um sistema instável em circuito aberto passa a ser estável. Isso ocorre apenas quando o circuito opera em malha fechada. O sistema em malha fechada sem compensação pode ser estudado, olhando o gráfico do lugar das raízes aplicado ao sistema. Pode observar esse gráfico na Fig. 3.12.

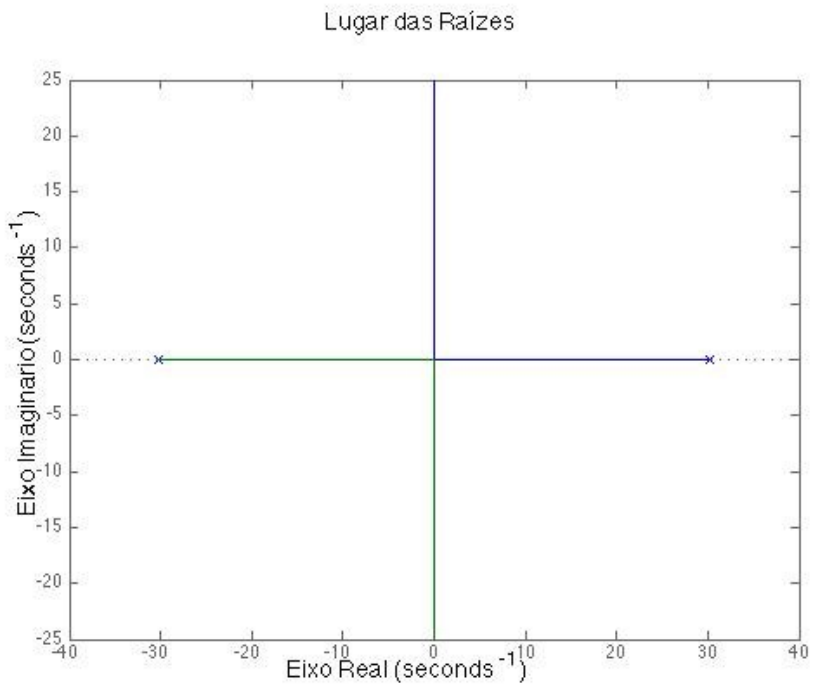


Figura 3.12 – Lugar das Raízes do sistema sem compensação.

O gráfico revela que o sistema não pode ser controlado simplesmente utilizando um circuito em malha fechada com realimentação unitária. Não importando o valor do ganho aplicado ao circuito em malha fechada, uma parte permanece na região instável do plano do domínio de *s*. Isso torna o sistema impossível de ser controlado utilizando a realimentação unitária.

3.8 - Projeto de um Compensador

É necessário um compensador para estabilizar o nosso sistema, pois vimos na sessão anterior que, mesmo com um circuito em malha fechada, o sistema permanece instável. Teremos que ajustar o sistema de modo que o lugar das raízes tenha as suas raízes no lado esquerdo do plano *s*, colocando-as assim na região estável.

Vamos rever aqui nossos objetivos para o sistema:

- Tempo de estabelecimento até 1 segundo.
- *Overshoot* deve ser menor que 10%.
- Fator de amortecimento superior a 0,5.
- O erro (ângulo com a normal) deve ser zero quando estiver estabilizado.

O fator de amortecimento e a frequência natural não amortecida foram determinados de acordo com as equações seguintes.

$$\text{Percentagem de overshoot} = PO = 100e^{-\xi\pi\sqrt{1-\xi^2}} \qquad (3.30)$$

$$\text{Tempo de estabelecimento} = T_s = \frac{4}{\xi\xi W_n} \qquad (3.31)$$

Tendo-se obtido:

$$\xi = 1{,}3$$
$$W_n = 3{,}1 \qquad (3.32)$$

3.8.1 - Compensador Lugar das raízes

Para a manipulação do sistema, através da introdução de pólos e zeros, vamos utilizar a ferramenta *SISO* do Matlab 8. Esta é especialmente desenhada para a compensação de sistemas de uma entrada e uma saída. Na Fig. 3.13 temos o modelo de compensador que iremos utilizar.

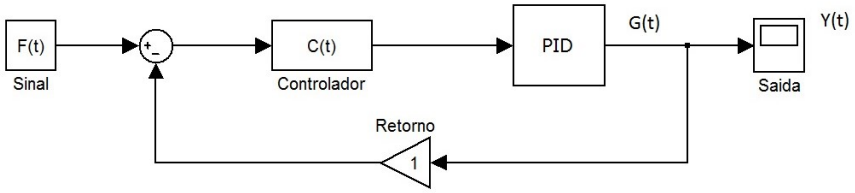


Figura 3.13 – Sistema compensado.

Compensador considerando coeficiente de atrito

Em primeiro lugar, introduzimos um pólo na origem para cancelar o zero. Com isso, temos a seguinte configuração no Lugar das raízes:

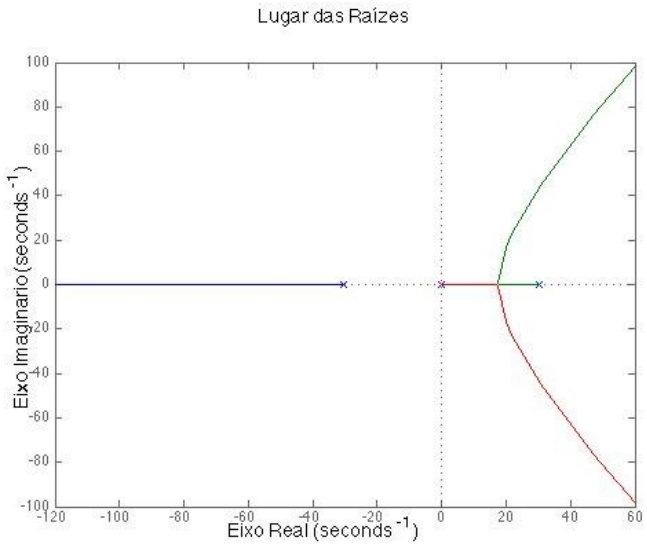


Figura 3.14 – Lugar das Raízes do sistema adicionando um pólo em zero.

Temos que adicionar dois zeros para puxar as curvas que estão a tender para o infinito no lado instável. Colocando dois zeros em -30, temos o seguinte resultado (observar Fig. 3.15).

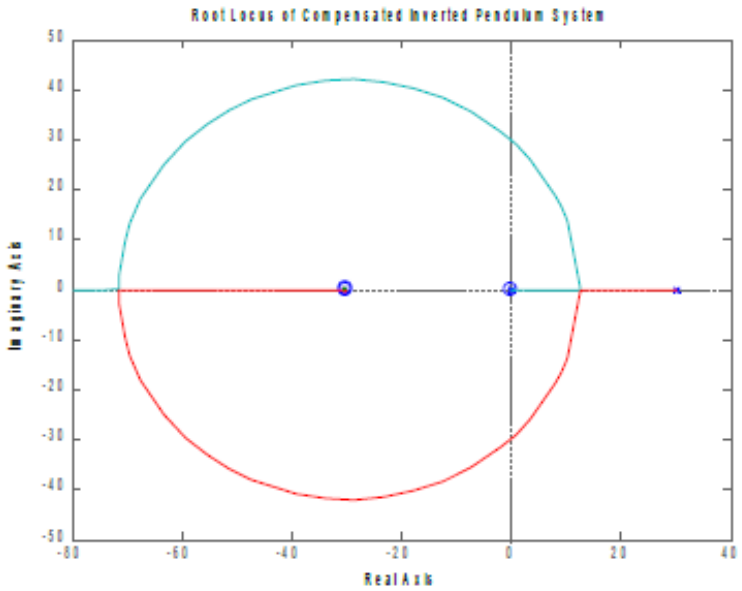


Figura 3.15 – Lugar das Raízes do sistema adicionando mais dois zeros em -30.

A equação do compensador encontrado foi:

$$C(s) = 35.\frac{s^2+60s+900}{s} \qquad (3.33)$$

3.9 - Controlo PID

3.9.1 - Introdução ao controlo PID

Um controlador PID é constituído por três componentes: um termo proporcional, um integral e um derivativo. Podemos dizer então que sua função de transferência teria a seguinte apresentação:

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \qquad (3.34)$$

Nesse caso, Kp é a nossa constante proporcional, Ki é a integral e Kd é a derivativa.

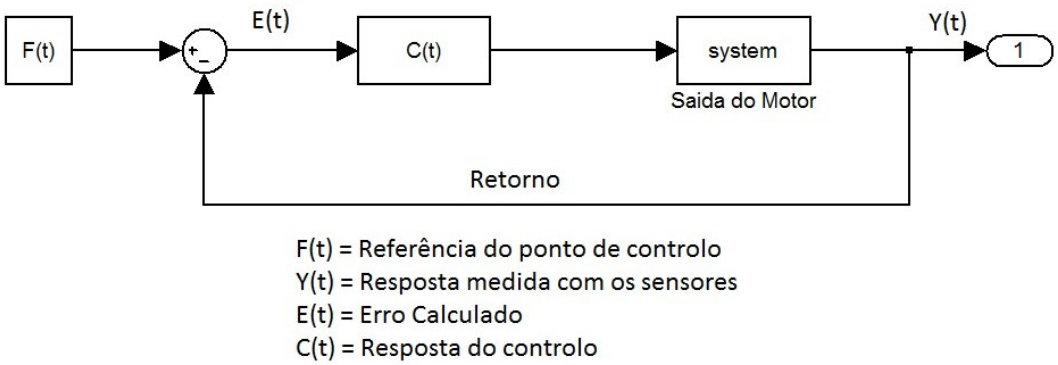


Figura 3.16 – Descrição de um sistema de controlo.

Primeiro, iremos analisar os efeitos de um controlador PID num sistema em malha fechada, tal como mostra a Fig. 3.16. A variável E(t) verifica o erro, mais precisamente, a diferença entre o valor desejado F(t) e a saída real Y(t). A partir do erro é possível determinar a sua derivada e integral. A G(t) será providenciado a soma do erro com seu ganho proporcional Kp, mais a integral do erro com seu ganho integral Ki, mais a derivada do erro com seu ganho Kd.

Para analisarmos os efeitos de cada uma das três constantes sobre o sistema, vamos olhar a função de transferência na forma canónica de segunda ordem de um sistema em malha aberta mostrada abaixo:

$$\frac{1}{s^2+2\xi W_n s+W_n{}^2} \qquad (3.35)$$

A constante Kp reduz o tempo de subida e também reduz o erro com a referência, porém sem nunca eliminá-lo. A constante integral Ki tem a capacidade de eliminar o erro com a referência, porém, a resposta transitória será afetada. Se for necessário utilizar uma componente integral no controlo, devemos sempre tentar um valor baixo. O controlo derivativo irá aumentar a estabilidade do sistema, reduzindo o *overshoot* e melhorando a resposta transitória. Podemos ver um resumo dos efeitos causados pela introdução de um controlo PID na Tabela 2 apresentada em baixo.

Componente	Tempo de subida	*Overshoot*	Tempo de assentamento	Erro c/ R(t)
K_P	Diminui	Aumenta	Não afeta	Diminui
K_I	Diminui	Aumenta	Aumenta	Elimina
K_D	Não afeta	Diminui	Diminui	Não afeta

Tabela 2 - Resposta de um sistema a um controlador PID

É claro que essas informações não são exatas, pois como as constantes são relacionadas, um termo afeta o outro. Esta informação e útil caso se queira determinar os valores de Kp, Ki e Kd por tentativa e erro.

3.10 - Análise do Controlador

Quando comparamos o resultado que encontramos na estabilização do sistema e a equação geral do controlo PID, podemos tirar algumas conclusões. Vamos então analisar as equações (3.33) e (3.34). Repetimo-las abaixo para efeitos visuais:

$$C(s) = 35.\frac{s^2 + 60s + 900}{s}$$

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

Podemos então concluir que:

$$C(s) = \frac{K_D s^2 + K_P s + K_I}{s} = 35.\frac{s^2 + 60s + 900}{s}$$

$$\text{com} \begin{cases} K = 35 \\ K_P = 60 \\ K_D = 1 \\ K_I = 900 \end{cases} \qquad (3.36)$$

3.11 - Aplicação do Controlador ao Sistema

É importante termos em mente que o sistema será implementado com diversas limitações, incluindo um microprocessador de baixo custo. Este é o principal motivo pelo qual devemos utilizar um controlador de simples implementação. A Fig. 3.16 representa o sistema de controlo que iremos utilizar.

Com a entrada F(t) desejada, sendo que θ = 0º, temos o sistema equilibrado. Qualquer valor diferente de 0º representa o erro Y(t) = E(t).

Implementando o sistema de controlo PID, temos três termos baseados na medida do erro:

- Termo Proporcional : $K_I.E(t)$ – onde K_P e a constante proporcional.
- Termo Integral : $K_I . \int_0^t E(t)dt$ – onde K_I e a constante integral.
- Termo Derivativo : $K_D.dE(t)\ dt$ – onde K_D e a constante derivativa.

Juntando os termos, temos a seguinte equação:

$$C(t) = K_P.E(t) + K_I.\int_0^t E(t)dt + K_D\frac{E(t)}{dt} \qquad (3.37)$$

Neste sistema, o sinal de saída do controlador irá definir a direção de rotação do motor. A amplitude de C(t) corresponde diretamente à amplitude do sinal enviado ao motor, definindo a velocidade a que o motor irá rodar.
Outra representação para essa equação pode ser encontrada abaixo:

$$C(t) = K.\left[E(t) + \frac{1}{T_i}.\int E(t)dt + T_d\frac{dE(t)}{dt}\right]$$

$$\text{com}\begin{cases} K_P = K \\ K_I = \frac{K}{T_i} \\ K_D = K.T_d \end{cases} \qquad (3.38)$$

3.12 - Integral *Windup*

Ao representarmos um controlo PID é comum baseá-lo no conceito de linearidade, mas temos que levar em consideração os componentes não lineares do sistema real, tais como o motor. A tensão enviada para o motor tem limites e isso significa que temos que considerar a saturação do sinal enviado ao motor. Se nenhuma ação for tomada no controlo em relação aos limites do atuador, este chegara ao limite, independente da saída do processo. Na componente integral do controlador, o sinal continuará a ser integrado, ficando excessivamente grande, ou *Windup*. O erro deve mudar de sinal durante um tempo prolongado para o sinal voltar ao normal. Isso ocasiona um transitório muito grande na saturação, podendo até instabilizar o sistema.

Para evitar o efeito *Windup*, adicionaremos mais uma realimentação no controlador utilizando como erro e_s a diferença do sinal de saída do atuador e a saída do sinal do controlo. O sinal e_s alimenta o integrador já existente através do sinal $1/T_t$. enquanto não há saturação no sinal do controlador, o erro e_s é zero, mas quando o sinal satura, o erro e_s cresce. A realimentação faz com que esse erro volte a ser zero.

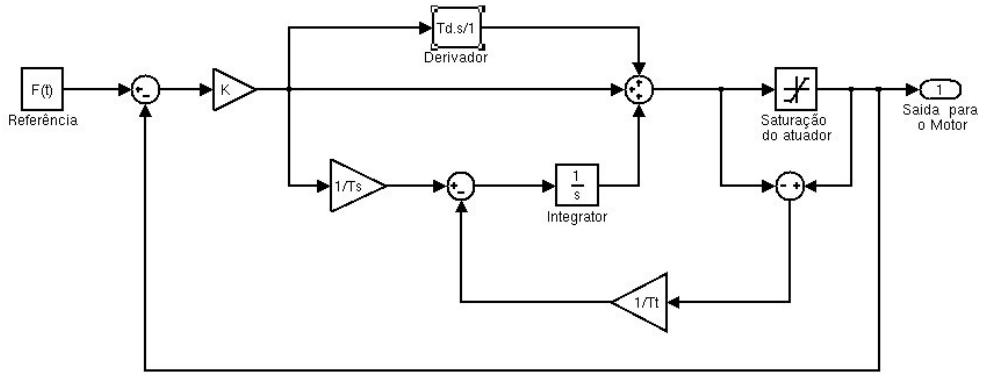


Figura 3.17 – Controlador PID com componente integral *Windup*.

Em termos práticos, os limites de nosso atuador estão no sinal para o motor. Possuímos uma velocidade máxima para cada direção, diretamente relacionada pela tensão. No nosso caso, controlamos a tensão utilizando um sinal modulado por pulsos e um filtro passa baixas (próprio motor). Por isso, os limites de nosso atuador são o menor e o maior pulso que pudermos emitir. Com isso temos uma nova fórmula para o controlador mostrado abaixo:

$$C(t) = K.\left[E(t) + \left(\frac{1}{T_i} + \frac{1}{T_t}e_s(t)\right).\int E(t)dt + T_d\frac{dE(t)}{dt}\right] \qquad (3.39)$$

Para facilitar o *tunning* das constantes, e aconselhado que o valor de T_t seja igual a T_i.

3.13 - Discretização para um Sistema Digital

Programaremos o sistema de controlo num microcontrolador, e serão necessárias algumas aproximações devido a implementação dos termos integral e derivativo.

