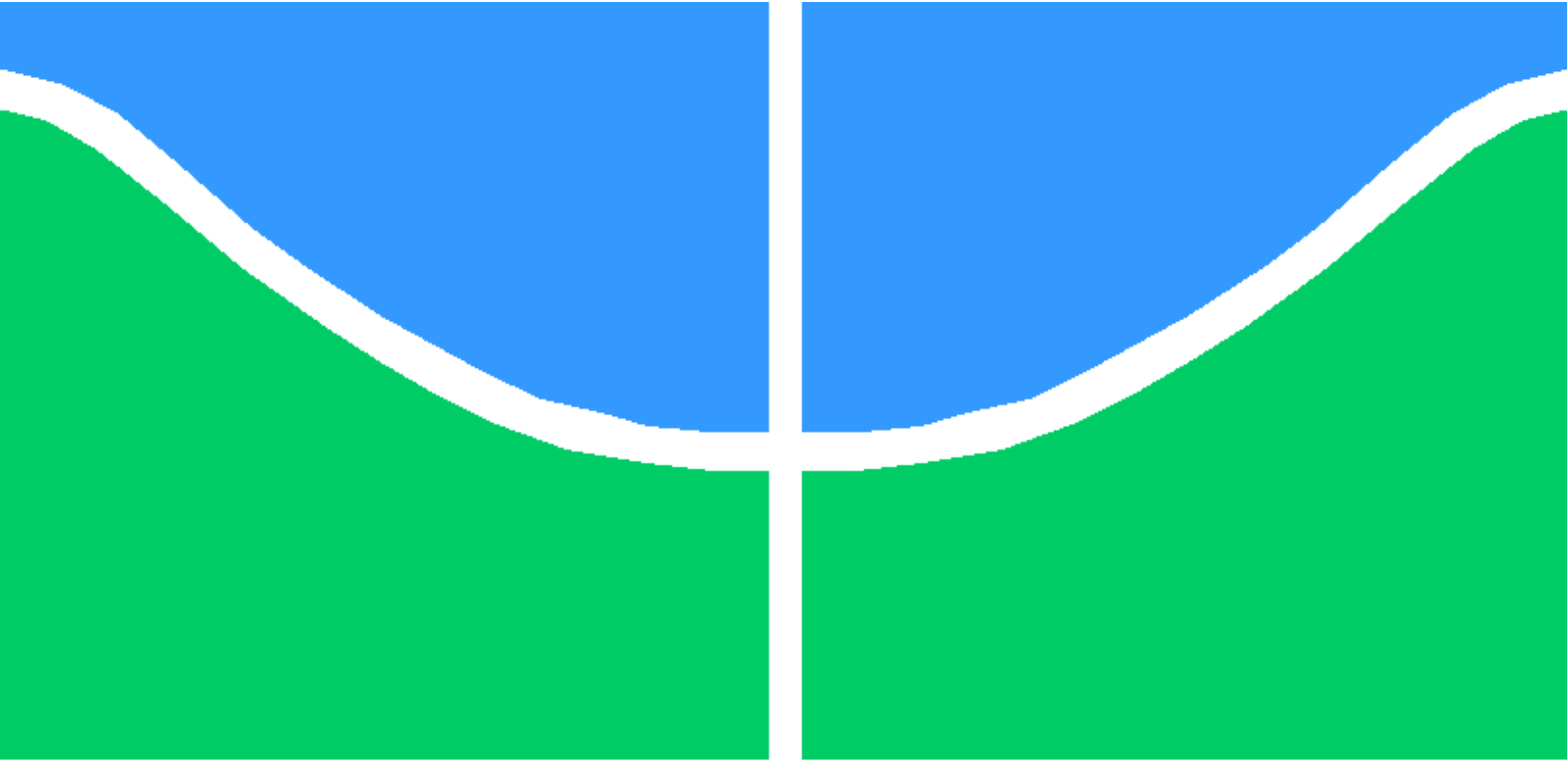


TRABALHO DE GRADUAÇÃO

ADEQUAÇÃO DE UM SISTEMA DE LOCOMOÇÃO DE UM ROBÔ QUADRÚPEDE PARA AVALIAÇÃO DE ALGORITMOS DE APRENDIZAGEM

Gauss Fernandis Batista

Igor Ferreira Cardoso

Brasília, Julho de 2007

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ADEQUAÇÃO DE UM SISTEMA DE LOCOMOÇÃO DE UM ROBÔ QUADRÚPEDE PARA AVALIAÇÃO DE ALGORITMOS DE APRENDIZAGEM

Gauss Fernandis Batista

Igor Ferreira Cardoso

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Gradação em Engenharia Elétrica

Banca Examinadora

Prof. Geovany Araújo Borges - Docteur, ENE/UnB _____
Orientador

Prof. Alexandre R.S. Romariz - Ph.D.,ENE/UnB _____
Co-orientador

Prof. Adolfo Bauchspiess - Dr.,ENE/UnB _____
Examinador interno

Ricardo Zelenovsky - Dr., ENE/UnB _____
Examinador interno

Dedicatórias

Aos meus pais, irmãos e minha amada Alice

Igor Ferreira Cardoso

Aos meus pais e à minha irmã Gausielle.

Gauss Fernandis Batista

Agradecimentos

A Deus, Luz e Inspiração da minha vida.
A meus pais, por se preocuparem com o meu bem e com a realização de meus sonhos.
À minha irmã, por entender à minha ausência e torcer pela minha felicidade.
Aos meus amigos, pela amizade que demonstraram nesses momentos difíceis.
Ao Igor, amigo que não me deixou desanimar.
Aos professores Geovany Borges e Alexandre Romariz, pela boa orientação.

Gauss Fernandis Batista

Aos meus pais, por terem sempre me apoiado nos meus momentos mais difíceis e auxiliado com minhas decisões.
À minha amada, Alice por seu amor acima de tudo e paciência em todos os momentos difíceis durante minha graduação.
Aos meus orientadores, professores Geovany Borges e Alexandre Romariz, pela paciência compreensão nesse projeto.
Ao professor Ricardo Zelenovsky, meu primeiro orientador na universidade, por sua alegria, bondade e presteza em ajudar os outros.
A todos os meus colegas de curso que se tornaram uma segunda família, principalmente ao Renan Utida, Rogério Richa e Arthur Assumpção que se tornaram como irmãos inseparáveis e que compartilharam os melhores, piores e mais difíceis momentos de minha vida.
Em especial ao Gauss Fernandis pela garra e determinação de tentar achar as soluções até no último instante.

Igor Ferreira Cardoso

RESUMO

O objetivo desse trabalho é desenvolver um algoritmo de treinamento para o sistema de locomoção de um robô quadrúpede. Para obter tal objetivo foram desenvolvidos métodos de geração de sinal de modulação de pulso que controlam múltiplos servomotores. O sistema de comunicação apresenta um protocolo próprio para melhorar a performance e proporcionar maior robustez e adaptabilidade ao sistema. Um algoritmo de aprendizagem, treinado com os dados coletados por um acelerômetro de três eixos, foi utilizado para desenvolver uma marcha para essa plataforma.

ABSTRACT

The objective of this work is to develop an algorithm of training for the locomotion system of a quadruped robot. To achieve such objective, methods of Pulse Width Modulation to control multiple motors had been developed. A protocol had been used to improve robustness and adaptability of the communication system. An algorithm of learning, trained with the data collected by an accelerometer of three axles, was used to develop a march for this plataform.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

T	Período	[s]
L	Largura de um sinal de MLP	[s]
P	Posição de calibração que varia de 0 a 432 para o *clock* de 11,0592Mhz com *prescaler* em 64	[Ciclos de *Clock*]
g	Aceleração da gravidade	9,8 [m/s^2]
a	Aceleração	[m/s^2]
v	Velocidade	[m/s]
s	Posição	[m]
V	Potencial elétrico	[V]
inc	Inclinação	[V]
$Sens$	Sensibilidade	[mV/g]
LSb	1g/1024 em 12 bits de representação a uma escala de 2-g para um LIS3LV02DQ	
U	Conjunto de tempos de disparo das patas do robô quadrúpede	
L	Conjunto de tempos gerados aleatóriamente pelo algoritmo de passeio aleatório	
R	Razão entre o espaço percorrido no eixo X com relação ao espaço percorrido no eixo Y	
t	Intervalo de tempo	[s]
r	Valor aleatório de tempo obtido no intervalo de $-t_{max}$ a t_{max}	[s]

Símbolos Gregos

τ	Ciclo de trabalho	[%]
Δ	Variação entre duas grandezas similares	
θ	Inclinação máxima obtida durante o movimento	[^{o}]

Subscritos

$pulso$	Refere-se ao intervalor de pulso dos sinais de MLP
min	Valores mínimos(a)
max	Valores máximos(a)
mov	Movimento (ex: Dado de movimento, Período de movimento etc)
MLP	Refere-se a dados de sinal com Modulação por Largura de Pulso
in	entrada
out	saída
ref	referência
dd	suprimento
x	valor referente ao eixo x
y	valor referente ao eixo y
z	valor referente ao eixo z
A	Valor referente à pata A
B	Valor referente à pata B
C	Valor referente à pata C
D	Valor referente à pata D
$base$	Refere-se aos valores utilizados como base para os cálculos no passeio aleatório
$teste$	Refere-se aos valores utilizados como teste do algoritmo de treinamento

Siglas

LARA	Laboratório de Robotica e Automação
MLP	Modulação por Largura de Pulso
A/D	Analógico-digital
PAA	Passeio Aleatório Adaptativo

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Dentre os grandes acontecimentos do século XX, o elevado desenvolvimento técnico-científico merece destaque. O advento da eletrônica, com o desenvolvimento de transistores e circuitos integrados, o desenvolvimento das telecomunicações, com meios de transmissão cada vez mais eficientes, dentre outras inovações, têm disseminado a utilização de máquinas em todas as camadas sociais.