Termo Proporcional

Não há necessidade de uma aproximação no termo proporcional. O resultado é:

$$P = K.E(t) \quad (3.40)$$

Termo Integral

A seguinte equação pode ser usada para o termo integral:

$$I(t) = K\left(\frac{1}{T_i} + \frac{1}{T_t}e_s\right).\int_0^t E(t)dt = K\left(\frac{1}{T_i} + \frac{1}{T_t}e_s\right)T_s\sum_0^N E(n) \qquad (3.41)$$

Para $T_t = T_i$

$$I(k) = \frac{K}{T_i}.(1 - e_s).T_s\sum_0^N E(n) \qquad (3.42)$$

Aqui T_s é o período de amostragem.
Outro método para se fazer a discretização do sinal e derivando os dois lados da equação (3.41). A seguinte equação pode ser usada para o termo integral:

$$I(t) = K\left(\frac{1}{T_i} + \frac{1}{T_t}e_s\right).\int_0^t E(t)dt = \frac{K}{T_i}.(1 - e_s).\int_0^t E(t)dt \quad (3.43)$$

$$\frac{dI(t)}{dt} = \frac{K}{T_i}.(1 - e_s).E(t) \quad (3.44)$$

Após discretizarmos:

$$\frac{I_{K+1} - I_K}{T_s} = \frac{K}{T_i}.(1 - e_{s_K}).E_K \quad (3.45)$$

$$I_{K+1} = I_{K+}\frac{K}{T_i.F_s}.(1 - e_{s_K}).E_K \quad (3.46)$$

Termo Derivativo

Para o termo derivativo, podemos utilizar a seguinte equação:

$$D(t) = K.T_D.{}^{dE(t)}/_{dt} \quad (3.47)$$

$$D(n) = K.T_D.\frac{[E(n) - E(n-1)]}{T_s} \quad (3.48)$$

Temos E(n) como o erro atual, e E(n-1) como o erro anterior.

Considerando as aproximações nas ultimas duas equações, podemos reescrever C(t) como mostra a equação (3.49).

$$C(t) = K_P.E(t) + K_I.(1 - e_s).T_s\sum_0^N E(n) + K_D.{}^{[E(n) - E(n-1)]}/_{T_S} \quad (3.49)$$

Capítulo 4

Descrição do funcionamento do “Wheelie”

4.1 - Introdução

O “Wheelie” consiste num veículo de duas rodas auto-balanceado para transporte de uma pessoa. O controlo do equilíbrio é obtido através da leitura e processamento de dados por um microprocessador, o Atmega32. Os dados para controlo de velocidade angular e equilíbrio provêm de dois sensores, um acelerómetro e um giroscópio. Após o processamento do algoritmo de controlo o microprocessador envia sinais de comando para dois motores DC, que estão acoplados às rodas por meio de caixas redutoras. Sendo que o veículo pode mover-se em qualquer direção. Todo o projeto foi baseado no “Elektor Wheelie - self-balancing vehicle." [9]

4.2 - Descrição de funcionamento

O Wheelie, apenas se desloca para frente ou para trás em resposta à inclinação da pessoa que o transporta. O sistema de controlo do “Wheelie”, tenta manter o veículo na posição vertical. Se este começar a tombar para frente, o sistema de controlo acelera o veículo nessa direção tentando compensar a queda, mantendo-o assim na vertical. Uma pessoa que se desloca em cima do “Wheelie”, esta constantemente a forçar uma inclinação, fazendo com que o sistema de controlo a tente corrigir, adicionando assim velocidade ao veículo. A direção é controlada pela inclinação da coluna de

direção que ao movimentar-se faz rodar um potenciómetro que indica ao controlador (ATMega32) a direção pretendida.

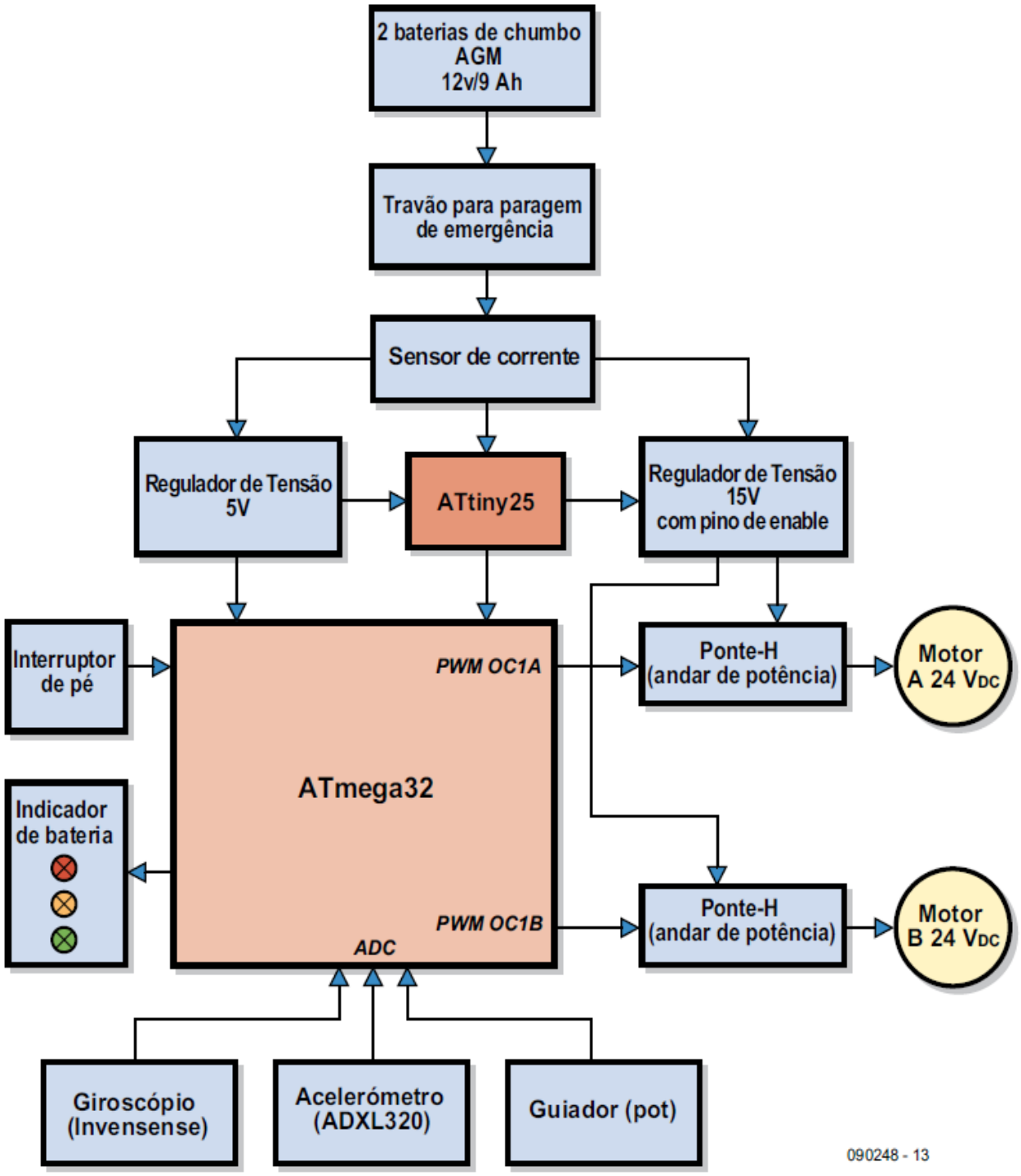


Figura 4.1 – Diagrama de blocos e sentidos de comunicação entre os diferentes

Na figura anterior é mostrado um diagrama de blocos que ilustra a ligação entre os diversos componentes e o sentido do fluxo de informação.
Como descrito anteriormente o movimento do veículo é controlado pela inclinação da sua base, isto porque os sensores de inclinação estão montados na parte inferior da base do veículo e tendo assim esta como referência para o ângulo de inclinação do veículo.
Para o equilíbrio do veículo são usados dois sensores que são o giroscópio IDG500 e o acelerómetro ADXL335, estes dois sensores fazem parte de um circuito implementado pela SparkFun, e para a direção é usado um potenciómetro (Fig.4.2) acoplado ao volante de direção.

Figura 4.2 – Potenciómetro utilizado para aquisição do ângulo da coluna de direção

Os sinais de aceleração provenientes do acelerómetro e de posição do giroscópio são lidos e interpretados pelo microcontrolador ATmega32 que controla a direção e fornece o sinal para gerar a força necessária a cada um dos motores utilizando duas saídas em PWM e MOSFET's em ponte (Fig.4.3 e 4.4), permitindo assim aplicar diferentes velocidades aos motores e realizar curvas ou mesmo fazer o veículo girar sobre ele próprio.
Para efeitos de proteção é utilizado o sensor de efeito de Hall (ACS755 SCB-100) que é vigiado pelo microcontrolador ATtiny25 que, verificando uma corrente excessiva, superior a 100A (amperes) a fluir pelo sistema, possivelmente devido a um curto circuito, corta a tensão de 15V (circuito de

alimentação da ponte) utilizando para esse efeito a entrada de *shutdown* do regulador (MIC2941).

Caso ocorra uma falha total ou o "Wheelie" fique fora de controlo, para proteção do utilizador, é usado um sistema eletromecânico que consiste num interruptor de segurança que se deve encontrar preso ao utilizador que permite que o sistema desligue em caso de queda.

Existe também um interruptor de pé que informa ao ATmega32 que se encontra alguém em cima da plataforma ("Wheelie"), e caso este interruptor deixe de ser pressionado, 2 segundos depois é cortada a alimentação dos motores evitando assim que o veículo se desloque sem controlo.

Em modo normal de funcionamento o ATtiny25 é usado para informar o ATmega32 quando a corrente no motor excede o valor nominal de 100A, o que leva a que o ATmega32 controlo a corrente tentando diminuir os sinais de controlo PWM.

Os sinais provenientes dos sensores acelerómetro, giroscópio e do potenciómetro que se encontra acoplado ao volante são lidos pelas entradas analógicas do ATmega32, às quais se encontra associado um conversor analógico/digital. As entradas analógicas são amostradas a uma frequência de 100Hz (10ms).

A tensão da bateria é também monitorizada pelo ATmega32 noutra entrada do conversor A/D e posteriormente visualizada em leds que indicam o estado de carga das baterias.

O equilíbrio e estabilização, como dito anteriormente, estão a cargo de dois sensores, o acelerómetro de 3 eixos ADXL335 (só vamos utilizar dois, pois a plataforma só trabalha em dois eixos) e do giroscópio de 2 eixos IDG500.

O acelerómetro mede a posição/inclinação do veículo, em relação ao plano horizontal, devido ao ângulo que a gravidade (vertical) atua sobre o plano em que se encontra o dispositivo estando esses valores disponíveis nas saídas

Xout e Yout que variam respetivamente com a inclinação do plano no eixo dos X e dos Y.

O giroscópio mede a velocidade de viragem. Quando a velocidades elevadas apresenta na sua saída uma tensão de valor elevado e que varia rapidamente e quando se encontra parado a tensão na sua saída é aproximadamente metade da tensão de alimentação (3,3V).

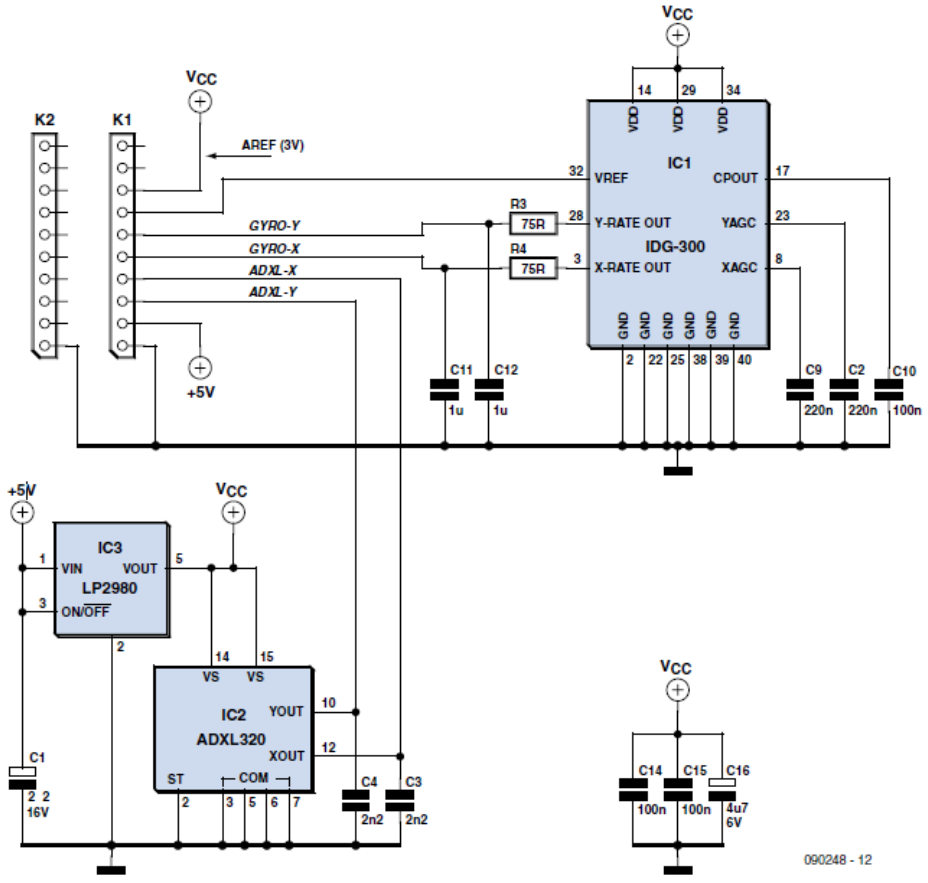


Figura 4.3 - Esquema de ligação do acelerómetro e do giroscópio

A Fig.4.3 mostra o esquema de ligação dos sensores numa placa independente do restante circuito. Esta placa é conectada ao restante circuito pelos conectores K1 e K2 (ver Fig.4.3 e 4.4).

A estabilidade é então obtida determinando a inclinação da plataforma em cada instante de tempo, para isso a saída do acelerómetro é integrada durante um longo período de tempo para se obter um sinal mais suave e posteriormente é adicionado o sinal do giroscópio em proporções otimizadas.

O sinal da aceleração entregue ao controlador do motor é então calculado como uma combinação linear do erro de inclinação ou seja a diferença entre o ângulo de inclinação da plataforma e o ângulo pretendido que será o mais horizontal possível para que o utilizador se equilibre com mais facilidade, em termos práticos quanto maior for a inclinação da base, maior vai ser a velocidade e a potência a entregar aos motores para corrigir o ângulo de inclinação.

$$-Vl.\,sen\theta - Hl.\cos\theta = -I\ddot{\theta} \quad (3.7)$$

Isto pode ser verificado no Capítulo 3 onde é feita a modelagem do sistema do pêndulo da Fig.3.2, em que pela equação, $-Vl.\,sen\theta - Hl.\cos\theta = -I\ddot{\theta}$ (3.7), se pode verificar que quanto maior for θ, o ângulo do pêndulo (l),maior será a força H, que representa a deslocação do pêndulo, visto que $Vl.\,sen\theta$ vai ser praticamente nula, logo podemos concluir, como é descrito acima, que quanto maior for a inclinação da base, maior vai ser a velocidade.