O ser humano comum, que possuía até então pouco acesso às tecnologias, vê-se cercado por inúmeros equipamentos eletrônicos, dentre os quais o computador pessoal e os eletrodomésticos. Não só integra a utilização desses aparelhos à sua vida cotidiana, como também necessita da existência deles.

Campos de pesquisa como a robótica, até então inviáveis pela tecnologia da época, não só tornam-se almejáveis como também precursores de um novo modo de pensar sobre as máquinas. O homem já não se satisfaz com máquinas que necessitem explicitamente de seu controle, nem com máquinas que, pré-programadas, executem seqüências fixas de decisão. Sua inteligência exige o desenvolvimento de máquinas inteligentes. Surge o conceito de robôs autônomos.

Robôs autônomos são máquinas inteligentes capazes de realizar tarefas no mundo por si mesmas, sem controle explícito do homem sobre seus movimentos [5] . São máquinas que simulam características humanas como o sentir, o pensar e o agir. Para tal, dispõem de componentes sensores, processadores e atuadores.

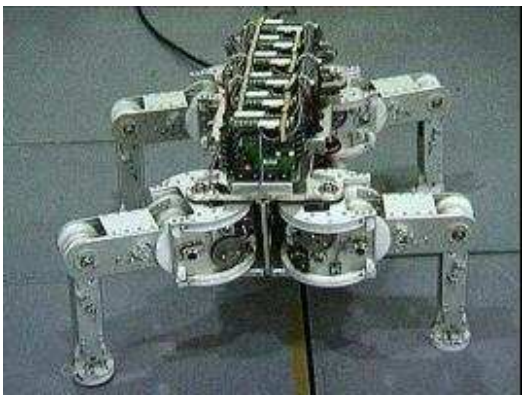
Para conhecer esta inteligência artificial, é necessário entender o funcionamento destes componentes.

Os sensores são necessários para capturar informações do ambiente externo e para monitorar seu ambiente interno. Freqüentemente, estes sensores são mecanismos que tentam imitar algumas propriedades sensitivas dos animais. Tais propriedades compõem complexos sistemas com inúmeras dimensões sensoriais. A esses sistemas denominam-se sentidos, sendo um o sentido da visão. Nos animais, o olho é o sensor de visão capaz de captar a intensidade luminosa. Nos robôs, câmaras são utilizadas com a mesma função.

Todo sentido, por sua vez, abrange, além de dimensões sensoriais, centros de processamento de informações. Esses centros são hábeis interpretadores das informações recebidas pelos sensores. Nos animais, tal função é desempenhada precipuamente, pelo cérebro. Em robôs, softwares são utilizados para essa função.

Os atuadores, por sua vez, são necessários para que o robô interaja fisicamente com o ambiente em que opera. Nos animais, os músculos e os ossos são os principais mecanismos de atuação. Em robôs, músculos artificiais de vários tipos, motores elétricos e atuadores pneumáticos e hidráulicos realizam essa interação.

O desenvolvimento de robôs autônomos se intensificou bastante nos últimos anos. Nesse período, alguns desses robôs foram comercializados, dentre eles, os quadrúpedes TITAN, desenvolvidos no Laboratório Hirose-Yoneda do Instituto Tóquio de Tecnologia, e o robô de entretenimento AIBO, desenvolvido pela Sony.



(a) TITAN VIII

(b) TITAN IX

Figura 1.1: Os mais recentes robôs quadrúpedes da série TITAN

1.1.1 Os quadrúpedes TITAN

Os quadrúpedes TITAN participam da segunda geração de robôs. Tal geração inova em tecnologia ao adquirir mobilidade. Diferem dos robôs de manipulação, únicos existentes até então, pois não mais fixos a uma sustentação, possuem rodas ou patas para se locomover.

Os primeiros robôs móveis utilizavam rodas como meios de locomoção. Eram equipados com sensores a laser, receptores GPS e outros dispositivos sensores. O controle da posição, da orientação e da velocidade era obtido através dos motores elétricos que impulsionavam as rodas.

O advento dos robôs sobre rodas expandiu bastante o conhecimento sobre controle de mobilidade, tornando possível almejar a criação de robôs que imitassem o movimento de animais. sses passaram a ser desenvolvidos com quatro, seis ou oito patas. Um desses robôs é o TITAN VIII, mostrado na Figura 1.1 (a), cujas características são:

- 12 graus de liberdade (cada pata tem 3 deles);
- Um peso de 19kg, sem incluir bateria e computador, e habilidade para carregar uma carga de 5 a 7kg;
- Um potenciômetro em cada articulação para um controle por realimentação de posição;
- sistemas únicos de controle conhecidos como Titech, projetado no laboratório do Hirose, como atuadores de servo-amplificadores para motores;
- habilidade para caminhar com velocidade máxima de 0,9 m/s, o que é consideravelmente rápido para uma máquina deste peso.

O último da série TITAN é o TITAN IX, mostrado na Figura 1.1(b). Esse robô é equipado com pés adaptativos, possibilitando que caminhe sobre terrenos irregulares; ele também é capaz de se recuperar de quedas e subir degraus autonomamente. Uma das mais interessantes características deste robô é que ele pode se sustentar sobre três patas e usar a quarta, equipada com um pegador ajustável, como um manipulador.[5]

(a) AIBO ERS-7M3, da Sony.

(b) JoinMax, da MCII Robot.

(c) Robopet, da WowWee.

Figura 1.2: Robôs quadrúpedes da atualidade.

1.1.2 Robôs de entretenimento

Os robôs de entretenimento inovaram ao simular comportamentos típicos de animais. Eles são bastante amigáveis, tanto em suas respostas a atitudes humanas, quanto em sua aparência. Dentre eles, podemos destacar três modelos comerciais: o AIBO ERS-7M3, da Sony, o JoinMax, da MCII Robot, e o Robopet, da WowWee. Tais modelos são apresentados na Figura 1.2 .

Dos robôs supracitados, o mais conhecido é o AIBO (Artificial Intelligence Robot). Sua terceira geração AIBO ERS-7, lançada em 2004, apresenta as seguintes características:

1. É capaz de um número de comportamentos autônomos, tais como seguir uma bola e com ela brincar, em virtude de seu excelente sistema de visão e do acoplamento da visão ao movimento das patas.
2. Se cair, é capaz de reerguer-se.
3. Pode receber comandos de entrada provenientes de usuários humanos pelo toque ou pela voz, o que permite executar outros comportamentos, tais como deitar-se, sentar-se e abanar o rabo.
4. O computador on-board é tão sofisticado que laboratórios (se tiverem acesso ao código da Sony) podem programar novos comportamentos.
5. Um AIBO interage com outro, de modo que eles são usados atualmente em competições de “futebol de robô” por algumas instituições.

1.2 OBJETIVOS DO PROJETO

Propõe-se desenvolver um sistema de locomoção para robôs quadrúpedes. Esse sistema utilizará a aprendizadem empiríca para a obtenção de uma sequência de movimentos eficiente. Para tal, empregar-se-ão algoritmos de passeio aleatório adaptivo com dados de entrada obtidos pelo processamento das medidas de aceleração, que são capturadas por um acelerômetro. Objetiva-se também mostrar os elementos de hardware e de software que possibilitam esta aprendizagem. Nesses processos pretende-se padronizar o sistema de modo que facilite a utilização da plataforma em futuros trabalhos.