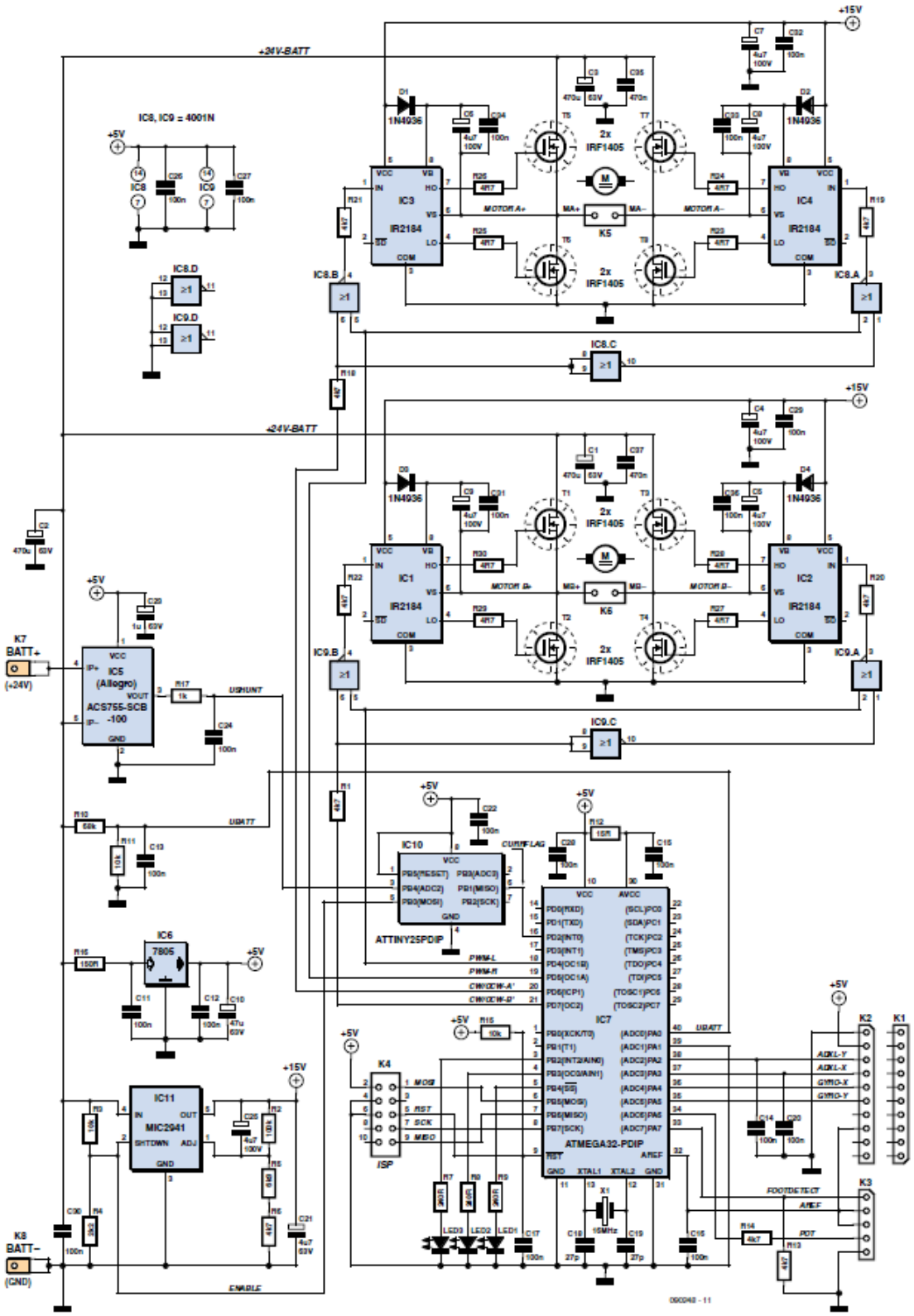


Figura 4.4 - Esquema de ligação do ATmega32 com os restantes dispositivos

O controlo do motor é efetuado pelo esquema indicado na Fig. 4.4 no qual se encontra o Atmega32 com um cristal a funcionar a frequência de 16MHz e diretamente ligado a um conector de programação ISP (conector utilizado para programar o ATmega32).

A placa de sensores encontra-se separada desta e é ligada no conector K2 que por sua vez liga as saídas dos sensores correspondentes aos eixos as entradas ADC2 a ADC5 e no pino 32 (Aref) é aplicada uma tensão de 3V proveniente do regulador de tensão da placa dos sensores.

Uma tensão de 3V é também aplicada através do conector K3 ao potenciómetro que controla a direção e que posteriormente fornece uma tensão na entrada ADC6 que indica a posição do volante.

A tensão da bateria é monitorizada na entrada ADC0 através de um divisor de tensão (R10 e R11).

A posição interruptor do pé é monitorizado pela entra entrada ADC7 que se encontra ligado pelo conector K3.

O sinal de deteção de falha proveniente do circuito integrado que monitoriza a corrente ATtiny25 é aplicado na entrada de interrupção INT0 que por sua vez recebe o sinal proveniente do sensor de efeito de HALL (ACS755-SCB-100) que oferece um funcionamento linear até 100A e através do ATtiny25 coloca a variável CURRFLAG em nível alto fazendo assim com que a corrente no motor seja limitada através da limitação dos sinais de PWM.

Os sinais que controlam a ponte de MOSFETs são PWM-L, PWM-R, CW/CCW-A e CW/CCW-B e quando combinados determinam a potencia e sentido da rotação dos motores. Sendo assim cada motor é controlado por dois sinais e um andar em ponte H completa constituída por dois circuitos integrados de meia ponte do tipo IR2184 e 4 MOSFETs IRF4105.

O circuito da ponte é alimentado pelos 24V provenientes das baterias e os excitadores de meia ponte recebem a sua própria tensão de alimentação de 15V do regulador MIC2941.

O MIC2941 está também ligado ao ATtiny25 que o informa caso detete um excesso de corrente, o que desliga a alimentação dos motores.

Todos os restantes componentes estão alimentados a 5V pelo regulador 7805.

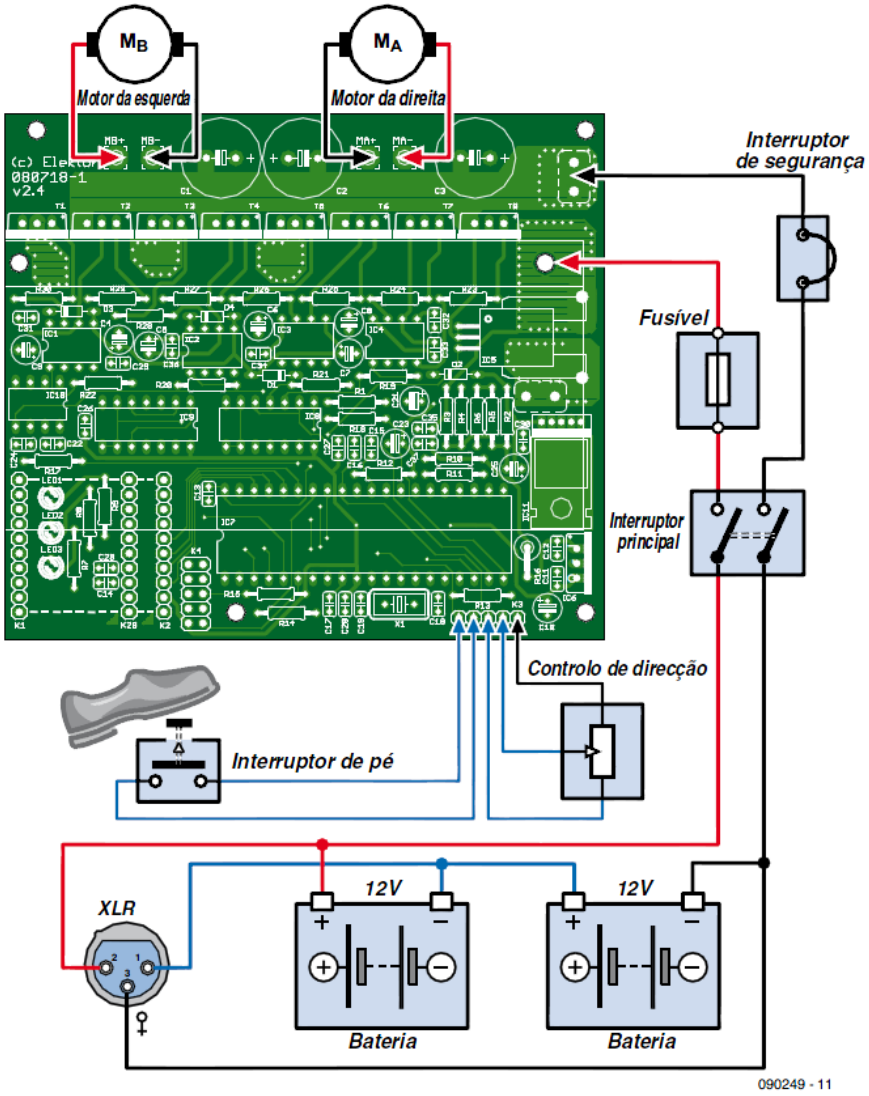


Figura 4.5 - Esquema de ligação de todas as partes constituintes do projeto

A Fig. 4.5 indica a forma como os componentes externos à placa principal são ligados como as baterias de alimentação, o interruptor de pé, o

potenciómetro da direção, o interruptor principal, o fusível de proteção e o interruptor de segurança.

Figura 4.6 - Placa com todo o hardware montado

Na Fig.4.6 é mostrado o aspeto na placa de controlo que se encontra montada na parte inferior da base com todos os componentes montados.

4.3 - Descrição dos Módulos

Esta secção diz respeito à descrição das características técnicas dos sensores, atuadores, e processadores associados ao diagrama de blocos de funcionamento do sistema apresentado no bloco anterior.

4.3.1 - Microcontrolador ATMega32

Este bloco é responsável pelo controlo e gestão de todas as entradas e saídas do sistema sendo constituído pelo microcontrolador principal.

ATMega32

O ATMega32 é o microcontrolador utilizado neste projeto para a leitura do sensor de aceleração, do giroscópio e do potenciómetro e processamento desses dados para posterior atuação dos motores, sendo por isso muito útil o facto de já ter conversores A/D incorporados, sendo as suas principais características são:

- Micro controlador de 8 bits AVR® de capacidade elevada, baixa potência
- Arquitetura avançada RISC
- 131 instruções poderosas - a maioria de execução num impulso de relógio
- 32 x 8 registos de uso geral de funcionamento
- Velocidades de relógio até 16 MHz
- Memória não volátil de programa e de dados
- 1024 bytes EEPROM
- 2K byte SRAM interno
- Programação do flash, do EEPROM

Características periféricas:

- Dois temporizadores/contadores de 8 bits com Prescalers separado e modo de comparação

- Temporizador/contador de 16 bits com Prescaler separado, modo de comparação, e modo de captura
- Contador tempo real com oscilador separado
- Quatro canais de PWM
- 8 canais, 10 bit ADC
- 8 canais Single-ended
- 7 canais diferenciais
- 2 canais diferenciais com ganho programável em 1x, em 10x, ou em 200x
- USART série programável
- Comparador do analógico
- Oscilador interno RC
- Fontes da interrupção externa e interna

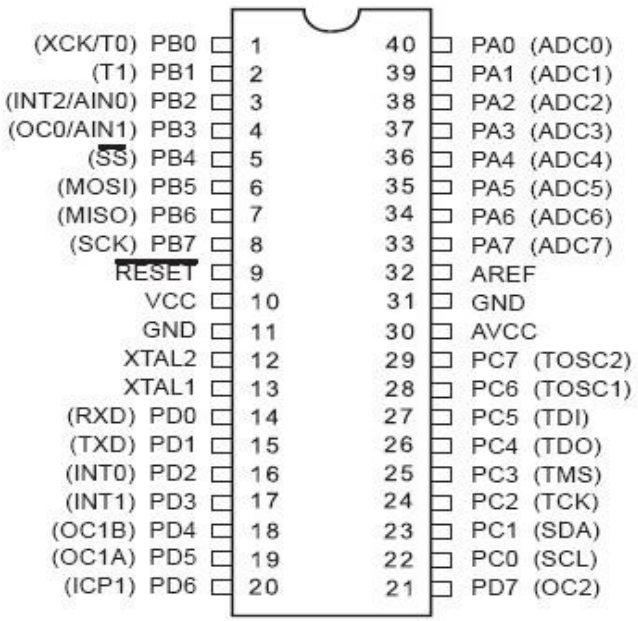


Figura 4.7 – Esquema de pinos do ATmega32

4.3.2 - Módulo de perceção

Este bloco será responsável pela aquisição do posicionamento e velocidade angular do Wheelie, constituído pelo acelerómetro ADXL325 que deteta a inclinação e pelo giroscópio IDG500 que deteta a velocidade angular.

Acelerómetro ADXL335

O ADXL335 é um acelerómetro de estado sólido de baixo custo e de baixo consumo, que como o próprio nome indica mede a aceleração do corpo baseando-se no princípio no qual um corpo por ação da inércia tem tendência a conservar a sua velocidade deslocando-se em relação ao eixo, sendo essa deslocação a magnitude da aceleração. Possui três saídas analógicas em tensão que variam conforme a aceleração seja no eixo dos X, Y ou Z. A aceleração é medida numa escala até ±5G (típico). Pode igualmente medir a aceleração dinâmica (vibração) e a aceleração de estática (gravidade).

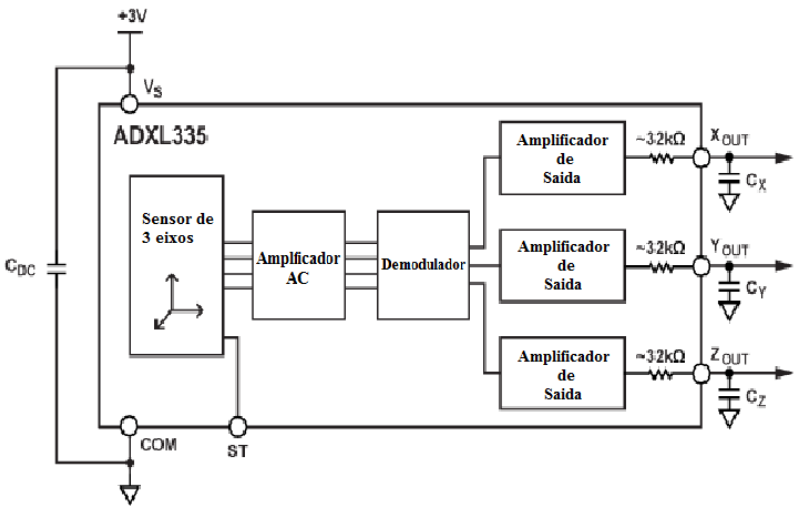


Figura 4.8 – Esquema de blocos funcionais do acelerómetro ADXL335

Na figura anterior podemos verificar o esquema de blocos funcionais do ADXL335 que é constituído por um amplificador AC, que amplifica os sinais X, Y e Z provenientes do sensor e de seguida são retificados para se obter um sinal mais preciso, no final os sinais são de novo amplificados por amplificadores independentes para cada canal X, Y e Z ficando o sinal analógico disponível nas saídas X_{out}, Yout e Zout.

IDG-500

O IDG 500 é um sensor de deslocação angular, giroscópio, de dois eixos (X e Y). Para isso utiliza dois sensores independentes cujo funcionamento é baseado na teoria da vibração do silicone[1]. No qual um desses sensores deteta a rotação em torno do eixo dos X e o outro deteta a rotação em torno do eixo dos Y. Quando um destes sensores roda em torno do eixo X ou Y origina uma vibração provocada pelo "*efeito de Coriolis*[2]" que é detetada por um detetor capacitivo, o resultado final é amplificado e filtrado para produzir uma saída em tensão analógica proporcional ao ângulo de inclinação.

1 http://invensense.com/mems/gyro/documents/whitepapers/MEMSGyroComp.pdf

2 O efeito Coriolis é resultante de uma força inercial que se aplica a corpos que se movimentam sobre uma superfície em rotação, como a Terra. A força inercial atua à esquerda da direção do movimento para rotação em sentido horário e à direita para rotação em sentido anti-horário. O efeito Coriolis causa uma deflexão aparente na trajetória de um objeto que se move em linha reta num sistema de coordenadas em rotação. O objeto na verdade não se desvia da linha reta, mas aparenta fazê-lo em função do movimento da superfície sob ele.

Todos estes circuitos eletrónicos estão integrados num único módulo para a sua rápida aplicação.

Sendo as suas principais características:

- Tensão de alimentação de 3V
- Escala completa de +/- 67 graus/segundo

- Sensibilidade de 15mV graus/segundo
- Auto calibração
- Sensor de temperatura integrado
- Resistentes a choques até 10000 g

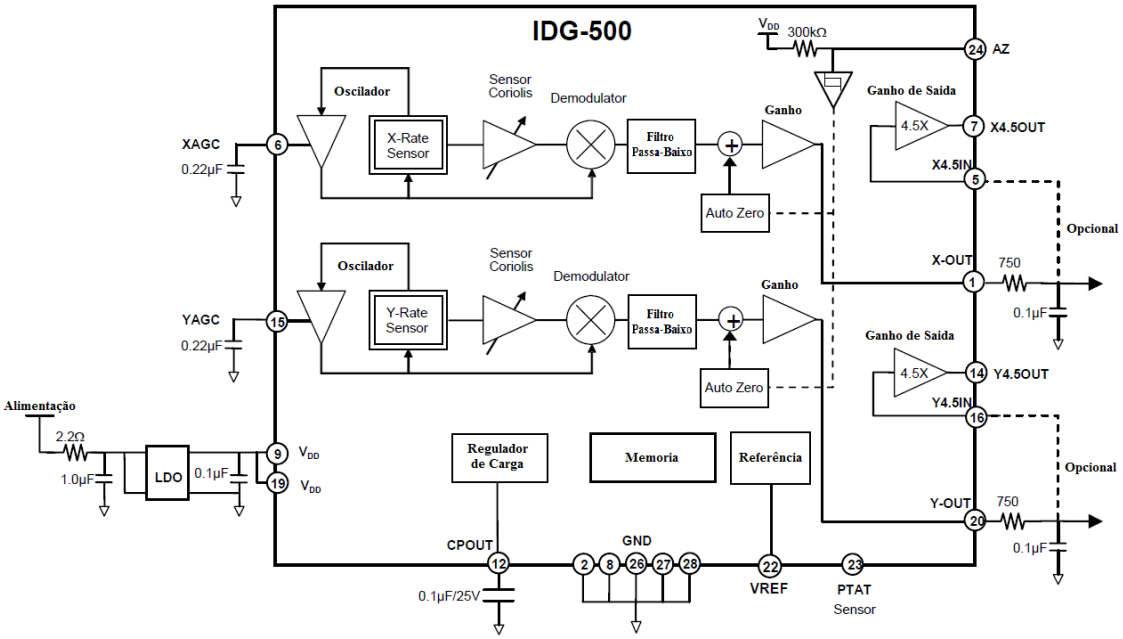


Figura 4.10 – Esquema de blocos funcionais de IDG500

4.3.3 - Modulo dos reguladores de tensão

Este bloco é o responsável pela alimentação, regulação da tensão e proteção de todos os componentes, sendo constituída pelo microcontrolador ATtiny25 que monitoriza o excesso de corrente cujo valor é fornecido pelo ACS755-SCB-100. O MIC2941 é um regulador de tensão de 15V do circuito de alimentação das pontes H.