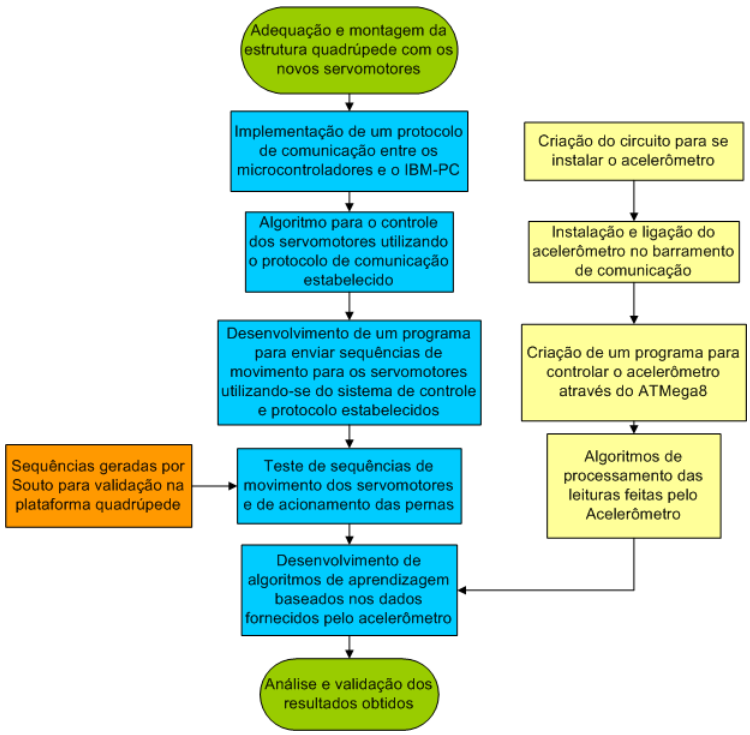


Figura 1.3: Fluxograma do desenvolvimento do projeto.

1.3 APRESENTAÇÃO DO MANUSCRITO

O Capítulo 2 descreve a arquitetura proposta para o sistema eletrônico, explicando, além do funcionamento dos sensores empregados, os softwares desenvolvidos para o processamento dos dados provenientes desses sensores e para a implementação da aprendizagem por passeio aleatório adaptativo. Em seguida, os resultados experimentais, obtidos empiricamente e por simulação, são discutidos no Capítulo 3.As conclusões deste trabalho são mostradas no Capítulo 4. Os anexos contém material complementar.

1.4 ETAPAS DE DESENVOLVIMENTO

Esse projeto é a continuação de dois outros desenvolvidos no laboratório LARA, na Universiada de Brasília. Alguns problemas encontrados durante esses projetos foram corrigidos para um melhor aproveitamento no presente projeto. Dentre os problemas encontrados, podemos citar a falta de potência dos servomotores utilizados, que foram substituidos logo no início do presente projeto. O fluxograma dos processos desenvolvidos nesse projeto é apresentado na Figura 1.3.

Para controlar os novos servomotores era necessário a geração de sinais de modulação por largura de pulso (MLP). Circuitos com microcontroladores ATMega8 foram utilizados para gerar os sinais. Esses microcontroladores se comunicam através de um barramento RS-485 e são controlados pelo IBM-PC através de um barramento RS-232. A integração dos dois barramentos é feito por um circuito central, onde um acelerômetro foi instalado posteriormente. Foi adotado para o sistema de comunicação um protocolo que tem o intuito de facilitar a integração de novos sistemas ao barramento de comunicação.

Um circuito para o acelerômetro foi projetado para que o mesmo se integrasse no barramento e pudesse transmitir os seus dados coletados para o IBM-PC. Algoritmos de processamento desses dados foram implementados no microcontrolador afim de calcular parâmetros que pudessem ser utilizados para avaliar sequências de movimento do quadrúpede. Tais sequências são enviadas aos servomotores pelo barramento de comunicação e são controladas por *softwares* desenvolvidos na plataforma *Linux*.

As sequências testadas foram desenvolvidas paralelamente a este projeto por Souto[6]. Utilizando como base as sequências geradas por [6] e os dados coletados pelo acelerômetro, pretende-se treinar o robô com algoritmos de aprendizagem computacional, para que ele caminhe para frente com a maior velocidade possível.

2 DESENVOLVIMENTO

Este capítulo apresenta a definição do problema, a solução proposta e a teoria que a rege.

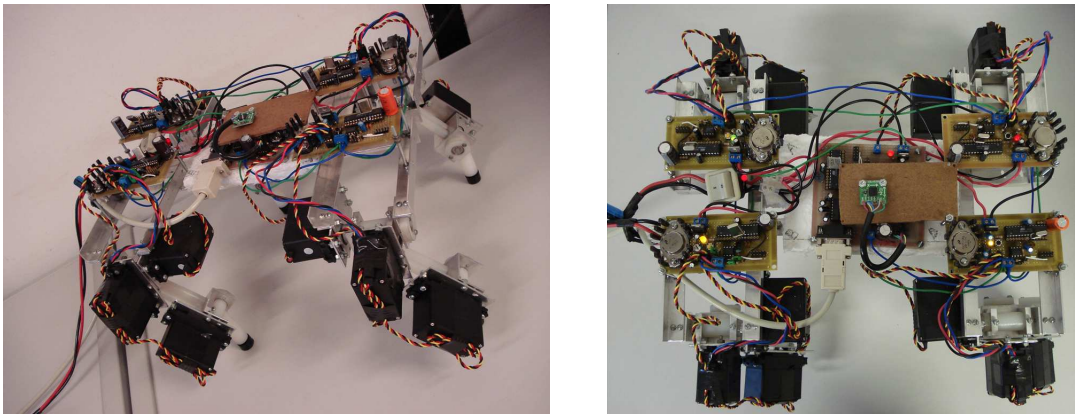
2.1 SISTEMA DE LOCOMOÇÃO

2.1.1 Introdução

Para a criação do robô tomou-se como base de inspiração alguns já existentes, como o JoinMax da MCII Robot (Figura 1.2(a)), o AIBO ERS-7M3 da Sony (Figura 1.2(b)) e o Robopet da WowWee (Figura 1.2(c)). Todos eles são quadrúpedes e possuem três graus de liberdade em cada pata. A pesquisa com quadrúpedes com três graus de liberdade é ainda recente, tornando-se mais intensiva após o lançamento desses modelos, pois os mesmos são reprogramáveis e permitem a implementação de algoritmos em seus sistemas embarcados. Algumas pesquisas já foram feitas utilizando estas plataformas como a base do desenvolvimento [7]. Neste trabalho será utilizada uma plataforma única, desenvolvida por membros do LARA da Universidade de Brasília em projetos anteriores [2, 8]. As fotos da plataforma quadrúpede desenvolvida no LARA podem ser vistas nas Figuras 2.1(a) e 2.1(b).

2.1.2 Servomotores

Os servomotores utilizados na plataforma original [8] foram substituídos nesse projeto pelo modelo HS-755HB da HITEC, pois os antigos não possuiam torque suficiente para sustentar o próprio peso. O HS-755HB possui um torque máximo de 1,08 N.m e uma velocidade de 215°/s, contra 0,31 N.m a uma velocidade de 260°/s dos servos anteriores [8].



(a) Vista em perspectiva (b) Vista por cima

Figura 2.1: Plataforma quadrúpede usada como base para esse projeto.

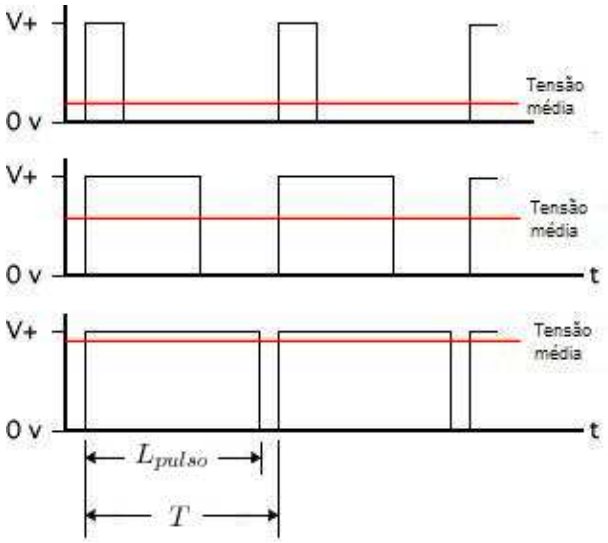


Figura 2.2: Exemplo de sinais com Modulação por Largura de Pulso (MLP). Modificado de: http://www.micromouseinfo.com/introduction/dcmotors.html

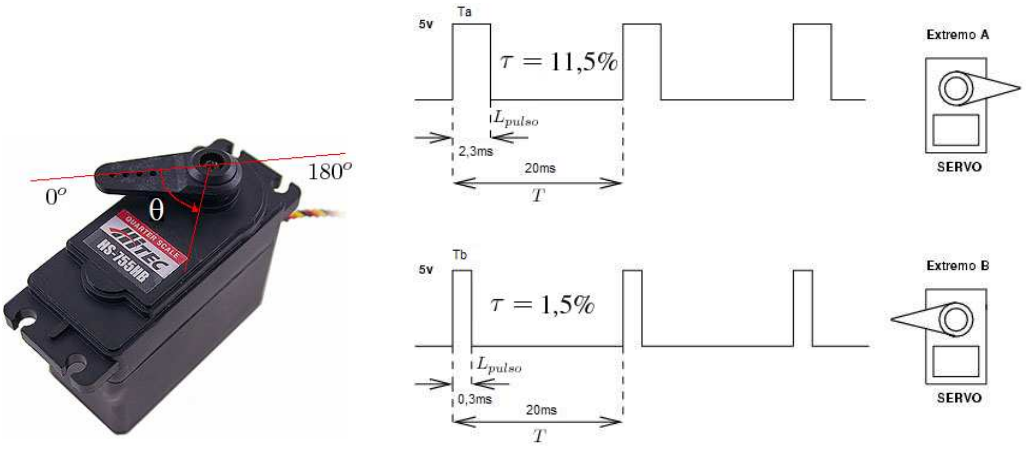
2.1.2.1 Acionamento por Modulação de Largura de Pulso (MLP)

Cada servomotor possui três terminais: dois de alimentação e um para o sinal de controle. O sinal de controle é feito através da modulação por largura de pulso (*PWM - Pulse-Width Modulation*). A MLP é usada para vários tipos de aplicação: em telecomunicações, onde a largura dos pulsos corresponde a valores de dados específicos codificados/decodificados; em circuitos elétrico/eletrônicos onde é também utilizada para variar o valor da transferência de potência entregue a uma carga sem as perdas ocorridas normalmente devido à queda de tensão por recursos resistivos; para controlar a velocidade de motores de corrente contínua e etc.

Pode-se pensar que o controle dos servomotores HS-755HB seria feito da mesma maneira que os motores de corrente contínua comuns, porém não é a mesma coisa. Os servomotores HS-755HB possuem um circuito de controle interno que só responde a um determinado tipo de MLP, enquanto que os motores comuns de corrente contínua (aqueles sem circuito interno) não possuem um terminal de controle e são controlados pela tensão média fornecida em seu terminal de alimentação. O HS-755HB usa o sinal de MLP para determinar o seu posicionamento entre 0^o e 180^o, permanecendo-se fixo na posição até a próxima instrução (a maneira de se passar as instruções será tratada a seguir), enquanto os motores comuns usam o sinal de MLP para controlar a sua velocidade de acordo com tensão média fornecida pelo sinal(Figura 2.2).

A MLP é basicamente um sinal de onda retangular que tem o ciclo de trabalho (τ) variável a uma frequência constante. Quando o sinal possui a largura do pulso (L_{pulso}) igual ao seu período (T), diz-se que está em seu valor máximo, portanto com o ciclo de trabalho $\tau = 100\%$, enquanto se a largura for mínima, não existirá onda, então $\tau = 0\%$. O ciclo de trabalho em uma MLP pode ser determinado através da seguinte equação:

$$\tau = (\frac{L_{pulso}}{T}) * 100[\%], \tag{2.1}$$



(a) Um exemplo da medida de angulação no HS-755HB. Modificado de: http://www.servocity.com/html/hs-755hb_1_4_scale.html

(b) Controle da angulação feita com o sinal de MLP. Modificado de: http://www.iearobotics.com/personal/juan/publicaciones/art3/html/nodo9.html

Figura 2.3: Ilustrações da posição do servomotor com relação ao ciclo de trabalho do sinal de MLP.

2.1.2.2 A geração dos sinais de MLP e o controle dos servomotores

O HS-755HB responde a sinais de MLP com frequência de 50Hz ($T = 20ms$) e ciclos de trabalho que variam de 1,5% a 11,5%, aproximadamente. Esse valores nunca são exatos, havendo diferenças de um servo para outro devido à fabricação. Idealmente, para um sinal com ciclo de trabalho de 1,5%, o servo se posicionaria em seu mínimo, ou seja, na referência de 0^o. Enquanto para sinais de MLP com 11,5%, em seu máximo, 180^o (Figuras 2.3(a) e (b)). Um programa de calibração foi desenvolvido para contornar a descalibração de fábrica dos ciclo de trabalho, porém ele será explicado mais adiante, quando for tratado a geração do sinal. Já a amplitude do sinal é estabelecida de acordo com a especificação de cada projeto. O HS-755HB responde a sinais de MLP com amplitudes padronizadas para circuitos TTL, ou seja, de 0 a 5 volts.

Neste projeto, para se gerar o sinal de MLP que controla os servomotores foi utilizado o microcontrolador ATMega8 da ATMEL [9]. Um ATMega8 é capaz de controlar vários servomotores simultaneamente. Num total de quatro foram usados para controlar todos os servomotores da plataforma, sendo cada um responsável por uma perna e portando controlando três servomotores. Essa configuração estrutural favorece o sistema de endereçamento no barramento de comunicação, mas esse tópico será abordado na seção seguinte.

No microcontrolador pode-se gerar os sinais de MLP controlando-se os pinos manualmente ou utilizando o seu gerador automático e interno de sinais. Inicialmente implementado no projeto por Cotta&Neto[8], os sinais gerados para cada um dos três servomotores eram feitos no modo automático. Infelizmente esse modo têm a sua disposição apenas um relógio com resolução de 16 bits e outros dois com 8 bits, limitando a resolução para dois dos três servomotores. Essa falta de resolução não era relevante nos servomotores utilizados anteriormente, porém ao instalarmos o servomotor de modelo HS-755HB foi verificado que os 8 bits eram insuficientes. O HS-755HB é mais sensível a alterações de ciclo de trabalho e ficava alternando entre dois valores possíveis de posição devido à truncamentos de valores causados pela baixa resolução. Essa alternância gera trepidações e prejudicam a estabilidade da plataforma quadrúpede. A solução encon-

trada foi gerar o sinal de MLP controlando manualmente os pinos do microcontrolador.

O controle manual dos pinos deve ser feito com uma temporização precisa, pois o HS-755HB exige que os sinais de comando possuam a frequência de 50Hz. Lembrando que a falta de precisão na temporização pode levar ao mesmo problema observado com as MLP geradas automaticamente pelo microcontrolador. O ATMega8 fornece opções de utilização de relógios internos gerados por circuitos RC, que são relativamente precisos e com frequência máxima de 8MHz, e relógios externos que podem ser tanto de circuitos RC quanto de cristais.

Primeiramente utilizou-se o relógio interno configurado em seu máximo, 8MHz. Para se controlar o tempo na rotina que gera o sinal de MLP utiliza-se a interrupção dos comparadores A e B [1] do contador 1 de 16 bits. A resolução de 16 bits permite ao contador em incrementar até 65536 ciclos de relógio, porém, como cada ciclo de relógio ocorre em $0,125\mu s$, os 65536 ciclos só permitiriam um período de no máximo $8,192ms$, que é insuficiente para gerar os 50Hz ($T = 20ms$) necessários.

Pode-se pensar que a solução seria reconfigurar o relógio interno para um valor mais baixo, porém isso não é necessário, pois o ATMega8 possui um registrador de divisor de escala (*prescaler*) que nos permite dividir o relógio atual por um determinado valor [2]. Então qual seria a diferença entre configurar o divisor de escala e selecionar um relógio mais baixo? Selecionando um relógio mais baixo fica-se sujeito a pequenas oscilações geradas pela imperfeição do circuito RC, enquanto utilizando-se o divisor de escala essas oscilações são também divididas, diminuindo o ruído e auxiliando na estabilização do valor do relógio. Para esse projeto foi utilizado o divisor de escala em 64, portanto resultando um relógio de $\frac{8000000}{64} = 125000Hz$ ou ciclos de relógio a cada $8\mu s$. Desta forma precisamos apenas de 2500 ciclos de relógio para $T = 20ms$.

Alternando-se entre as interrupções do comparador A e B é possível acionar e desativar os pinos do microcontrolador de maneira a gerar o sinal para os três servomotores. Como o relógio é suficientemente rápido, é possível gerar sinais para diversos servomotores utilizando apenas um contador. A Figura 2.4 mostra uma maneira inteligente de gerar os sinais intercalando-se o controle dos pinos dentro de um período de 20ms. Nesse período tem-se que 11,5% corresponde a $L_{pulso} = 2,3ms$ e 1,5% em $L_{pulso} = 0,3ms$ (Figura 2.3(b)). Para permitir uma calibração[3] mais abrangente (mesmo até para modelos diferentes de servomotor), configurou-se os contadores para gerar sinais que variam de 0% a 12,5% de ciclo de trabalho. Com o ciclo de trabalho máximo desejado(12,5%) no período de $T = 20ms$, pode-se controlar até 8 microcontroladores, pois 12,5% ocupam a apenas $L_{pulso} = 2,5ms$.