ATtiny25

O ATtiny25 é um microcontrolador da Atmel e é utilizado neste projeto para o controlo de corrente para evitar sobrecargas nos circuitos de controlo e nos motores, sendo as suas principais características:

- 2KB Flash programável,
- 128 Bytes de EEPROM programável,
- 128 Bytes de SRAM interna,
- Dois Temporizadores/contadores de 8 bits com PWM e prescaler,
- 10-bit ADC,
- USI-Universal Serial Interface
- Debug WIRE for on-chip-debug.
- Frequência de operação até 20 MHz.

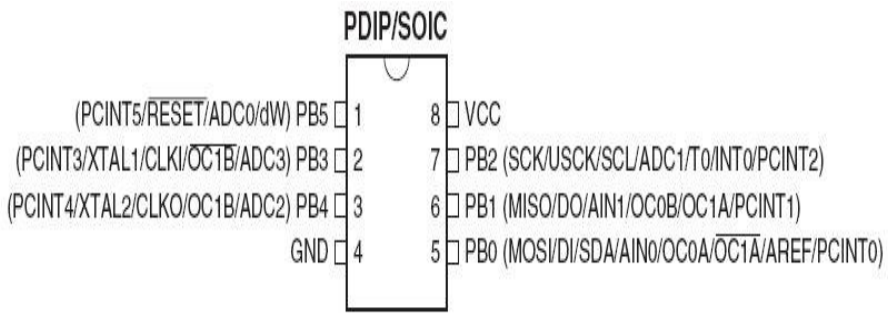


Figura 4.11 – Esquema de pinos do ATtiny25

ACS755-SCB-100

Este dispositivo consiste num circuito que atua como um sensor de corrente por efeito de HALL[3] e para isso possui no seu interior um circuito condutor em cobre que, quando percorrido por uma determinada corrente, gera um campo magnético que é convertido numa tensão proporcional à corrente que o percorre.

A amostragem de tensão de saída é então obtida quando a corrente flui do terminal 4 para o terminal 5 ou seja entre IP+ e IP- (ver Fig.4.12)
Para que as medidas sejam as mais exatas possíveis e com o menor número de perdas a resistência interna do condutor de cobre é muito baixa, cerca de 100μΩ.

3

O **efeito Hall** refere-se à diferença de potencial (potencial **de Hall**) nos lados opostos de uma fina folha de material condutor ou semicondutor na forma de uma 'barra Hall' (ou um *elemento de van der Pauw*) através da qual uma corrente elétrica flui, criada por um campo magnético aplicado perpendicularmente ao elemento Hall. A razão da tensão média pela intensidade de corrente é conhecida como *resistência Hall*, e é característica do material. O físico Edwin Herbert Hall descobriu esse efeito em 1879

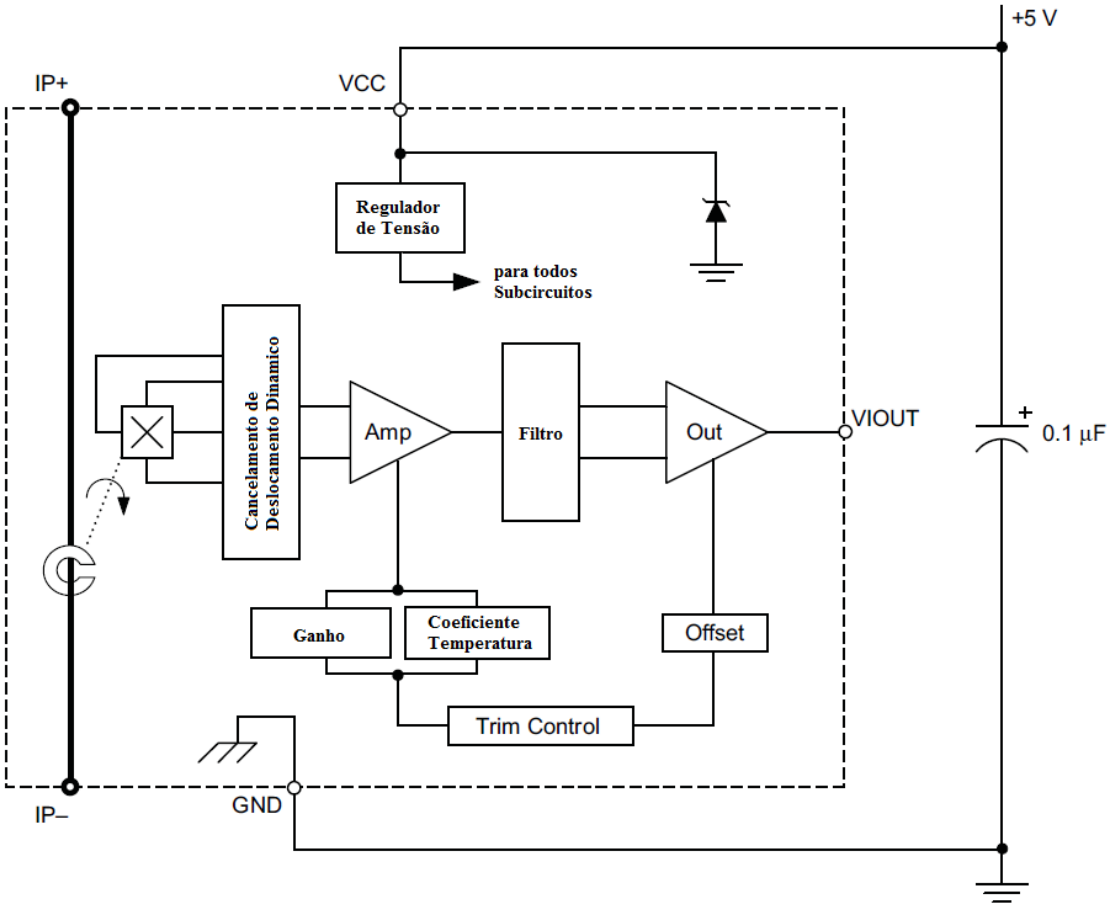


Figura 4.12 – Esquema interno do ACS755-SCB-100

MIC2941

O MIC2941 é um regulador de tensão que se encontra disponível em configuração de tensões fixas (3.3V, 5V, e 12V) e em configurações ajustáveis de tensão de 1,24V até 26V. Os dispositivos MIC2940Axx são três reguladores de 3 pinos de tensão fixa. Mas o MIC2941A usado neste projeto possui uma entrada lógica de *shutdown* programada, que permite que o regulador seja ligado ou desligado através de um sinal proveniente do ATtiny25 caso seja detetado um excesso de corrente.

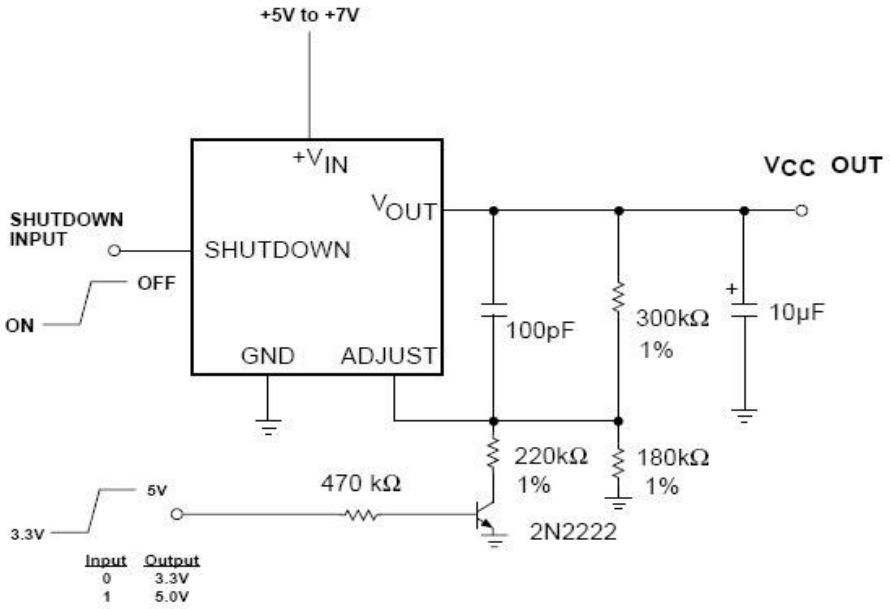


Figura 4.13 – Esquema de pinos e ligações do regulador de tensão MIC2941

4.3.4 - Módulo do PWM

Este bloco é responsável pela ligação entre o sinal de PWM proveniente do microcontrolador e os motores, sendo os seus constituintes o IR2184 que

corresponde a meia ponte H e sendo necessários dois destes componentes para criar a ponte H de cada motor.

IR2184

Este componente é um dos constituintes da ponte H que alimenta os motores sendo que cada componente representa meia ponte que suporta uma carga máxima de 600V que consiste num MOSFET de alta tensão e de um IGBT alta velocidade com as saídas “high” e “low” dependentes. As entradas lógicas são compatíveis com CMOS ou LSTTL, ativadas até uma tensão mínima de 3.3V.

Devido as altas correntes que percorrem o MOSFET IRF1405 é aconselhável a colocação de dissipadores de calor para evitar sobreaquecimentos e a consequentemente a destruição do componente.

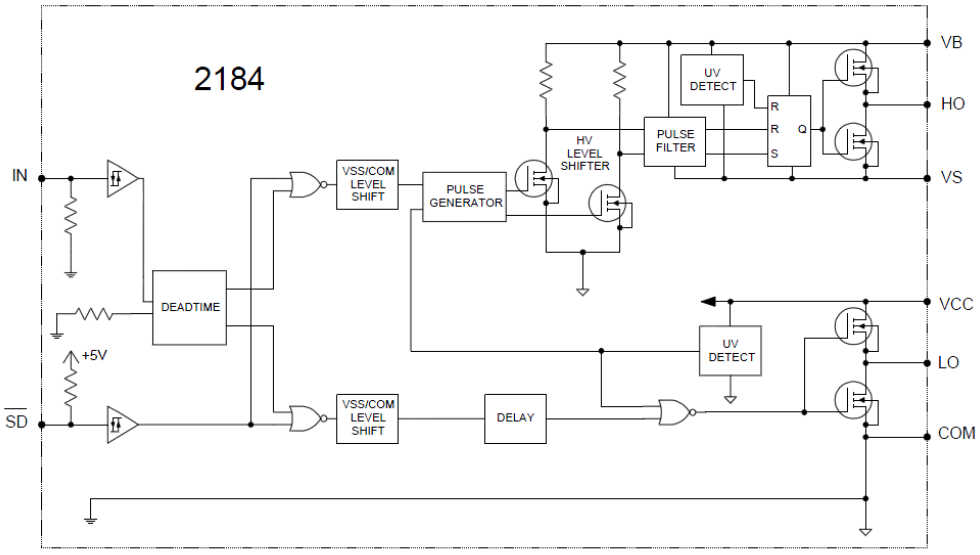


Figura 4.13 – Esquema interno do IR2184

4.4 - Descrição de funcionamento do circuito dos sensores magnéticos

4.4.1 - Placa principal (Microcontrolador ATMega8)

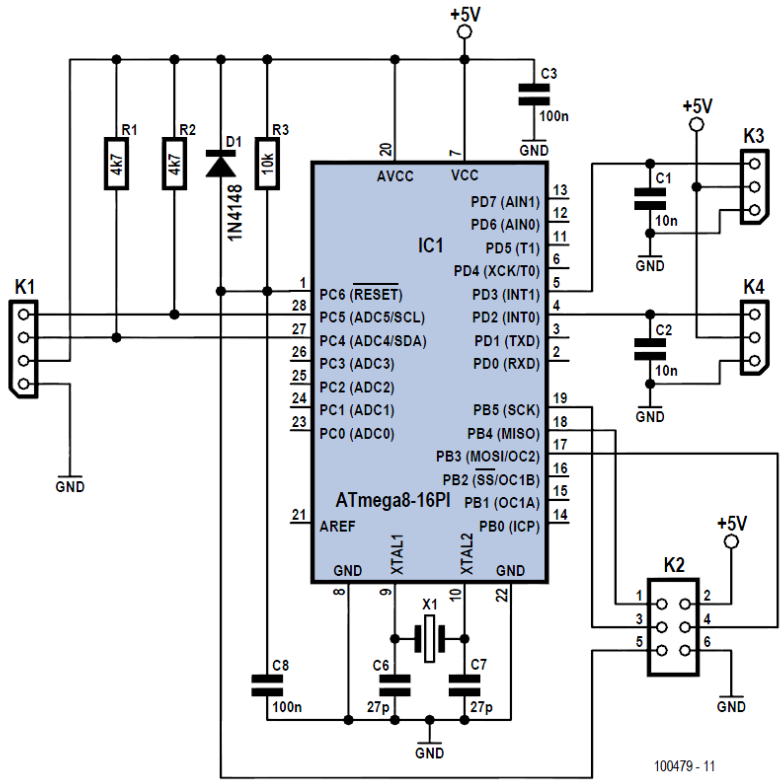


Figura 4.15 – Esquema do circuito dos sensores

O ATMega8 tem por função controlar a velocidades dos motores, ou seja assim que detetar uma diferença de rotação entre os dois motores, analisa e repõem a velocidade o motor em causa para não haver desvios de direção (ver Fig.4.15). O microcontrolador processa os impulsos fornecidos pelos sensores, ligados a K3 e K4, que provem dos sensores. O conector K1 permite ligar o ATMega8 ao ATMega32 (pinos 22 e 23) da placa principal do "Wheelie", através de uma ligação I2C (protocolo ATmega). A

velocidade pode ser calculada simplesmente contando o número de pulsos num dado intervalo de tempo, todavia, este método não garante uma precisão suficiente quando a velocidade é mais baixa. A alternativa consiste em medir o tempo entre dois pulsos, ou seja, o intervalo de tempo correspondente à passagem de dois dentes consecutivos em frente ao sensor. Utilizando um temporizador/contador a correr continuamente, o programa consegue medir este período, tendo sempre em conta quando é que o temporizador/contador passa por zero. A placa de desenvolvimento é configurada como escravo I2C e responde à placa principal do "Wheelie" com os valores relevantes, correspondentes à velocidade de cada roda. O ATMega32 calcula a velocidade, em km/h, de cada uma das rodas, e soma os dois valores. O valor obtido é utilizado para definir o nível de sensibilidade do sistema de controlo.

4.4.2 - Placa do sensor magnético

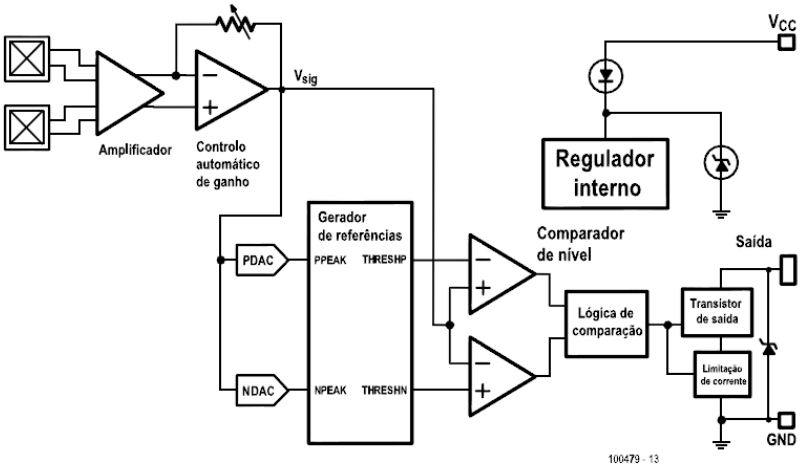


Figura 4.16 – Esquema elétrico do sensor.

Quando o conjunto se encontra próximo de uma engrenagem a passagem de cada um dos dentes vai provocar uma alteração no campo magnético. Esta alteração é medida pelo sensor de efeito de Hall que vai apresentar na sua saída uma onda dente de serra. O detetor ATS667 da Allegro inclui um circuito de acondicionamento de sinal constituído por um amplificador e um comparador (ver Fig.4.16), fornecendo um sinal na saída mais fácil de utilizar. O "Wheelie" utiliza um sistema de excitação direta dos motores, com a última engrenagem solidária com a roda. Esta configuração facilita a utilização de um sensor nas engrenagens. Os sensores de rotação sem contacto são muito fiáveis e não sofrem qualquer tipo de desgaste mecânico. Alem disso, podem ser completamente isolados do ambiente que os rodeia. Foi desenhada uma pequena placa de circuito impresso (ver Fig.4.5) para a montagem do sensor juntamente com outros três componentes. Esta placa simplifica bastante a montagem do sensor próximo das engrenagens.