Posteriormente o relógio do ATMega8 foi alterado de interno para externo, onde um cristal de 11,0592 Mhz foi instalado. O motivo da mudança não foi a pela geração de sinais de MLP, mas sim porque a frequência de 11,0592 Mhz nos fornece vantagens na comunicação serial. Mais detalhes sobre essas vantagens, a comunicação serial e o barramento de comunicação dos microcontroladores será explicado na Seção 2.1.3. Com o relógio em 11,0592 Mhz e utilizando o mesmo divisor de escala (64), obtemos

[1] O contador 1 do ATMega8(*Timer/Counter 1*) possui dois comparadores, A e B. Uma interrupção é gerada toda vez que o contador 1 atinge um dos valores programados nos comparadores, portanto gerando a interrupção $SIG\_OUTPUT\_COMPARE1A$ ao atingir o valor programado no comparador A e $SIG\_OUTPUT\_COMPARE1B$ ao atingir o do comparador B. Porém essa interrupção só pode ocorrer novamente quando o contador 1 reiniciar. O valor de ciclos de relógio para o reinício é configurado em uma variável de *Overflow* [9].

[2] Os valores permitidos no divisor de escala do ATMega8 são potências de 2, ou seja, 2,4,8,16,32,64, etc...

[3] Detalhes da calibração será explicado em uma seção dedicada mais adiante.

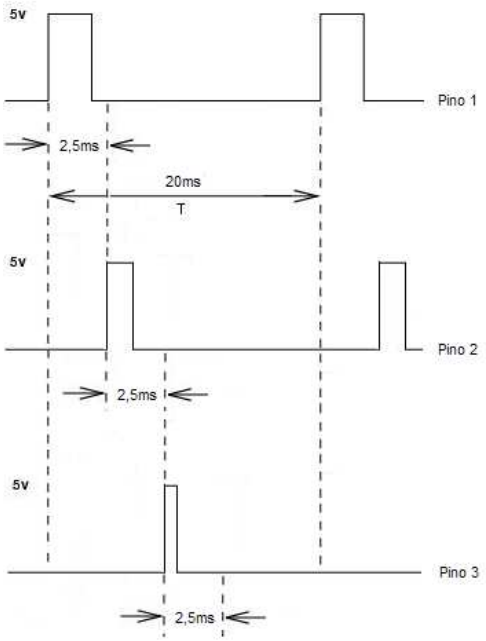


Figura 2.4: Geração de sinais de MLP com intercalação de pinos utilizando o *Timer/Counter 1* do ATMega8.

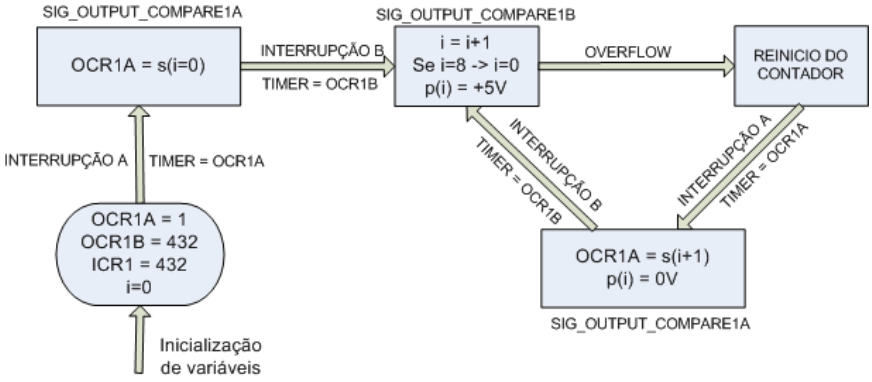


Figura 2.5: Fluxograma do algoritmo de geração de sinais de MLP.

uma frequência de relógio de 172800Hz. Para gerar o sinal com essa frequência e utilizando o método exemplificado na Figura 2.4, devemos configurar uma das interrupções e o *overflow* do contador 1 para ser disparado em $L_{pulso} = 2,5ms$[4]. Para o relógio de 172800Hz isso corresponde a 432 ciclos.

Para esse projeto, a interrupção do comparador B foi selecionada para disparar aos 432 ciclos, junto do *Overflow*. Já o comparador A irá controlar o ciclo de trabalho em cada pino gerador de sinal. No o ATMega8 o comparador A é configurado através do registrador chamado OCR1A; o comparador B de OCR1B, e o *overflow* de ICR1. A Figura 2.5 mostra o fluxograma desse algoritmo que gera o sinal de MLP. No fluxograma temos dois vetores importantes: $p(i)$ e $s(i)$, sendo que $p(i)$ corresponde ao estado atual do pino i (+5 ou 0 volts) e $s(i)$ é o valor correspondente ao ciclo de trabalho do pino i expresso em ciclos de relógio, sendo $i = \{0, 1, 2, ..., 7\}$.

[4]A interrupção irá ocorrer instantes antes do *Overflow* do contador.

Toda vez que a interrupção do comparador B é disparada, i é acrescido de uma unidade. Por exemplo: De acordo com a Figura 2.5, a interrupção do contador A é configurada inicialmente para disparar em 1 ciclo de relógio, e a do contador B e do *overflow*(ICR1) em 432 ciclos. Portanto a interrupção de A irá disparar logo no início, antes de B. A rotina da interrupção A irá configurar o seu próximo valor de disparo, como por exemplo, para $i = 0$, $Servo(i = 0) = 260$ ciclos. Esse novo valor só entrará em vigor depois do próximo reinício do contador.

Imediatamente antes do final do primeiro ciclo, a interrupção de B irá disparar. Sua rotina irá acrescer em uma unidade a variável i e colocar o pino i do microcontrolador em estado de alta ($Pino(i) = +5$ volts). Com o reinício do contador, entra em vigor o novo valor de disparo da interrupção A, que corresponde a uma largura de pulso de $L_{pulso} = 1,5ms$. Ao ser disparada, a interrupção A irá voltar com o pino i para o valor lógico 0($Pino(i) = 0$ volts) e também atualizar o disparo da interrupção para o próximo ciclo do contador($OCR1A = Servo(i + 1)$).

Desta forma, o ciclo de trabalho para esse pino no período total foi de $\tau = \frac{1,5ms}{20ms} = 7,5\%$. No segundo disparo da interrupção B, a variável i é novamente acrescida e desta vez outro pino será colocado em alta (normalmente a contagem dos pinos acompanha a contagem da variável i). E assim se repete até que i atinja o seu valor máximo($i = 7$) e seja reiniciado. Neste ponto, a interrupção B já terá disparado 8 vezes, formando um período exato de $T = 20ms$.

Para controlar a posição dos servomotores basta controlar o valor gravado no registrador OCR1A. Para que o sistema funcione com mais praticidade, os dados são gerados no PC e enviados via serial para o microcontrolador, que por sua vez os armazena em um vetor na sua memória RAM e atualiza os registradores necessários. Como são três servos para cada pata, os valores enviados são três, cada um controlando o ciclo de trabalho de um servo. A quantidade de bytes necessários para enviar esse comando de controle via serial e como esses dados são tratados pelo microcontrolador serão tratados na seção seguinte.

2.1.3 Barramento de comunicação

As microcontroladoras são controladas através de mensagens recebidas/enviadas em um barramento de comunicação com padrão RS-485, onde o IBM-PC é o único mestre e todas as microcontroladoras se comportam como escravo. Além dos quatro ATMega8 que controlam as pernas existe também um microcontrolador central, responsável por controlar o acelerômetro. Em projetos anteriores, esse microcontrolador era responsável por controlar o rabo e o pescoço do quadrúpede [2, 8]. Esse circuito central também é responsável por integrar os dois padrões de comunicação utilizados no sistema: o padrão entre as microcontroladoras, RS-485, e o padrão de comunicação do mestre com o circuito central, serial RS-232. A Figura 2.6 ilustra o barramento de comunicação de toda a plataforma quadrúpede.

A comunicação entre o microcontrolador central e o acelerômetro é feito utilizando-se a padrão SPI, mas esse tópico será abordado em uma seção especial sobre o sistema de sensoriamento (Seção 2.3.2).

A *Eletronics Industry Association* (EIA) foi quem produziu os padrões RS-232, RS-422 e RS-485. EIA *Standards* eram antes marcados com o prefixo “RS” para indicar *Recommended Standard*. Basicamente existem dois sistemas de transmissão de dados [5]:

[5]http://www.rs485.com/rs485spec.html

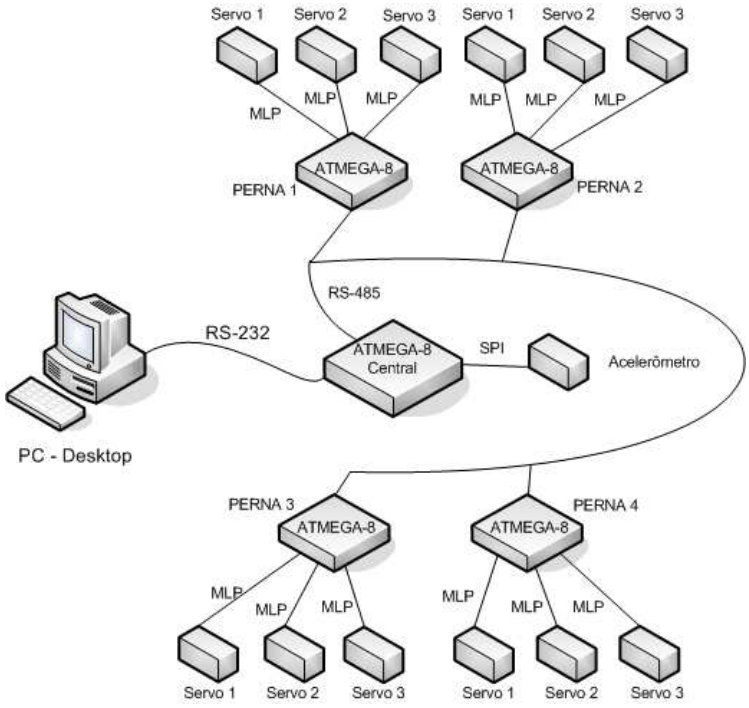


Figura 2.6: Barramento de comunicação de toda a plataforma quadrúpede.

- A de terminal simples, onde se utiliza apenas um condutor para transportar o sinal elétrico e o outro usado para transportar a referência elétrica (terra de sinal). Um tipo de trasmissão simples é o padrão RS-232.
- E a transmissão diferencial, onde são usadas duas linhas de transmissão para os dados. Um condutor transporta o sinal de dados enquanto o outro transporta o sinal invertido (diferencial), sendo a leitura de sinal no receptor efetuada pela medição da diferença de potencial elétrico entre os dois condutores (em lugar da diferença de potencial medida relativamente ao condutor de terra no caso da transmissão não diferencial). Tipos de transmisão diferencial estão os padrões RS-422 e RS-485.

2.1.3.1 O padrão RS-485

Quando comunicação de grande volume de dados ou sobre longas distâncias é necessário, métodos de terminal simples são inadequados. A transmissão diferencial oferece uma performance muito superior nesses casos. O sinal diferencial ajuda a anular os efeitos de oscilação do terra (referência para terminais simples) e diminuir os ruídos causados por tensões induzidas e interferência eletromagnética em longos cabos. Os padrões diferenciais RS-422 e RS-485 foram projetados para distâncias de até $1219,2$ metros (4000 pés) e volume de dados que variam entre 10Mb/s a 100kb/s, dependendo da distância. Ambas essas características superam o padrão RS-232.

A outra vantagem do padrão RS-485 com relação ao RS-232 é a capacidade de comunicação de multi-ponto, sendo que em um mesmo barramento é possível a comunicação de até 32 dispositivos (cada um com o seu *driver* de comunicação). Com a introdução de repetidores no barramento e a utilização de *drivers* de alta impedância é possivel extender esse número para centenas ou até milhares de dispositivos. Por esse motivo que foi estabelecido apenas um mestre no sistema de comunicação, evitando que múltiplos

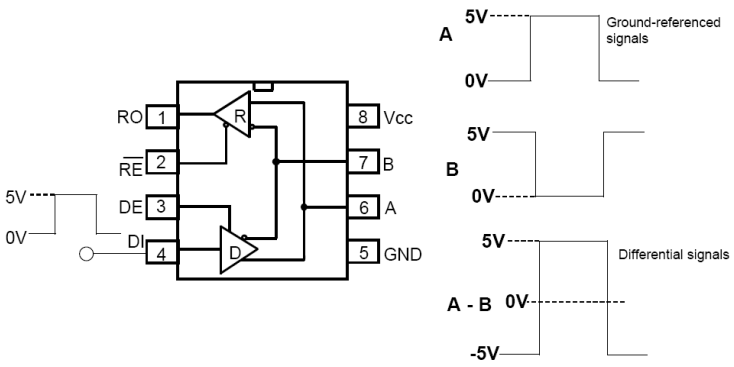


Figura 2.7: Interface entre o padrão de terminal simples e o RS-485. Fonte:[1]

dispositívos tentem enviar pacotes de dados ao mesmo tempo e gerando colisão de dados.

Como mencionado anteriormente, o mestre do barramento é o IBM-PC, porém o mesmo não possui interface de comunicação RS-485. A solução para tal problema foi utilizar a comunicação serial entre o IBM-PC e dispositivo central do barramento(Figura 2.6) e converter os dados recebidos pelo RS-232 para o barramento RS-485 com os drivers de ambos os padrões[6]. Os circuitos responsáveis por tais conversões são os mesmos utilizados em [10] e são apresentados nos anexos deste trabalho. A Figura 2.7 mostra como o sinal diferencial é gerado a partir de um sinal com referência no terra.

Conforme a Figura 2.7, o *driver* do RS-485 transforma um sinal TTL de entrada em dois sinais diferentes. O sinal A possui o nível lógico do sinal TTL da entrada, enquanto o sinal B é o seu complementar. Tanto o sinal A quanto o sinal B possuem amplitude entre 0V e 5V. Na recepção a diferença entre A e B é analisada. Se ela for maior do que 200mV, é considerado como nível lógico “1”. Por outro lado, se esta diferença for menor do que -200mV, é considerado como o nível lógico “0”. Se o sinal recebido estiver entre -200mV e 200mV, o nível lógico é indefinido [10].

2.1.3.2 O padrão RS-232

O padrão RS-232 foi proposto em 1962, e desde então tem sido usado de intensivamente pela indústria[7]. Esse padrão permite a comunicação entre um transmissor e um receptor a uma taxa de dados relativamente baixa (até aproximadamente 20Kb/s) e com baixas distâncias (15,24 metros @ velocidade máxima). Ao contrário do RS-485, o RS-232 não permite mais do que dois dispositivos ao mesmo tempo ligados ao mesmo barramento.

O RS-232 trabalha com tensões mais elevadas do que as do padrão TTL. No caso, o nível alto (5V) TTL é convertido em -12V e o nível baixo TTL (0V) em +12V. Na recepção, qualquer sinal entre +3V e +12V é considerado como o nível lógico baixo (0V para TTL), enquanto sinais entre -3V e -12V são considerados como nível lógico alto (5V para TTL). Caso o sinal recebido esteja entre -3V e +3V, ele será considerado como indefinido (Figura 2.8). Por se tratar de um padrão não diferencial, o sistema é altamente susceptível à interferência eletromagnética, prejudicando transmissõesa distâncias médias e grandes. Além disso, por trabalhar com uma faixa relativamente alta de tensões (-12V a +12V) é necessária a utilização de drivers

[6]O *driver* utilizado para o padrão RS-485 é o DS-485 da *National Semicondutor* e para o RS-232 o MAX232 da *Texas Instruments*.

[7]http://www.rs485.com/rs485spec.html

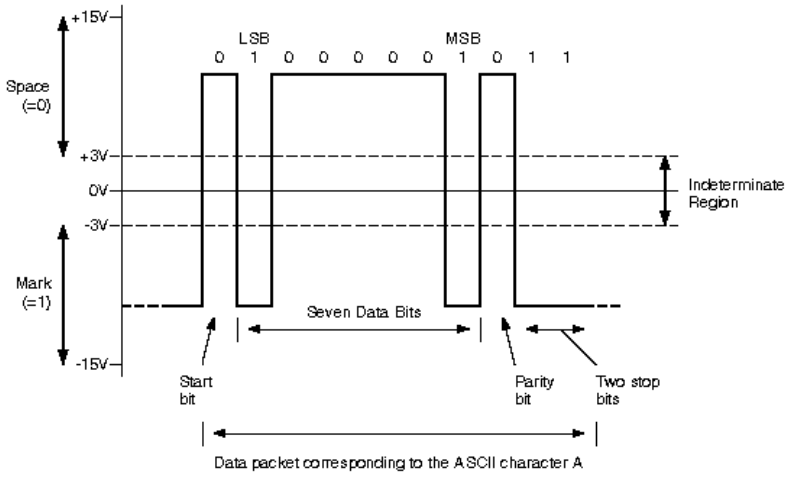


Figura 2.8: Ilustração gráfica das regiões de interpretação do sinal no padrão RS-232. Fonte: http://www.arcelect.com/rs232.htm

com alto *slewrate* para adquirir taxas elevadas de transmissão de dados.