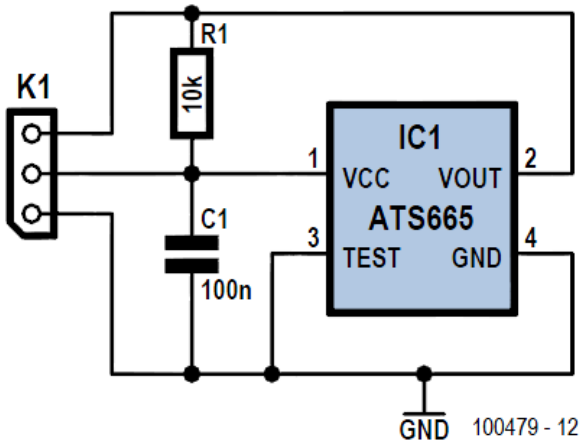


Figura 4.17 – Esquema elétrico da placa do sensor.

4.5 - Conclusões

Neste capítulo foi explicado de uma forma mais pormenorizada as características dos principais componentes eletrónicos que constituem o Wheelie.

A explicação de alguns dos componentes foi baseada nos “datashets” que se encontram os seus link’s abaixo referenciados:

http://invensense.com/mems/gyro/documents/whitepapers/MEMSGyroComp.pdf

http://www.sparkfun.com/products/11072

http://www.sparkfun.com/datasheets/Components/SMD/Datasheet_IDG500.pdf

http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf

Capítulo 5

Montagem e desenvolvimento

5.1 - Introdução

O presente capítulo será dedicado à apresentação do material essencial para a elaboração da dissertação. Na construção do "Wheelie" houve material que foi adquirido e material que foi desenvolvido, como tal, a sua descrição será realizada em subcapítulos distintos. No subcapítulo material adquirido, será demonstrado o material comprado e apresentada a análise tida em conta durante a escolha de cada componente. No subcapítulo material desenvolvido, será apresentado o material que foi desenvolvido e evidenciados os processos de construção levados a cabo na sua elaboração. Por fim, será mostrada uma tabela com os detalhes de todo o material comprado, indicando o custo de cada componente, bem como o custo total em material para o projeto.

5.2 - Material Adquirido

A estratégia escolhida para o desenvolvimento deste projeto, foi adquirido um KIT de desenvolvimento à fun-components.com. Optou-se pela compra deste KIT visto o custo do mesmo ser relativamente barato e muito mais barato se fora comprado todo o material em separado, para a construção do "Wheelie".

KIT “Wheelie”

O KIT “Wheelie” foi adquirido à fun-components.com.

Figura 5.1 – KIT “Wheelie”

Composição do KIT:

- Caixa para suporte de todo mecanismo e componentes do “Wheelie” (2 peças)
- Motores 24V 500W com rotor de acoplamento das rodas (2 peças)
- Baterias 12V 12Ah (2 peças)
- Braço de condução e direcionamento do “Wheelie” (2 peças)
- Eixo de acoplamento do braço de condução ao potenciómetro

- Rodas com pneus de câmara-de-ar (2 peças)
- Controlador do “Wheelie”
- Potenciómetro (para direção do “Wheelie”)
- Interruptor geral
- Interruptor de pressão (detetor de presença de pessoa)
- Parafuso e anilhas diversas para fixação dos diversos componentes

5.3 - Material Desenvolvido

Para além do material comprado foi necessário também desenvolver algum hardware, como Placa de circuito dos sensores magnéticos e circuito de ultrassons (para deteção de obstáculos)

5.3.1 - Circuito dos Sensores magnéticos

O sistema de controlo do “Wheelie” baseava-se apenas num acelerómetro e num giroscópio. E por isso o sistema não recebia qualquer informação sobre a velocidade de rotação de cada uma das duas rodas, sendo por isso incapaz de detetar e consequentemente compensar qualquer diferença de tração entre elas, provocando mudanças de direção indesejadas. Logo, esta limitação prejudicava o desempenho do “Wheelie” em terrenos acidentados. Para resolver este problema será acrescentando sensores de rotação a cada uma das rodas. Estes sensores vão informar o sistema de controlo sobre a velocidade de cada uma delas, permitindo que este atue de modo a ajustar cada motor, facilitando a progressão em terrenos mais difíceis.

Figura 5.2 – Circuito dos sensores

Um dos métodos para medir a velocidade de uma engrenagem é o sensor de efeito de Hall e um íman. Neste caso vamos utilizar o ATS667, e para controlar os seus sinais, que depois serão enviados para a placa de controlo do "Wheelie", um microcontrolador da Atmel, o Atmega8.

Figura 5.3 – Circuito com o sensor de Hall (ATS667)

5.3.2 - Circuito dos ultrassons

Para o circuito dos sensores de ultrassons, para deteção de obstáculos, vamos utilizar os sensores HC-SR04 e um Arduíno UNO para o processamento de dados e respetivo comando.

Figura 5.4 – sensor ultrassons HC-SR04

O sensor de ultrassons HC-SR04 funciona como um detetor de objetos e permite medir distancias mínimas de 2 centímetros podendo chegar a distancias máximas de até 5 metros com uma precisão de 3 milímetros . Estes sensores emitem um sinal ultrassónico que reflete num objeto e retorna ao sensor (eco). O sinal de retorno é captado, permitindo-se calcular a distância do objeto ao sensor tomando o tempo de voo do sinal.

A velocidade do sinal ultrassónico é de aproximadamente 340 m/s, assim, se o sensor estiver a uma distância "d" do objeto, o sinal percorrerá uma distância equivalente a "2d" para sair e retornar ao sensor. Sabendo esses conceitos é possível calcularmos a distância de um objeto pela fórmula:

$$Velocidade = \frac{Distancia}{Tempo} \Rightarrow v = \frac{2d}{t} \Rightarrow d = \frac{v \times t}{2} \Rightarrow d = 120 \times t \qquad (5.1)$$

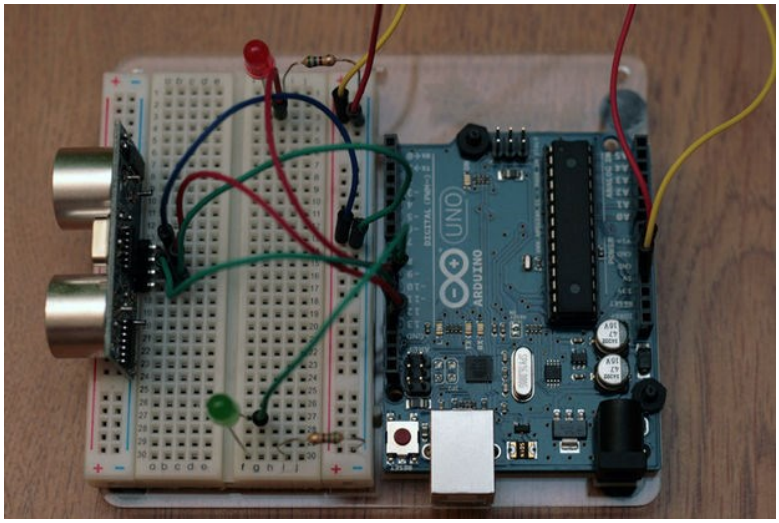


Figura 5.5 – montagem experimental de um sensor ultrassons

Como já visto, a função do sensor de ultrassons é detetar e medir a distância de objetos que passam por ele. O sensor utilizado foi o HC-SR04 que possui quatro pinos de ligação, onde os dois pinos das extremidades são os de alimentação (VCC e GND) e os do centro são os pinos em que haverá a receção (Echo) e emissão (Trig) dos pulsos de ultrassons. Conforme a programação e o esquema de montagem, o circuito funcionará da seguinte forma: enquanto não houverem objetos ao alcance (pré-programado no Arduíno), um LED verde estará permanentemente aceso, caso contrário um LED vermelho indicarão que há objetos ao alcance.

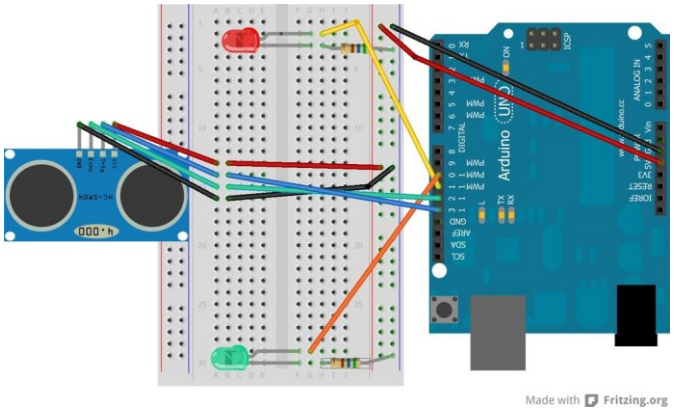


Figura 5.6 – esquemático de um sensor ultrassons

No código utilizado para testar o HC-SR04 sensor de distancia, a alimentação utilizada é VCC do Arduíno 5v e GND do Arduíno 0v, o pin 13 é utilizado para o Echo e para o Trig o pin 12, o Led Vermelho é ligado ao pin 11 e Led Verde ao pin 10, é aplicada uma resistência de 330Ω para cada um dos LEDs. O funcionamento é simples e descreve-se da seguinte forma: em primeiro lugar emite-se um sinal ultrassónico que reflete num objeto e retorna ao sensor, o sinal de retorno é captado, permitindo-se determinar a distância do objeto ao sensor tendo em conta o tempo de voo do sinal.

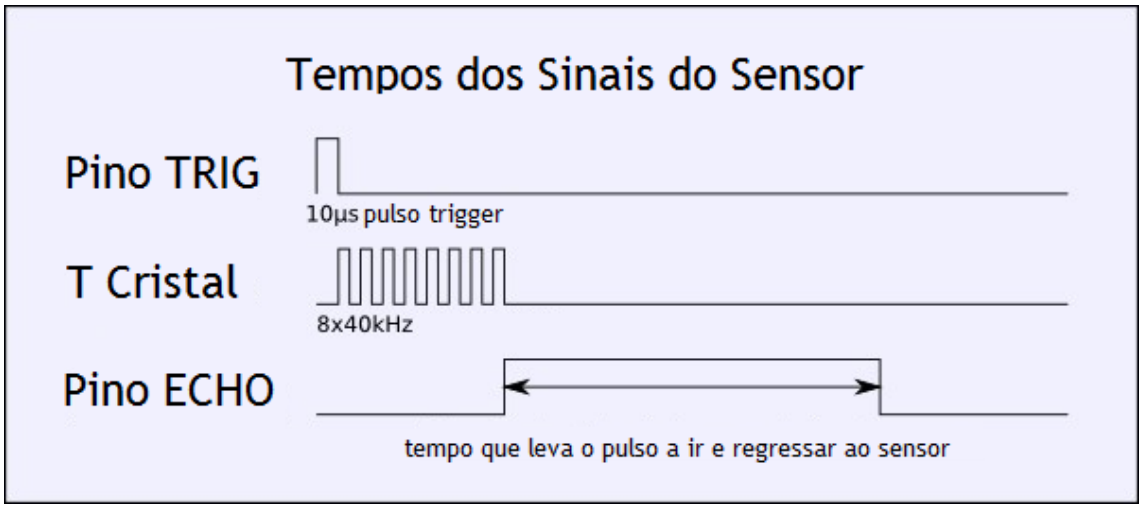


Figura 5.6 – tempos dos sinais do sensor ultrassons

Foi desenvolvido, em seguida, o programa tendo em conta que se iria utilizar dois sensores na frente, um posicionado à esquerda e outro a direita, com um ligeiro anglo para as laterais, que têm a finalidade de deteção de obstáculos, e um terceiro sensor que ficaria voltado para o chão, para verificar qualquer ausência de terreno, possíveis escadas. Pode ser visto nos anexos (anexo D).

Foi desenhada uma placa de circuito impressa para suportar o controlador dos sensores, que se pode visualizar na Fig.5.7.

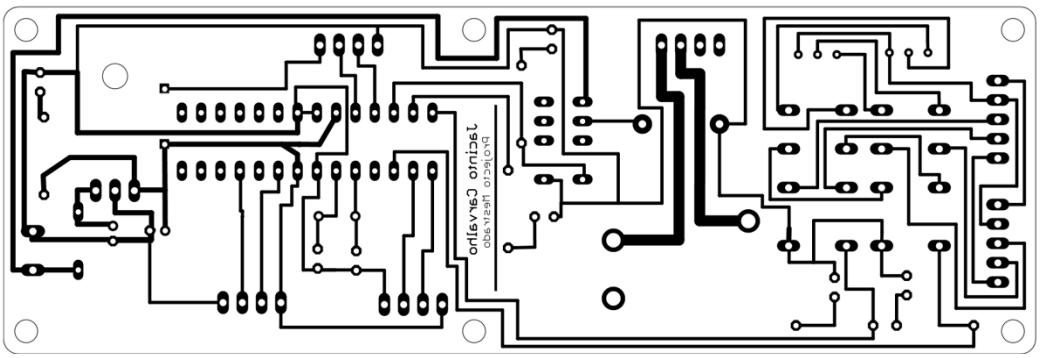


Figura 5.7 – Placa de circuito impresso para implementar os sensores de ultrassons

5.4 - .Custos de desenvolvimento da plataforma

Neste ponto foi analisado o material principal utilizado na construção do "Wheelie". Foi uma parte que consumiu um tempo considerável neste projeto. Na compra do material, teve-se o cuidado de procurar apenas os componentes que satisfizessem os requisitos mínimos pretendidos, de forma a não encarecer demasiado o projeto. Por isso chegamos à conclusão que seria melhor a comprar de um KIT. Que se encontrava à venda no mercado e que cumpriria os objetivos necessários. Houve também alguns atrasos na entrega do material pedido.

Relativamente ao material desenvolvido, como as placas de circuito impresso e a estrutura mecânica, exigiram bastante tempo de projeto e construção. As PCBs tiveram que passar por uma segunda iteração, para corrigir alguns defeitos e adicionar coisas que ficaram esquecidas, no entanto o resultado final foi bastante satisfatório.

De seguida será mostrado, numa tabela, todo o material adquirido indicando o preço de cada componente e o preço total do projeto no final. De notar que os preços aqui indicados estão ausentes dos custos de portes de envio. De referir ainda, que apesar de não constar na tabela, terem sido adquiridos os

sensores magnéticos, estes componentes não foram adquiridos, devido a problemas com a aquisição nos respetivos fornecedores. A programação necessária para os colocar a funcionar teve de ser desenvolvida e será analisada mais à frente neste documento.

Lista de material encomendado:

- CONSUMÍVEIS -	Referência Fornecedor	Quant.	Preço Unit.	Preço Linha	Fornecedores	
KIT Wheelie	Wheelie	1	970.00 €	970.00€	Fun-components	970.00€
PBB Sensores Magnéticos	#		€	€	-	
Sensor - ATS667	-	2	9.36 €	18.72 €	-	
Cristal 16.000MHz HC49S	672-0299	2	0.40 €	0.80 €	RS	
ATMega8-16PI	628-1788	1	3.40 €	3.40 €	RS	
Díodo - 1N4148	671-0286	1	0.04 €	0.04 €	RS	
PCB Sensores Ultrassons	#		€	€	-	
Sensor ultrassons HC-SR04	-	3	2.40 €	7.20 €	RS	
ATMega328	696-2260	1	2.73 €	2.73 €	RS	
Regulador - 7805	298-8508	1	0.56 €	0.56 €	RS	
Rele 30A	418-6170	1	4.03 €	4.03 €	RS	
Rele Duplo contacto 1A	369-719	2	4.19 €	8.38 €	RS	
Led vermelho 5mm	228-5067	1	0.20 €	0.20 €	RS	
Resistências variáveis 25KΩ	521-9754	2	2.10 €	4.20 €	RS	
Condensadores vários	RS	-	€	€	RS	
Resistências várias	RS	-	€	€	RS	
Socket header,20 vias , 2,54mm	360-6342	2	1.59 €	3.18 €	RS	
Socket header 40 vias, 2,54mm	782-9186	2	3.25 €	6.50 €	RS	
Conector hembra, 45 pines , 2.54mm	230-4893	2	4.65 €	9.30 €	RS	
						69.24€
TOTAL						1039.24€

Tabela 3 – Material encomendado

Capítulo 6

Software de controlo

6.1 - Descrição

O programa projetado para o microcontrolador tem como função o controlo e processamento do fluxo de dados provenientes das entradas do sistema de perceção (sensores) e posterior comando dos atuadores. Para isso o fluxo de dados proveniente dos sensores é atualizado a uma frequência de 100Hz fazendo assim com que o controlo do veículo seja suave, pois com uma atualização constante, não existe o problema de haver uma mudança rápida do estado da plataforma, ou seja dos dados dos sensores .

Após efetuadas as inicializações de todas as variáveis, configuração do Timer0 como interrupção, e configuração do Timer1 como PWM, ler os valores iniciais de calibração e ativar as interrupções, todas as rotinas são efetuadas dentro da rotina de interrupção de tempo do Timer0 que dispara em intervalos de tempo constantes. O Timer0 é um temporizador de 8 bits e dispara uma interrupção cada 16,32ms (1/16.000.000Hz * 255 * 1024 = 16,23ms), para termos a frequência de atualização dos sensores, pretendida de 100Hz, ou seja cada 10ms (1/100 = 10ms), colocamos o Timer0 = 100 (255 - 100 = 155) logo vamos ter uma interrupção cada 9,92ms que é aproximadamente os nossos 10ms pretendidos (1/16.000.000 * 155 * 1024 = 9,92ms).

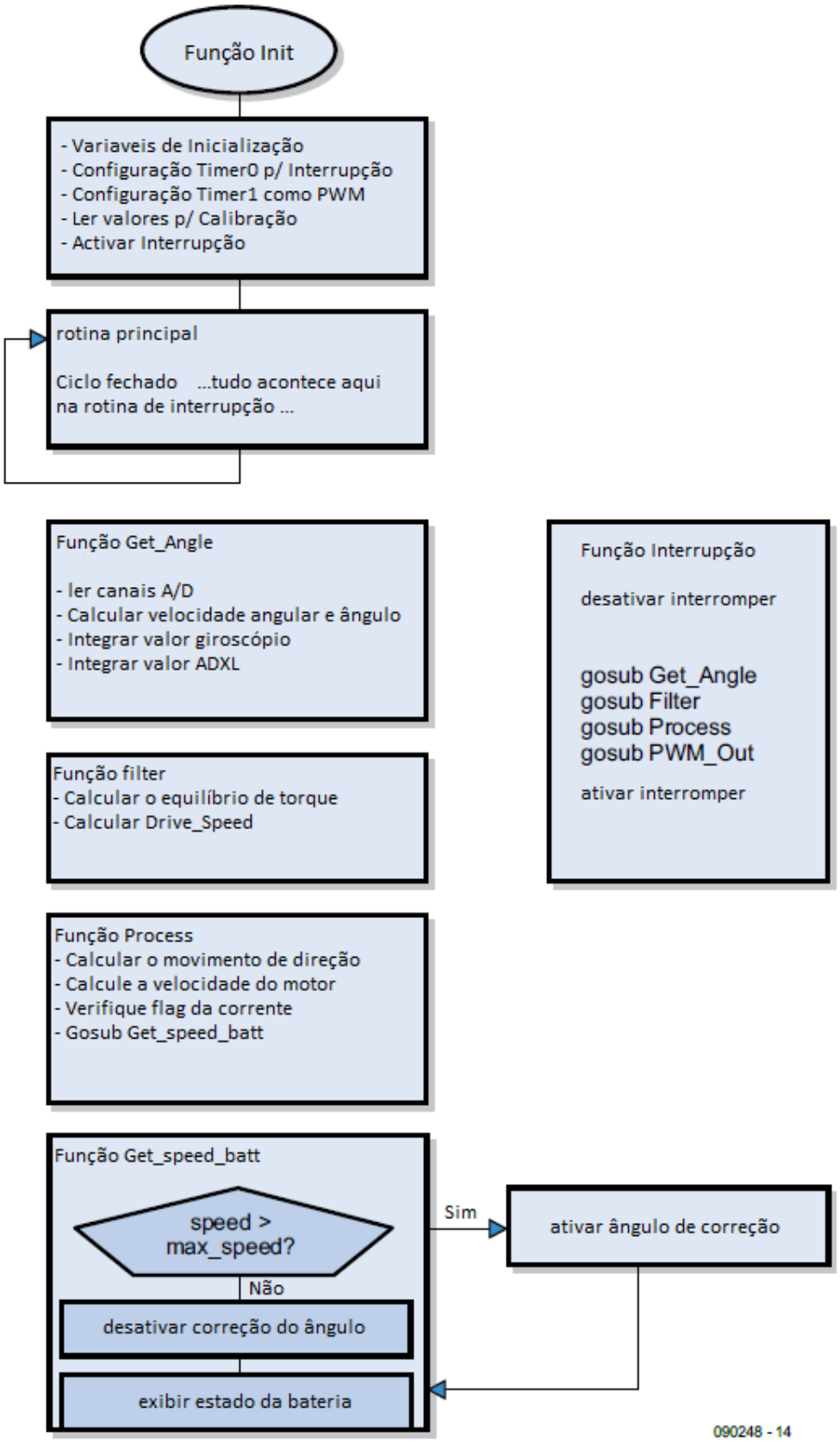


Figura 6.1 – Fluxograma do programa de controlo

Nas rotinas chamadas dentro da interrupção de tempo são lidos os valores provenientes dos vários canais dos conversores A\D. De seguida calcula-se o valor do ângulo (ver equações 3.12 e 3.13) que é necessário corrigir, integrar o valor do giroscópio e do acelerómetro. Posteriormente é calculado o valor que deve ser entregue aos módulos de potência dos motores para compensar a inclinação tendo também em conta o valor proveniente da coluna de direção para calcular a direção do movimento. Adicionalmente faz-se ainda a verificação de existência de alarme de excesso de corrente e se o interruptor de pé está pressionado. Caso a velocidade máxima seja atingida e seja necessário corrigir o ângulo da plataforma é então enviada uma potência extra para os motores (em modo normal funcionam a 70% da sua potência máxima) que "empurra" o condutor para trás e diminui a velocidade. Tendo em conta que para uma conversão A/D completa são necessários 13 ciclos de relógio ADC, e com um prescaler = 128, logo o ciclo completo leva 104µs (1/16.000.000 * 13 * 128 = 104µs).

O programa foi desenvolvido utilizando o ambiente de desenvolvimento Bascom-AVR e encontra-se transcrito no anexo A e pelo que seguidamente se explica as principais funções envolvidas, para o controlo proporcional.

Função init:

Inicializa as configurações TIMER0, TIMER1, as saídas PWM, as variáveis e calibra o giroscópio, o acelerómetro e o controlo de direção.

Função Get_Angle:

Lê os valores das entradas analógicas, integra os valores do giroscópio e da bateria durante um período de 50 amostras e posteriormente calcula a velocidade angular (Angle_Rate) e a inclinação absoluta (Tilt_angle).

Função Filter:

Calcula a alteração necessária na aceleração dos motores (Balance_Diff) e a velocidade final dos motores (Drive_Speed).

Função Process:

Utiliza a velocidade e a posição atual do volante para obter a alteração necessária na velocidade dos motores para virar o veículo no sentido pretendido, sendo que por motivos de segurança quanto maior for a velocidade menor será o ângulo de viragem. Verifica se o controlador de corrente reportou algum evento de sobrecarga de corrente nos motores e reduz a velocidade (Drive_Speed) caso necessário. A condição de sobrecarga e o alarme de estado do interruptor de pé são sinalizados pelos LEDs a piscar. A função invoca também a função Get_speed_batt.

Função Get_speed_batt:

Esta função adiciona o valor Angle_correction no caso da velocidade máxima ter sido ultrapassada e define o estado dos leds indicadores do estado da bateria

Função PWM_Out

Configura as saídas PWM de acordo com a aceleração necessária para os motores, bem como as saídas que indicam o sentido de rotação desejado. Impõem também um limite máximo para a potência de excitação (PWM_MAX)

Função interrupt:

Esta função funciona a frequência de 100Hz, invocando para isso as funções: Get_Angle, Filter, PWM_Out.

6.2 - Arquitetura de Software

Aqui serão descritos alguns pontos essenciais na arquitetura do software, como é o caso da escolha e configuração das portas do microcontrolador, como o programa é um pouco extenso e com algumas sub-rotinas, só serão descritas as configurações das portas e a sub-rotina do PWM "Sub Set_pwm"

6.2.1 - Configuração de portas e definir variáveis

Para iniciarmos, vamos primeiro definir as portas do microcontrolador e as variáveis

ATMega32		
Pino	**Utilizacão**	**Descricão**
1	Não utilizado	-
2	Não utilizado	-
3	Led 3	Erro/Estado da Bateria
4	Led 2	Estado da Bateria
5	Led1 (Verde)	Funcionamento/Estado da
6	MOSI	Ficha ISP (para
7	MISO	Ficha ISP (para
8	SCK	Ficha ISP (para
9	RST	Ficha ISP (para
10	Positivo 5V	Alimentação
11	Negativo 0V	Alimentação
12	Cristal 16MHz	Ciclo de Relógio
13	Cristal 16MHz	Ciclo de Relógio
14	Não utilizado	-
15	Não utilizado	-
16	CURRFLAG	Limite de corrente
17	Não utilizado	-
18	PWM-L	PWM do motor esquerdo
19	PWM-R	PWM do motor direito
20	CW/CCW-A	Sentido de rotação do
21	CW/CCW-B	Sentido de rotação do
22	Não utilizado	-
23	Não utilizado	-
24	Não utilizado	-
25	Não utilizado	-

26	Não utilizado	-
27	Não utilizado	-
28	Não utilizado	-
29	Não utilizado	-
30	AVCC	Alimentação
31	Negativo 0V	Alimentação
32	AREF	Tensão de referência
33	FOOTDETECT	Interruptor de pé
34	POT	Potenciómetro de direção
35	GYRO-Y	Giroscópio posição Y
36	GYRO-X	Giroscópio posição X
37	ADXL-X	Acelerómetro posição X
38	ADXL-Y	Acelerómetro posição Y
39	Negativo 0V	Ligado á massa
40	UBATT	Monitorização da tensão

Tabela 4 – Descrição dos pinos utilizados no ATMega32

Do pino 1 ao 8 PortB, do pino 14 ao 21 PortD, do pino 22 ao 29 PortC e do pino 33 ao 40 PortA. Os portos podem ser configurados como entradas ou saídas e o PortA pode ser configurado como ponto de entradas analógico, como vai ser o caso, ou seja os pinos 33 ao 40 vão ser entradas do conversor analógico/digital, ou seja:

```
"Config Porta = Input
 Config Adc = Single , Prescaler = 128 , Reference = Off
 Start Adc"
```

Os PortB vamos configurar como entrada também no entanto os pinos 2, 3 e 4,vamos configurar como as saídas para os led's, como abaixo descrito:

```
"Led3 Alias Portb.2
 Led2 Alias Portb.3
 Led1 Alias Portb.4
 Config Portb = Input
 Confif Pinb.4 = Output
 Confif Pinb.3 = Output
 Confif Pinb.2 = Output"
```

Os portC e PortD vamos configurar como saídas, no entanto o pino 16 (Portd.2) vamos configurar como entrada e configurar também as portas do PWM e do CW/CCW:

```
"Pwm_al Alias Ocr1al
  Pwm_bl Alias Ocr1bl
  Cw_ccw_a Alias Portd.6
  Cw_ccw_b Alias Portd.7

  Confif Portc = Output
  Confif Portd = Output
  Confif Pind.2 = Input"
```

De seguida vamos configurar o Timer0 e definir todas as variáveis e atribuir valores às constantes:

```
"Config Timer0 = Timer , Prescale = 1024
  On Timer0 Tinter

  Dim Buzzer As Byte
  Dim Ad_adxl As Long
  Dim Ad_gyro As Long
  Dim Ad_batt As Integer
  Dim Ad_swi As Integer
  Dim Total_adxl_gyro As Long
  Dim Average_gyro As Long
  Dim Average_batt As Long
  Dim Drivea As Integer
  Dim Driveb As Integer
  Dim Buf1 As Long
  Dim Buf As Long
  Dim Buf2 As Long
  Dim Buf5 As Integer
  Dim Tilt_angle As Long
  Dim Drive_a As Integer
  Dim Drive_b As Integer
  Dim Tcount As Integer
  Dim Drivespeed As Long
  Dim Steeringsignal As Long
  Dim Anglecorrection As Long
  Dim Angle_rate As Long
```

```
Dim Balance_moment As Long
Dim Mmode As Byte
Dim Drive_sum As Long
Dim Ad_rocker As Integer
Dim Blinkcount As Byte
Dim Timeout As Long
Dim Delta As Byte
Dim Buf3 As Long
Dim Rockersq As Long
Dim Rocker_zero As Integer
Dim Adxl_zero As Word
Dim Gyro_zero As Word
Dim Loopct As Integer
Dim Limitcurr As Integer
Dim Errno As Byte
Dim Adxl_offset As Integer
Const First_adc_input = 0
Const Last_adc_input = 6
Const Adc_vref_type = $0x40
Const Mdrivesumlimit = 20000
Const Total_looptime = 500
Const Total_looptime10 = 50
Const _run = 1
Const _standby = 0
Const _warmup = 2
Const _down = 3
Const Safespeed = 5000
Const Msafespeed = -5000
Const Sw_down = 50
Const Critical = 80
Const Max_pwm = 180
Const Mmax_pwm = -180
Const Battdead = 487000
Const Battok = 505000”
```

6.2.2 - Descrição do PWM

		Sentido de rotação Horário	Sentido de rotação Anti-horário	Parado	Parado
IC1 ou IC3 (IR2184)	LO	X		X	
	HO		X		X
IC2 ou IC4 (IR2184)	LO		X	X	
	HO	X			X

Tabela 5 – Descrição do Funcionamento do controlo da ponte em H

Na sub-rotina Set_pwm são impostos os limites para a "Drive_a" e "Drive_b" que corresponde ao motor direito e motor esquerdo, respetivamente. Estes limites máximos são "Max_pwm", que é uma constante com valor definido de 180 e outro "Mmax_pwm" que é uma constante com valor definido de -180. Define também o sentido de rotação dos motores através da instrução que verifica se os valores "Drive_a" e "Drive_b" são maiores ou menores que 0 (zero), colocando "Cw_ccw_a" e "Cw_ccw_b" a 0 (zero) ou a 1 (um) se os valores de "Drive_a" e "Drive_b" forem menores que 0 (zero) ou maiores respetivamente, ou seja se um dos valores de "Drive_a" ou "Drive_b" forem menores que 0 (zero), coloca a 1 (um) "Cw_ccw_a" ou "Cw_ccw_b", respetivamente, caso contrário (se for maior que 0) coloca "Cw_ccw_a" ou "Cw_ccw_b" a 0 (zero). Faz também a inversão do sinal do PWMB, de modo a que o "wheelie " se desloque em linha reta, caso contrário rodopiava. Como se pode ver:

	Sentido de rotação Anti-horário	Parado	Sentido de rotação Horário	Parado
PWM_L \ R	~	1	~	1
CC/CCW_A \B	0	1	0	1

Tabela 6 – Descrição da rotação dos motores em relação aos sinais Enviados do Microcontrolador

```
Sub Set_pwm

  'Limitar PWM'

  If Drive_a > Max_pwm Then
    Drive_a = Max_pwm
  End If
  If Drive_a < Mmax_pwm Then
     Drive_a = Mmax_pwm
  End If

  'definir o bit de direção'

  If Drive_a < 0 Then
     Drivea = Drive_a * -1
     Cw_ccw_a = 1
  End If
  If Drive_a >= 0 Then
      Drivea = Drive_a
      Cw_ccw_a = 0
  End If

  'Inverter sinal no PWMB

  Pwm_al = 255 - Drivea

  'Limitar PWM'
  If Drive_b > Max_pwm Then
     Drive_b = Max_pwm
  End If
  If Drive_b < Mmax_pwm Then
     Drive_b = Mmax_pwm
  End If
```

```
    'definir o bit de direção'

    If Drive_b < 0 Then
       Driveb = Drive_b * -1
       Cw_ccw_b = 1
    End If
    If Drive_b >= 0 Then
       Driveb = Drive_b
       Cw_ccw_b = 0
    End If

    Pwm_bl = Driveb
    Return
End Sub
```

As variáveis “Pwm_al” e “Pwm_bl”, é que vão ser os valores finais de saída dos pinos 19 e 18 respetivamente, já que os valores de “Drive_a” e “Drive_b” são calculados na sub-rotina Process:

```
'definir a velocidade e direção
 Drive_a = Drivespeed - Steeringsignal
 Drive_b = Drivespeed + Steeringsignal
```

O valor “Steeringsignal” vai ser o valor de “Rockersq” que é um valor calculado que vai depender da variável de velocidade “Drive_sum” em função da variável de direção “Rockersq”. Já o valor “Drivespeed” depende da variável de velocidade “Drive_sum” em função da variável “Balance_moment” que depende das variáveis “Tilt_angle” e “Angle_rate”. As variáveis “Tilt_angle” e “Angle_rate” dependem dos valores adquiridos pelo acelerómetro e do giroscópio.

6.3 - Conclusões

Neste capítulo pretendeu-se dar uma breve explicação de como foram escolhidas as portas do ATMega32, em função dos dados que temos para processar e do que pretendemos vir a obter, e também dar uma pequena e breve descrição de como funciona o software desenvolvido, mais concretamente da escolha de tipo de portas, em função dos dados adquiridos e do que pretendemos e o tipo de variáveis e contantes envolvida, e no final uma também breve descrição de como funciona o software para o PWM.

Capítulo 7

Testes Finais

7.1 - Testes finais

A fim de verificar o comportamento da plataforma auto-balanceada, com e sem os desenvolvimentos introduzidos, foram realizados uma serie de testes com o intuito de se verificar a funcionalidade dos mesmos.
Na serie de testes realizados, cada um dos quais foram feitos para testar diferentes situações que envolvem diferentes sensores, a fim de se verificar o comportamento dos mesmo e se a introdução dos diferentes sensores era beneficia ou não.