O relógio externo escolhido para o ATmega8 (11,0592 Mhz) possui uma frequência especial. Aparentemente por ser um número "quebrado" é possível se pensar que é mais complicado de se trabalhar, porém isso não é verdade. Esse relógio possui uma frequência que é múltipla da frequência dos primeiros processadores construídos com a arquitetura IBM-PC. Historicamente todos os processadores desenvolvidos posteriormente com essa mesma arquiterura tem mantido a mesma lógica e portanto com frequências sempre múltiplas do primeiro. Isso influencia diretamente na comunicação serial, pois a comunicação entre dois dispositivos que possuem frequências não múltiplas gera uma perda de pacote de dados devido aos ciclos de relógio fora de sincronia, diminuindo a eficiência e até desconectando. Com o relógio de 8Mhz, por exemplo, essa perda chega a 8,5% para velocidades de 115200Kbps [9]. Essa é uma perda bastante significativa quando se trata de um sistema crítico e que necessita de uma alta velocidade de conexão. Para evitar problemas futuros, foi decidido instalar um relógio cuja frequência do IBM-PC seja múltiplo, portanto diminuindo as perdas para aproximadamente 0% na velocidade máxima do barramento serial.

O barramento RS-232 utilizado nesse projeto tem a finalidade de estabelecer a comunicação entre o IBM-PC e o circuito central. O circuito central faz a conversão do RS-232 para o RS-485, como mencionado anteriormente e ilustrado na Figura 2.6, porém isso não é a forma idealizada inicialmente para o projeto. Futuramente o IBM-PC será substituido por um sistema embarcado com o processador Geode de 500MHz. No sistema embarcado é possível montar um circuito que faça a comunicação diretamente com o barramento RS-485 e eliminando do sistema a limitada comunicação serial.

2.1.3.3 O protocolo de comunicação

Para todo sistema de comunicação entre múltiplos dispositivos, recomenda-se estabelecer um protocolo de comunicação. O protocolo tem como finalidade padronizar as mensagens que são enviadas/recebidas no barramento, evitando assim problemas de colisão de dados ou corrupção de mensagens, portanto aumen-

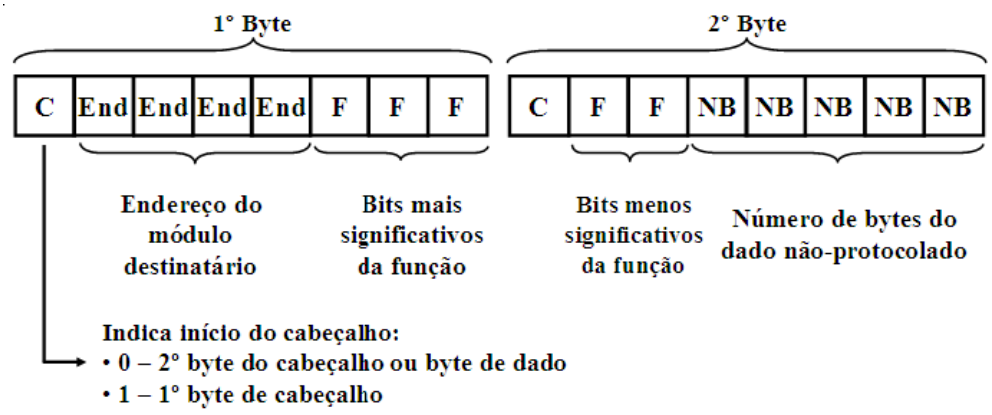


Figura 2.9: Composição dos bytes do cabeçalho do protocolo utilizado no projeto. Fonte: [2]

tando a performance da comunicação. Neste projeto foi estabelecido que o formato da mensagem seria composto de duas partes principais: o cabeçalho da mensagem e o pacote de dados.

O cabeçalho é criado para que toda a mensagem enviada/recebida tenha uma referência de início e fim da mensagem. Portanto é essencial que o mesmo possua os seguintes dados: tamanho do pacote de dados que vai ser enviado e endereço de destino da mensagem (no caso de múltiplos dispositivos). Outros dados podem ser adicionados ao cabeçalho para melhorar a comunicação. Nesse projeto a "função" a ser executada no dispositivo escravo também foi adicionada ao cabeçalho. Conforme formulado por Calmon et al[2], um cabeçalho de dois bytes é o suficiente para conter todos esses dados. A Figura 2.9 ilustra como esse cabeçalho é composto.

Com o cabeçalho presente em cada mensagem, é possível preparar o sistema receptor para a quantidade de dados que está por vir. Além disso é possível descartar as mensagens que não são destinadas ao disposito, economizando tempo de processamento naqueles que não estão sendo solicitados.

Apesar de ter sido formulado, o protocolo não foi inteiramente implementado nos projetos anteriores. Portanto fez-se uma padronização do sistema, ou seja, uma adequação de todos os módulos do robô quadrúpede ao "novo" protocolo. Dentre as modificações está a quantidade de bytes de dados e na maneira de se controlar os três servos de cada membro.

2.2 CONTROLE E CALIBRAÇÃO DO SISTEMA DE LOCOMOÇÃO

Com o protocolo e o barramento de comunicação estabelecido, torna-se também necessário o estabelecimento da maneira como os dados serão interpretados por cada sistema integrado ao barramento. Preferencialmente a quantidade de dados em uma mensagem de comando deve ser o menor possível, pois assim a comunicação se torna mais rápida e possibilita sistemas de controle mais robustos. A quantidade de *bytes* utilizado para a mensagem de comando vai estar diretamente ligado aos parâmetros dos dispositivos: como frequência de relógio, resolução do contador, faixa de sinal onde os servomotores são acionados e etc. Todos esses parâmetros serão detalhados ao longo dessa seção, enfatizando sua importância para a quantidade de dados utilizada na comunicação dos comandos.

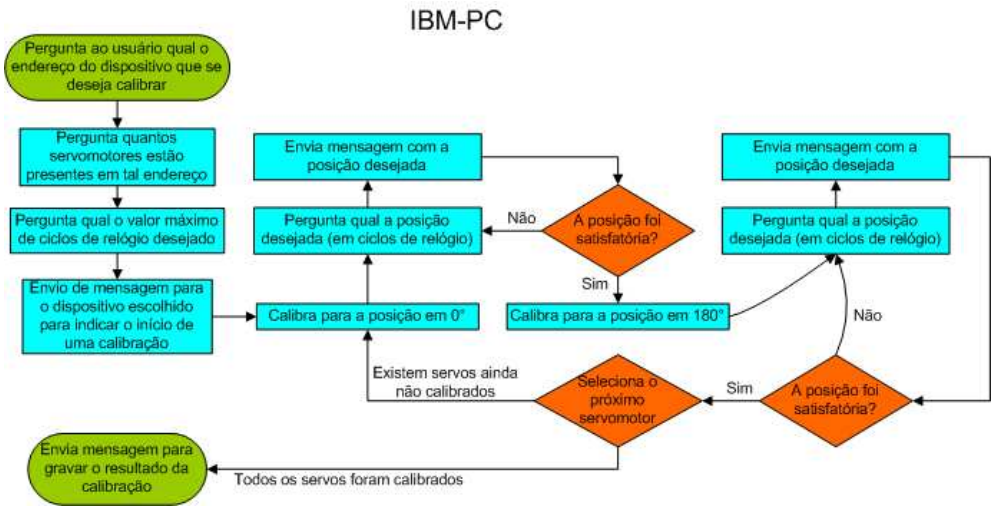


Figura 2.10: Fluxograma do programa de calibração utilizado no IBM-PC.

Ao se gerar os sinais de MLP percebeu-se que cada servomotor possuia um comportamento diferente com relação aos ciclos de trabalho a que eram submetidos. Para posicionar-se na referência em 0^o, por exemplo, alguns começavam a responder com $\tau = 4,5\%$, e outros com $\tau = 3,5\%$. Idealmente o sistema de controle (IBM-PC) ao controlar os servomotores não deveria se "preocupar" com esse tipo de diferença, ou seja, as mensagens enviadas deveriam ser indiferentes do destino e responder do mesmo modo. Para proporcionar isso, uma calibração do sistema foi feita necessária.

Finalmente após várias etapas de desenvolvimento e a calibração dos servomotores, é possível criar um sistema de controle dos servomotores da plataforma quadrúpede, que é um dos objetivos principais desse projeto. O algoritmo de controle é descrito na Seção 2.2.2, bem como a geração das sequências de movimento para que ele possa se movimentar da forma mais eficiente possível.