7.2 - Teste 1 – Andar em linha reta

O primeiro teste, é simples, andar em linha reta, e tem como finalidade verificar o auto-balanceamento da plataforma, assim como a adaptação do individuo à plataforma.
Como qualquer veiculo, existe sempre um período de adaptação e aprendizagem de funcionamento, destes, e este não é diferente também requer um período de aprendizagem e adaptação ao mesmo, que varia de individuo para individuo, o qual só depende da capacidade do individuo se adaptar a novas formas de locomoção.
Conclui-se logo de inicio, que quanto mais tempo o individuo tinha a andar na plataforma, maior era a sua facilidade de manobrar a plataforma. Ponto essencial para se fazer os teste com alguma precisão, foi necessário um tempo de adaptação para que os indivíduos tivessem mais ou menos a mesma destreza no manobrar da plataforma.

Figura 7.1 – Teste de controlo

7.3 - Teste 2 – Fazer um percurso

Neste teste é traçado um percurso, pelo qual o individuo tem de o percorrer, com este teste pretende-se verificar qual a agilidade da plataforma, de referir que neste caso também a influencia do individuo também é importante. pois por vezes a agilidade da plataforma esta dependente da agilidade do individuo. O percurso traçado foi um percurso simples, que se baseava a deslocar a plataforma de uma sala a outra, pelo corredor.

Figura 7.2 – Teste de percurso

7.4 - Teste 3 – Fazer percurso com obstáculos

A diferença deste teste em relação ao anterior foi a introdução de obstáculos, e com estes pretende-se verificar a reação e comportamento da plataforma para evitar obstáculos e a maior ou menor dificuldade de os contornar. Aqui a agilidade do individuo com a plataforma é que vez a diferença, pois a plataforma revelou boa agilidade em manobras apertadas e com pouco espaço para contornar os obstáculos.

Figura 7.3 – Teste de obstáculos

7.5 - Teste 4 - Fazer percurso de obstáculos com os sensores

Este teste é idêntico ao anterior só que aqui é adicionado à plataforma o sistema de sensores de ultrassons, a fim de se verificar o comportamento do sistema dos sensores de ultrassons e as vantagens da introdução do mesmo sistema. Verificou-se que a introdução dos sensores resultou muito bem, pois funcionou e cumpri-o o seu objetivo que era evitar os obstáculos mesmo que o individuo não estive-se atento aos mesmos.

7.6 - Conclusão finais dos testes

Podemos concluir que todos os testes foram concluídos com êxito, e que os resultados foram mais ou menos os esperados. A introdução, que dos sensores magnéticos quer dos sensores de ultrassons foram uma mais valia para locomoção da plataforma, no entanto os sensores magnéticos o seu desempenho não é tão visível quanto os sensores de ultrassons. Nos sensores de ultrassons o desempenho é notório, e é de logo visível o desempenho dos mesmos para evitar os obstáculos.

Capítulo 8

Conclusões e Futuros Desenvolvimentos

8.1 - Conclusões gerais

A montagem não ofereceu muita dificuldade tratando-se de um KIT, no entanto alguns materiais de alguns componentes não se mostraram muito resistentes o que levou a alguns ajustes, mesmo o controlo teve de ser revisto, pois não funcionava corretamente.

No final, a montagem de todo o material foi concluída com sucesso bem como os testes de funcionamento da Plataforma "Wheelie".

Este trabalho apresentou a análise, modelagem de uma plataforma auto-balanceada. Com este projeto, pudemos aplicar em nossos conhecimentos de controlo, eletrónica, computação e programação. Alem dos conhecimentos específicos da área de eletrónica e computação, aplicamos também conhecimentos relativos a modelagens de sistemas físicos.

No Sistema de Controlo foi utilizado em um microcontrolador de baixo custo, isto se apresenta como uma grande vantagem, já que todas outras referencias de projetos semelhantes foram utilizando microprocessadores. Ficou clara a utilidade dos microcontroladores modernos, com suas muitas funcionalidades e seu potencial para os mais diversos projetos. Com eles, boa parte da eletrónica que antes devia ser feita com CI's dedicados, pode ser integrada nos microcontroladores utilizando-se de algumas linhas de código.

Sensores de obstáculos (ultrassons) foram incorporados no projeto, para evitar que quando a plataforma caminhe, sem que embata em qualquer obstáculo que estiver à frente.

8.2 - Futuros Desenvolvimentos

A primeira melhoria a ser feita seria dotar a plataforma auto-balanceada de movimento autónomo. Para isto, necessitaríamos da leitura do angulo de inclinação e implementarmos um sistema de controle remoto. Com base nesta mesma ideia, e considerando que nossos motores já são independentes, podemos permitir a plataforma realizar curvas, aplicando um controle diferencial no sinal de cada motor. Pensando mais adiante, poderíamos acoplar uma câmara e um sistema de radiofrequência para que se possa controlar a plataforma remotamente, ou ela mesmo tomar suas próprias decisões.

Referências Bibliográficas

[1] Segway - descrição
http://pt.wikipedia.org/wiki/Segway
Acedido em 04/05/2014.

[2] Wheelie – Projeto
Relatório de Estágio apresentada ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Licenciado em Engenharia Eletrotécnica e de Computadores - 2012
Acedido em 04/05/2014.

[3] "Segway Simply Moving". Disponível em
http://www.segway.com/individual/models/index.php
Acedido em 04/05/2014.

[4] "Segway RMP (Robotic Mobility Platform)." Disponível em
http://rmp.segway.com/
Acedido em 04/05/2014.

[5] "Anybots QB Telepresence Robot". Disponível em
https://www.anybots.com/#front
Acedido em 04/05/2014.

[6] "GM/Segway EN-V Project," 08/01/2010. Disponível em
http://www.flickr.com/photos/24641875@N07/sets/72157623526768197/
Acedido em 16/06/2014.

[7] "Segway", Mundo das Marcas. Disponível em
http://mundodasmarcas.blogspot.pt/2008/06/segway.html
Acedido em 22/06/2014.

[8] "When My Avatar Went to Work". Disponível em
http://spectrum.ieee.org/robotics/industrial-robots/when-my-avatar-went-to-work
Acedido em 22/06/2014.

[9] "Anybot's QA: The Best Humanoid Robot Yet ". Disponível em

http://www.popularmechanics.com/technology/gadgets/4298623
Acedido em 24/06/2014.

[10] "Marines Magazine". Disponível em
http://marinesmagazine.dodlive.mil/2010/10/21/rover-the-mobile-robotic-target-system/
Acedido em 24/06/2014.

[11] "Elektor Wheelie - self-balancing vehicle." Disponível em
https://www.elektor.com/projects/elektorwheelie.986808.lynkxAce
Acedido em 08/01/2014.

[12] Khalil Sultan, Ashab Mirza, "Inverted Pendulum – Analysis, Design and Implementation", July 25, 2003 Institute of Industrial Electronics Engineering, Karachi, Pakistan.

[13] Karl Johnn Astrom and Tore Hagglung, "Automatic Tuning of PID Controllers", 1988, Instrument Society of America.

[14] BULLETIN LO-COG DC "Gearmotors Series GM8000, GM9000, GM14900", Pittman Company USA.

[15] Harvey Weinberg, "Using the ADXL202 Duty Cycle Output", Application Note AN604, 2002, Analog Device USA.

[16] John Geen and David Krakauer, "New iMEMS Angular-Rate-Sensing Gyroscope", 2003, Analog Dialog Micromachined Products Division.

[17] Greg Welch and Gary Bishop "An Introduction to the Kalman Filter", 5 de abril 2004, Department of Computer Science, University of North Carolina, USA.

[18] Steven W. Smith, Ph.D., "The Scientist and Engineer's Guide to Digital Signal Processing" Chapter 22 - Audio Processing / Human Hearing, 1997-2006 by California Technical Publishing.

Anexos

Pelo facto de as folhas de especificações (datasheets) de alguns componentes ser demasiado extensas para colocar em anexo foi decidido as colocar os link's dos PDF's neste relatório.

Folhas de especificações dos vários componentes;

[10] http://invensense.com/mems/gyro/documents/whitepapers/MEMSGyroComp.pdf

[11] http://www.sparkfun.com/products/11072

[12] http://www.sparkfun.com/datasheets/Components/SMD/Datasheet_IDG500.pdf

[13] http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf

[14]http://download.siliconexpert.com/pdfs/2010/5/4/1/58/8/46/alg_/manual/0755-100.pdf

[15] http://www.atmel.com/Images/doc2503.pdf

[16] http://www.atmel.com/Images/doc2586.pdf

[17] http://www.irf.com/product-info/datasheets/data/ir2184.pdf

[18] http://www.redrok.com/MOSFET_IRF1405_55V_169A_5.3mO_Vth4.0_TO-220.pdf

[19] http://www.micrel.com/_PDF/mic2940.pdf

Anexo A

Código de programação do controlador

```
' -----------------------------------------------------------------
'
'   _  _   _  _              _
'  | | | | |  | |___ ____ __| |_____
'  | |_| |_|  |   _  | --_| --_| | | --_|
'  |_______|_|  |_|___|____|_|_|___|
'
' -----------------------------------------------------------------
'   Wheelie motor control board source code
' -----------------------------------------------------------------



$crystal = 16000000
$baud = 19200




Pwm_al Alias Ocr1al
Pwm_bl Alias Ocr1bl

Led3 Alias Portb.2
Led2 Alias Portb.3
Led1 Alias Portb.4
Cw_ccw_a Alias Portd.6
Cw_ccw_b Alias Portd.7



Config Porta = Input
Config Portb = Input
Config Pinb.4 = Output
```

```
Config Pinb.3 = Output
Config Pinb.2 = Output


Config Portc = Output
Config Portd = Output
Config Pind.2 = Input

Config Adc = Single , Prescaler = 128 , Reference = Off
Start Adc
Config Timer0 = Timer , Prescale = 1024
On Timer0 Tinter




Dim Buzzer As Byte
Dim Ad_adxl As Long
Dim Ad_gyro As Long
Dim Ad_batt As Integer
Dim Ad_swi As Integer
Dim Total_adxl_gyro As Long
Dim Average_gyro As Long
Dim Average_batt As Long
Dim Drivea As Integer
Dim Driveb As Integer
Dim Buf1 As Long
Dim Buf As Long
Dim Buf2 As Long
Dim Buf5 As Integer
Dim Tilt_angle As Long
Dim Drive_a As Integer
Dim Drive_b As Integer
Dim Tcount As Integer
Dim Drivespeed As Long

Dim Steeringsignal As Long
Dim Anglecorrection As Long
Dim Angle_rate As Long

Dim Balance_moment As Long
```

```
Dim Mmode As Byte

Dim Drive_sum As Long
Dim Ad_rocker As Integer

Dim Blinkcount As Byte
Dim Timeout As Long
Dim Delta As Byte
Dim Buf3 As Long
Dim Rockersq As Long

Dim Rocker_zero As Integer
Dim Adxl_zero As Word
Dim Gyro_zero As Word
Dim Loopct As Integer
Dim Limitcurr As Integer
Dim Errno As Byte
Dim Adxl_offset As Integer



Const First_adc_input = 0
Const Last_adc_input = 6
Const Adc_vref_type = $0x40


Const Mdrivesumlimit = 20000

Const Total_looptime = 500                    ' / / Looptime para Filtros
Const Total_looptime10 = 50                   ' / / Looptime para Filtros

Const _run = 1
Const _standby = 0
Const _warmup = 2
Const _down = 3

Const Safespeed = 5000
Const Msafespeed = -5000
Const Sw_down = 50
Const Critical = 80
```

```
Const Max_pwm = 180
Const Mmax_pwm = -180


Const Battdead = 487000
Const Battok = 505000




Declare Sub Get_tilt_angle
Declare Sub Set_pwm
Declare Sub Algo
Declare Sub Process
Declare Sub Ini
Declare Sub Getspeedlimit
Declare Sub Err_value


Pwm_al = 255
Pwm_bl = 0


'Init Variaveis'
Gosub Ini


' Dar algumas piscadelas
Set Led1
Waitms 150
Set Led2
Waitms 150
Set Led3
Waitms 250
Reset Led1
```

```
Reset Led2
Reset Led3


'-------------------------------------------------------'
'verificar interruptor da plataforma'
Ad_swi = 0
For Buf = 1 To 10
  Ad_swi = Ad_swi + Getadc(7)
  Waitms 1
Next Buf
'calcular ponto médio'
Ad_swi = Ad_swi / 10
Waitms 100

'tem de estar fechado
If Ad_swi < 100 Then
  Errno = 4
  Goto Err_value
End If


'-------------------------------------------------------'
'Obter posição de direção Rocker_zero'
Ad_rocker = 0
For Buf = 1 To 10
  Ad_rocker = Ad_rocker + Getadc(6)
  Waitms 1
Next Buf
'calculate middle'
Rocker_zero = Ad_rocker / 10


Waitms 100
'-------------------------------------------------------'
'obter gyro_zero
Ad_gyro = 0
For Buf = 1 To 10
  Buf1 = Getadc(5)
  Buf1 = 1024 - Buf1
  Ad_gyro = Ad_gyro + Buf1
  Waitms 5
```

```
Next Buf
'calcular ponto medio'
Gyro_zero = Ad_gyro / 10

Average_gyro = Total_looptime * Gyro_zero


'--------------------------------------------------------'
'Obter adxl_zero
Ad_adxl = 0
For Buf = 1 To 10
  Buf1 = Getadc(3)
  Ad_adxl = Ad_adxl + Buf1
  Waitms 5
Next Buf
'calcular ponto medio'
Adxl_zero = Ad_adxl / 10




' verificar se esta fora de valores
If Adxl_zero < 400 Then
  Errno = 3
  Goto Err_value
End If

If Adxl_zero > 600 Then
  Errno = 3
  Goto Err_value
End If

If Gyro_zero < 350 Then
  Errno = 2
  Goto Err_value
End If
If Gyro_zero > 750 Then
  Errno = 2
  Goto Err_value
End If
```

```
Enable Interrupts
Enable Timer0


'main loop
S1:

' nada para fazer, todo o trabalho é feito na interrupção ;)




Goto S1
'-------------------------------------------------------'









'-------------------------------------------------------'

Sub Set_pwm
   'Limitar PWM'
   If Drive_a > Max_pwm Then
     Drive_a = Max_pwm
   End If
   If Drive_a < Mmax_pwm Then
      Drive_a = Mmax_pwm
   End If




   'defenir o bit de direcção'
   If Drive_a < 0 Then
```

```
      Drivea = Drive_a * -1
      Cw_ccw_a = 1
    End If
    If Drive_a >= 0 Then
       Drivea = Drive_a
       Cw_ccw_a = 0
    End If
    'Inverter sinal no PWMB

    Pwm_al = 255 - Drivea


    'Limitar PWM'
    If Drive_b > Max_pwm Then
      Drive_b = Max_pwm
    End If
    If Drive_b < Mmax_pwm Then
      Drive_b = Mmax_pwm
    End If


    'defenir o bit de direcção'
    If Drive_b < 0 Then
       Driveb = Drive_b * -1
       Cw_ccw_b = 1
    End If
    If Drive_b >= 0 Then
       Driveb = Drive_b
       Cw_ccw_b = 0
    End If

    Pwm_bl = Driveb
    Return
End Sub

'-------------------------------------------------------'



'-------------------------------------------------------'

Sub Algo
    'Buf = Tilt_angle + Anglecorrection
```

```
    Buf = Tilt_angle
    Buf = Buf * 17                                  '19
    Buf1 = Angle_rate * 17                             ' 19

    'calcular balance moment
    Balance_moment = Buf1 + Buf

    'calcular drive_sum
    Drive_sum = Drive_sum + Balance_moment

    ' limitar
    If Drive_sum > 55000 Then
      Drive_sum = 55000
    End If
    If Drive_sum < -55000 Then
      Drive_sum = -55000
    End If


    'calcular drive speed'
    Buf = Drive_sum / 155
    Buf1 = Balance_moment / 165
    Drivespeed = Buf + Buf1

    Return

End Sub

'-------------------------------------------------------'


'-------------------------------------------------------'

Sub Getspeedlimit

 Buf1 = 0
 If Drive_sum > 0 Then
   Buf1 = Drive_sum - Mdrivesumlimit
 End If

 Buf1 = Buf1 / 70
 Adxl_offset = 0
```

```
If Buf1 > 13 Then Buf1 = 13
If Buf1 < 0 Then Buf1 = 0

If Buf1 > 0 Then
 Adxl_offset = Buf1
 Buzzer = 2
End If




If Buzzer = 0 Then
  'verificar tensão da bateria
  If Average_batt > Battok Then
    Set Led1
    Reset Led2
    Reset Led3
  End If
  If Average_batt < Battok Then
    Set Led2
    Reset Led3
    Reset Led1
  End If

  If Average_batt < Battdead Then
    Reset Led2
    Reset Led1
    Set Led3
  End If


 End If


Return



End Sub
'-------------------------------------------------------'
```

```
'-----------------------------------------------------'

Sub Process

 Tcount = Tcount + 1
 'fazer alguns registros de situações
 If Tcount > 10 Then
   If Buzzer = 1 Then
     Set Led1
     Set Led2
     Set Led3
   End If

   If Buzzer = 2 Then
     Set Led3
   End If

 End If
 If Tcount > 16 Then
   If Buzzer > 0 Then
     Reset Led1
     Reset Led2
     Reset Led3
     Blinkcount = Blinkcount + 1
   End If

   Tcount = 1

   'alarme reset'
   If Blinkcount > 10 Then
     Blinkcount = 0
     Buzzer = 0
   End If

 End If


 'calcular steering'
 Rockersq = Rocker_zero - Ad_rocker
```

```
' Direcção depende da velocidade'
Buf1 = Drive_sum / 20000                    '6000
If Buf1 < 0 Then
   Buf1 = Buf1 * -1
End If

If Buf1 < 1 Then Buf1 = 1


Rockersq = Rockersq / Buf1

'algumas linhas de segurança, limites de direção
'drivesum 55000  max = +- 5

Buf1 = Drive_sum / 2000
If Buf1 < 0 Then Buf1 = Buf1 * -1

If Buf1 > 22 Then Buf1 = 22


Buf1 = 27 - Buf1
Buf2 = Buf1 * -1



If Rockersq > Buf1 Then Rockersq = Buf1
If Rockersq < Buf2 Then Rockersq = Buf2


Steeringsignal = Rockersq


'situação de sub corrente?
If Pind.2 = 0 Then
  Limitcurr = 125
End If

If Pind.2 = 1 Then
 If Limitcurr < 30000 Then
   Limitcurr = Limitcurr + 1
 End If
 Buzzer = 1
End If
```

```
Drivespeed = Drivespeed * 125
Drivespeed = Drivespeed / Limitcurr


'defenir a velocidade e direcção
Drive_a = Drivespeed - Steeringsignal
Drive_b = Drivespeed + Steeringsignal


'precisa de alguns segundos para compensar o desvio temporário do giroscópio
If Mmode = _warmup Then
   Drive_a = 0
   Drive_b = 0
   Drivespeed = 0
   Anglecorrection = 0
   'Overspeed = 0
   Drive_sum = 0
   Total_adxl_gyro = 0
   Tilt_angle = 0

   If Loopct > 200 Then
     Mmode = _standby
   End If
End If

   If Mmode = _standby Then
      Drive_a = 0
      Drive_b = 0
      Drivespeed = 0
      Anglecorrection = 0
      'Overspeed = 0
      Drive_sum = 0
      Total_adxl_gyro = 0
      Tilt_angle = 0
      Buf2 = Ad_adxl - Adxl_zero
      Buf2 = Buf2 + Adxl_offset
      Timeout = 0

      'stand on platform
```

```
    If Ad_swi > Sw_down Then
        Mmode = _run
    End If

End If

If Mmode = _run Then

    'verificar interruptor da plataforma
    If Ad_swi > Sw_down Then
      'Pessoa na plataforma,
      Timeout = 0
    End If

    If Ad_swi < Sw_down Then
      'sem Pessoa na plataforma!, verificar savespeed
      If Drive_sum < 0 Then
        If Drive_sum < Msafespeed Then
          Buzzer = 1
          Timeout = Timeout + 1
        End If
      End If

      If Drive_sum > 0 Then
        If Drive_sum > Safespeed Then
          Buzzer = 1
          Timeout = Timeout + 1
        End If
      End If

      If Timeout > Critical Then
          Mmode = _down
      End If

    End If


End If



If Mmode = _down Then
```

```
    For Buf1 = 1 To 255

      If Drive_a > 0 Then
        Drive_a = Drive_a - 1
      End If

      If Drive_a < 0 Then
         Drive_a = Drive_a + 1
       End If

      If Drive_b > 0 Then
        Drive_b = Drive_b - 1
      End If

      If Drive_b < 0 Then
        Drive_b = Drive_b + 1
      End If

      Waitms 2
      Gosub Set_pwm

    Next Buf1

    Mmode = _standby



  End If



  Return
End Sub


'--------------------------------------------------------------------'


Sub Ini
' fazer algumas assembler inits
  $asm
  ldi r16,0
  Out Porta , R16
```

```
 Out Ddra , R16
 ldi r16,0
 Out Portc , R16
 ldi r16,255
 Out Ddrc , R16
 ldi r16,$34
 Out Portd , R16
 ldi r16,251
 Out Ddrd , R16
 ldi r16,$b1                          'b1=8bit ,b2=9bit,b3=10bit
 Out Tccr1a , R16
 ldi r16,$01                          ' 01 no prescaler!
 Out Tccr1b , R16
 ldi r16,0
 Out Tcnt1h , R16
 Out Tcnt1l , R16
 Out Icr1h , R16
 Out Icr1l , R16
 Out Ocr1ah , R16
 ldi r16,$ff
 Out Ocr1al , R16
 ldi r16,0
 ldi r16,0
 Out Ocr1bl , R16
 Out Assr , R16
 Out Tccr2 , R16
 Out Tcnt2 , R16
 Out Ocr2 , R16
 $end Asm


Total_adxl_gyro = 0
Average_gyro = 0
Average_batt = Battok * Total_looptime

Drivea = 0
Driveb = 0
Tilt_angle = 0
Drive_a = 0
Drive_b = 0
Drivespeed = 0
Steeringsignal = 0
Anglecorrection = 0
```

```
Angle_rate = 0
'Voltage = 0
Balance_moment = 0
'Overspeed = 0

Mmode = _standby
'Overspeed_flag = 0
Drive_sum = 0
Ad_rocker = 0
Tcount = 0
Steeringsignal = 0

Timeout = 0
Delta = 0
Loopct = 0
Limitcurr = 100
Blinkcount = 0

Adxl_offset = 0


End Sub

'-------------------------------------------------'

Tinter:
    Disable Interrupts
    Gosub Get_tilt_angle
    Gosub Getspeedlimit
    Gosub Algo
    Gosub Process
    Gosub Set_pwm
    Timer0 = 100
    Enable Interrupts
    Return
'-------------------------------------------------'


Sub Err_value
  S3:
  For Buf = 1 To Errno
    Set Led3
    Waitms 150
```

```
    Reset Led3
    Waitms 150
  Next Buf

  Wait 2
  Goto S3
End Sub

'-------------------------------------------------'



Sub Get_tilt_angle
    Ad_gyro = Getadc(5)
    Ad_gyro = 1024 - Ad_gyro

    Ad_adxl = Getadc(3)
    Ad_batt = Getadc(0)
    Ad_rocker = Getadc(6)
    Ad_swi = 1024 - Getadc(7)


    Buf = Total_adxl_gyro / Total_looptime
    Total_adxl_gyro = Total_adxl_gyro - Buf

    ' ADXL
    Buf = Ad_adxl - Adxl_zero
    Buf = Buf + Adxl_offset

    Total_adxl_gyro = Total_adxl_gyro + Buf

    ' Gyro
    Buf1 = Average_gyro / Total_looptime
    Average_gyro = Average_gyro - Buf1

    Average_gyro = Average_gyro + Ad_gyro
    Buf1 = Average_gyro / Total_looptime10

    ' calcular o Angle_rate
    Buf = Ad_gyro * 10
    Buf1 = Buf1 - Buf
    Buf1 = Buf1 * 35
    Buf1 = Buf1 / 100
```

```
    Angle_rate = Buf1

    ' calcular o Tilt_angle
    Total_adxl_gyro = Total_adxl_gyro + Angle_rate
    Tilt_angle = Total_adxl_gyro / Total_looptime10

    'calcular Average_batt
    Buf1 = Average_batt / Total_looptime
    Average_batt = Average_batt - Buf1
    Average_batt = Average_batt + Ad_batt


    Loopct = Loopct + 1


    Return
End Sub

  End
```

Anexo B

Código de programação do controlador de corrente

```
' -----------------------------------------------------------
'
'   _   _   _  _                 _
'  | | | | |  | |___ ____ ___| |_____
'  | |_| |_|  |  _  | --_| --_|  | | --_|
'  |_______|_|  |_|___|____|_|_|_|___|
'
' -----------------------------------------------------------
' -----------------------------------------------------------
' Wheelie current controller..
' -----------------------------------------------------------



$regfile = "attiny25.dat"

$crystal = 8000000

' fuses
' 0xFF, 0xDD, 0xE2

Const Cur25 = 47250
Const Curdead = 133875


Portb = 1
Config Portb = Input
Config Adc = Single , Prescaler = Auto
Start Adc
Config Portb.1 = Output
Config Portb.0 = Output
```

```
Dim Ad_curr As Long
Dim Buf As Long
Dim Av_curr As Long
Dim Cflag As Byte

Portb = 1
'desactivar vreg
Cflag = 0

'esperar um pouco
Waitms 200

'Agora alimentar o chip



'activar vreg
Portb = 0


Av_curr = 0

S1:
Ad_curr = Getadc(2)

Buf = Av_curr / 75
Av_curr = Av_curr - Buf
Av_curr = Av_curr + Ad_curr

'Curr Flag
If Cflag = 0 Then
'0.8A'
If Av_curr > Cur25 Then
 Portb = 2
 Else
 Portb = 0
End If

End If
```

```
'sub corrente, desactivar vreg
' 3.5A'
If Av_curr > Curdead Then
 Portb = 3
 Cflag = 1
End If



Goto S1



End
```

Anexo C

Código de programação do circuito dos sensores magnéticos

```
'-----------------------------------------------------------
'
'   _  _   _  _              _
'  | | | | |  | |___ ____ __| |_____
'  | |_| |_|  |  _  | --_| --_| | | --_|
'  |_______|_|  |_|___|___|_|_|___|
'
'-----------------------------------------------------------
'  Gear tooth sensor sensors
'-------------------------------------------------------------------------------------------

-----



$regfile = "m8def.dat"
$crystal = 16000000

Config Timer0 = Timer , Prescale = 1024
On Timer0 Int_timer0
Start Timer0
Config Timer1 = Timer , Prescale = 1024      'timer clock = 15625Hz =
64µs/pulse
Start Timer1
Config Timer2 = Timer , Prescale = 1024      'Interrupt after 16,32ms
On Timer2 Int_timer2
Start Timer2
Config Int0 = Rising
Config Int1 = Rising
On Int0 Int_0
On Int1 Int_1

Dim Data_array(9) As Byte
Dim Speed_left As Word At Data_array(1) Overlay
Dim Speed_right As Word At Data_array(3) Overlay
Dim Count_left_old As Word
Dim Count_right_old As Word
Dim Timer_left As Word
```

```
Dim Timer_right As Word
Dim Timer0_overflow_l As Byte
Dim Timer2_overflow_r As Byte
Dim Cnt As Byte
Dim Twi_control As Byte
Dim Twi_status As Byte

Const Wordmax = 65535

Enable Interrupts
Enable Int0
Enable Int1
Enable Timer0
Enable Timer2
Twsr = 0
Twdr = &HFF
Twar = &H40
Twcr = &B01000100
Speed_left = 10000
Speed_right = 10000
Do
  If Timer0_overflow_l > 14 Then                    '229ms and no
Toothinterrupt (255 x 1024 x 14 x 62,5ns = 229ms)
      Timer0_overflow_l = 0
      Speed_left = 10000
  End If
  If Timer2_overflow_r > 14 Then
      Timer2_overflow_r = 0
      Speed_right = 10000
  End If
  Twi_control = Twcr And &H80
  If Twi_control = &H80 Then
    Twi_status = Twsr And &HF8
    Select Case Twi_status
      Case &HA8:
        Twdr = Data_array(1)
        Twcr = &B11000100
        Cnt = 2
      Case &HB8:
        Twdr = Data_array(cnt)
        Incr Cnt
        Twcr = &B11000100
      Case Else :
```

```
          Twcr = &B11000100
      End Select
    End If
Loop

Int_0:
    Timer0_overflow_l = 0
    Timer_left = Timer1
    If Timer_left >= Count_left_old Then
      Speed_left = Timer_left - Count_left_old
    Else
      Speed_left = Wordmax - Count_left_old
      Speed_left = Speed_left + Timer_left
    End If
    Count_left_old = Timer_left
    If Speed_left > 10000 Then Speed_left = 10000
Return

Int_1:
    Timer2_overflow_r = 0
    Timer_right = Timer1
    If Timer_right >= Count_right_old Then
      Speed_right = Timer_right - Count_right_old
    Else
      Speed_right = Wordmax - Count_right_old
      Speed_right = Speed_right + Timer_right
    End If
    Count_right_old = Timer_right
    If Speed_right > 10000 Then Speed_right = 10000
Return

Int_timer0:
    Incr Timer0_overflow_l
Return

Int_timer2:
    Incr Timer2_overflow_r
Return
```

Anexo D

Código dos sensores ultrassons controlados por microcontrolador ATmega 328

```
/*
 HC-SR04 sensor de distancia]
 VCC to arduino 5v, GND to arduino GND
 Echo Arduino pin 13, Trig Arduino pin 12, sensor de baixo
 Echo2 Arduino pin 7, Trig2 Arduino pin 8, sensor esquerdo
 Echo3 Arduino pin 2, Trig3 Arduino pin 4, sensor direito
 Led Vermelho Arduino pin 11
 Led Verde Arduino pin 10
 Led Amarelo 1 Arduino pin 9
 Led Amarelo 2  Arduino pin 6
 330 ohm resistencia para cada um dos LEDs
 */

const int trigPin = 13;
const int echoPin = 12;
const int echoPin2 = 7;
const int trigPin2 = 8;
const int echoPin3 = 2;
const int trigPin3 = 4;
#define led 11 // Led Vermelho
#define led2 10 // Led Verde
#define led3 9 // Led Amarelo 1
#define led4 6 // Led Amarelo 2

void setup() {
  Serial.begin(9600); // initializa comunicaçao serial
  pinMode(trigPin, OUTPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(echoPin2, INPUT);
  pinMode(echoPin3, INPUT);
  pinMode(led, OUTPUT);
```

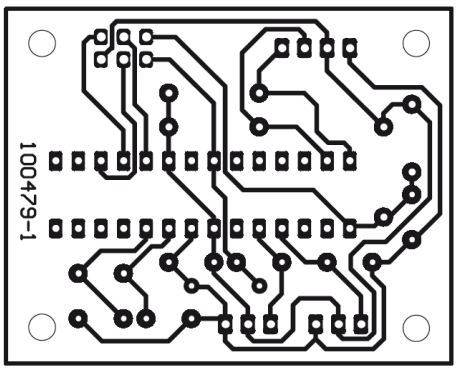
```
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
}

void loop()
{
  float duration, duration2, duration3, cm, cm2, cm3;
  digitalWrite(trigPin, LOW);  // pulso do trig
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // ve o tempo de resposta do echo

  digitalWrite(trigPin2, LOW); // pulso do trig2
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  duration2 = pulseIn(echoPin2, HIGH); // ve o tempo de resposta do echo2

  digitalWrite(trigPin3, LOW); // pulso do trig3
  delayMicroseconds(2);
  digitalWrite(trigPin3, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin3, LOW);
  duration3 = pulseIn(echoPin3, HIGH); // ve o tempo de resposta do echo3

 cm = (duration/2) / 29.1;    //sensor ultrasons 1 frente-esquerdo
 cm2 = (duration2/2) / 29.1;  //sensor ultrasons 2 frente-direito
 cm3 = (duration3/2) / 29.1;  //sensor ultrasons 3 baixo

  if (cm < 30 || cm2 < 30 || cm3 > 20) {    // Quando distancia menor que 30cm
e sensor de baixo maior que 20cm
    digitalWrite(led,HIGH); // Led Vermelho acende
    digitalWrite(led2,LOW); // Led Verde apaga
}
```

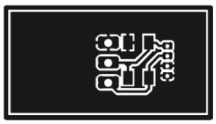
```
  else { // Quando distancia maior que que 30cm e sensor de baixo maior que
20cm
    digitalWrite(led,LOW); // Led Vermelho apaga
    digitalWrite(led2,HIGH); // Led Verde acede
  }

  if (cm < 60) {     // Quando distancia menor que 60cm e sensor de baixo
maior que 20cm
    digitalWrite(led3,HIGH); // Led Amarelo1 acende
}
  else { // Quando distancia maior que 10cm
    digitalWrite(led3,LOW); // Led Amarelo1 apaga
  }

  if (cm2 < 60) {     // Quando distancia menor que 60cm e sensor de baixo
maior que 20cm
    digitalWrite(led4,HIGH); // Led Amarelo 2 acende
}
  else { // Quando distancia maior que 10cm
    digitalWrite(led4,LOW); // Led Amarelo 2 apaga
  }

  if (cm >= 200 || cm <= 0){ // Quando distancia <=0 ou >=200
    Serial.print("fora de alcance"); // Quando distancia <=0 ou >=200
    Serial.print(" , ");
  }
  else {
    Serial.print(cm); //Display da distancia do echo em cm
    Serial.print("cm sensor frente-esquerdo");
    Serial.print(" , ");
  }
  if (cm2 >= 200 || cm2 <= 0){
    Serial.print("fora de alcance"); //Display 'fora de alcance'
    Serial.print(" , ");
  }
  else {
    Serial.print(cm2); //Display da distancia do echo2 em cm
    Serial.print("cm sensor frente-direito");
```

```
  Serial.print(" , ");
 }
 if (cm3 >= 200){
  Serial.print("fora de alcance"); //Display 'fora de alcance'
  Serial.println(" ");
 }
 else {
  Serial.print(cm3); //Display da distancia do echo2 em cm
  Serial.print("cm sensor baixo");
  Serial.println(" ");
 }
 delay(200);
}
```

Anexo E

Placa de circuito impresso dos sensores magnéticos





Anexo F

Placa de circuito impresso do controlo sensores de ultrassons