2.2.1 Sistema de calibração

Como mencionado na Seção 2.1.2.2, foi verificado que com o relógio em 11,0592Mhz e o divisor de escala de 64, eram necessários exatamente 432 ciclos de relógio para fornecer um período de 2,5ms em 50Hz, gerando um ciclo de trabalho de no máximo $\tau = 12,5\%$. O programa de calibração tem como objetivo verificar qual o ciclo de trabalho necessário para que cada servomotor se movimente entre 0^o e 180^o, de acordo com o referencial adotado. Para tal finalidade foi criado um programa que permite ao usuário selecionar o servomotor que deseja calibrar e o posicioná-lo com os valores de ciclos de relógio entre 0 e 432 (de $\tau = 0\%$ a $\tau = 12,5\%$). Esse programa foi desenvolvido no IBM-PC com o sistema operacional *SuSe Linux* versão 10.1. O fluxograma do programa de calibração implementado no IBM-PC e que será descrito nessa seção é ilustrado na Figura 2.10.

Os dados são enviados via serial e entram no barramento RS-485 através do circuito central (Figura 2.6). Cada ATMega8 possui um endereço dentro do barramento, portanto a mensagem enviada destina-se apenas a um dos microcontroladores. Ao receber a mensagem, o microcontrolador a interpreta e obtém a função a ser executada. As seguintes funções foram formuladas para o controle dos servomotores: função de calibração, função de movimento e função para a gravação dos resultados de calibração na EEPROM.

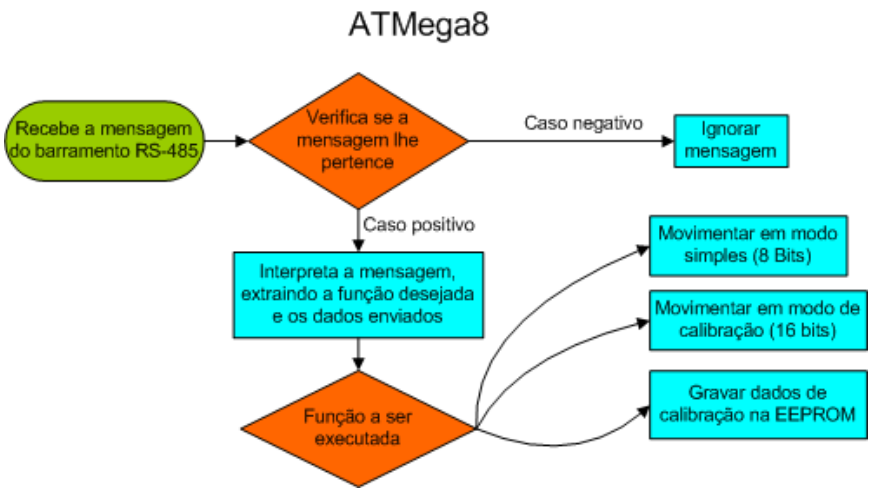


Figura 2.11: Fluxograma do programa de calibração implementado no ATMega8.

O fluxograma do algoritmo implementado no ATMega8 com essas funções é ilustrado na Figura 2.11.

O algoritmo implementado no IBM-PC interage com o usuário perguntando-lhe qual o endereço do dispositivo que se deseja calibrar (Figura 2.10. Ao especificar o endereço, é então solicitado a quantidade de servomotores presentes e qual o valor máximo do ciclo de relógio para o dispositivo. No caso específico desse projeto, utilizamos o valor de 432 ciclos de relógio, como o motivo já foi explicado nas seções anteriores. O programa não se restringe apenas à calibração dos servos desse projeto. Por exemplo, se o usuário desejar um ciclo de trabalho superior a 12,5%, ele poderá especificar um valor superior a 432 ciclos[8].

A calibração então se incia após a especificações dos parâmetros necessários. Quando existirem mais de um servomotor, o programa irá calibrar individualmente um por um. Como o processo de calibração se repete igualmente para cada servomotor, descreveremos aqui o processo geral. O programa indaga ao usuário qual o ciclo de relógio para que o servomotor se posicione na referência de 0^o. O usuário insere o valor desejado e então o programa o envia para o ATMega8. Ao chegar no microcontrolador, o mesmo o interpreta (após verificar se a mensagem lhe pertence) e extrai as informações da mensagem (Figura 2.11. Como os valores especificados pelo usuário podem ultrapassar a capacidade de 8 bits (ou 1 *byte*), decidiu-se que os comandos enviados para calibração sejam de 16 bits. Não é necessário mais do que isso, visto que o contador mais preciso do ATMega8 possui 16 bits.

A função de calibração interpreta o valor enviado e o utiliza para gerar o sinal de MLP e mover o servomotor para a respectiva posição. Se a posição não for satisfatória o usuário poderá especificar um novo valor até que se obtenha a posição desejada (o mais próximo de 0^o possível). Ao encontrar uma posição satisfatória para os 0^o, o usuário deverá iniciar a calibração para a posição máxima ou 180^o. Ao finalizar essa calibração, o usuário irá calibrar o próximo servomotor, até que todos estejam calibrados. Finalmente, ao final de tudo, o usuário confirma os dados de calibração e então uma mensagem é enviada ao ATMega8 indicando que os dados calibrados são válidos e podem ser gravados em sua memória EEPROM. Esse algoritmo de gravação é uma função específica do programa no ATMega8, como ilustrado no fluxograma da Figura 2.11.

[8]Valor calculado baseado nas condições especificadas para esse projeto, ou seja, mesma frequência de relógio, divisor de escala e etc.

A memória EEPROM do ATMega8 suporta o armazenamento de até 512 *bytes*. Como dois *bytes* são usados para armazenar cada posição da calibração, seriam necessários 4 *bytes* para cada servomotor: dois para a posição e referente aos 0^o e outros dois para os 180^o. Como num total são três servos por microcontrolador, apenas 12 *bytes* da EEPROM seriam ocupados com a calibração. Após a calibração de todos os servomotores do endereço, a calibração referente a esse microcontrolador é dita completa. O sistema que utiliza os valores calibrados para movimentar o servomotor é o sistema de controle e movimentação, explicado a seguir.

2.2.2 Sistema de controle

É interessante que, para o sistema de controle central (IBM-PC), as ordens de movimento não sejam dadas em valores de ciclos de trabalho, até porque o movimento em 180 graus ficaria limitado a valores entre 0 e 12,5%. Isso só seria possível com a utilização de variáveis do tipo *float*, mas é uma alternativa inviável, pois, além do ATMega8 não possuir processador matemático exclusivo e não estar preparado para trabalhar com esse tipo de variável, o envio de apenas um comando de movimento gastariam 4 *bytes*. A solução possível seria a relação direta entre a angulação desejada e o valor enviado, ou seja, a própria posição em graus seria enviada. Essa alternativa é bastante razoável, pois poder-se-ia trabalhar com variáveis do tipo *unsigned char* ou *unsigned short int*, ocupando apenas 1 *byte*, quatro vezes mais rápido do que a alternativa anterior. Porém, podemos aproveitar melhor a variável de 1 *byte* utilizando-a por completo, ou seja, aproveitando-se de todos os seus 256 valores, aumentando portanto a resolução de movimento de 1^o para $0,706^o$ para cada valor entre 0 e 255.

Utilizando-se de apenas um *byte* de dado para cada servo, podemos posicioná-lo em 0^o enviando o valor de 0 e em 180^o enviando o valor de 255. Como mencionado na Seção 2.2.1, exitem três funções interpretadas pelo microcontrolador com respeito aos dados recebidos: a de calibração(com dados de 16 bits), movimento(com dados de 8 bits) e gravação na EEPROM (Figura 2.11). A função de calibração e gravação já foram exclarecidas na seção anterior. Já a função de movimento tem como objetivo transformar esse *byte* recebido em valores correspondentes aos ciclos de relógio calibrados, de 0 a 432, que por sua vez geram os ciclos de trabalho para movimentar os servomotores.

Supondo que a posição mínima(0^o) de um servomotor calibrado seja P_{min}; a posição máxima(180^o) seja P_{max} e o *byte* correspondente ao seu movimento recebido via barramento, D_{mov}. A função de movimento utiliza esses dados para resolver as equações 2.2 e 2.3, e obter o valor D_{MLP}, que corresponde à quantidade de ciclos de relógio (dados em 16 bits) para se gerar o sinal de MLP com o ciclo de trabalho desejado.

A estrutura do robô quadrúpede foi montada de tal maneira que os servos da direita estão invertidos com relação aos da esquerda, ou seja, o movimento para a referência de 0^o para um, corresponde o oposto no outro, movendo-se para 180^o. Seria trabalhoso levar isso em consideração toda vez que se comandasse o robô. Para contornar tal problema, fizeram-se as equações de transformação dependentes do endereço do microcontrolador. A equação 2.2 corresponde ao endereço das patas presentes na esquerda do robô, enquanto a equação 2.3 aos da direita. Direita e Esquerda é relativo sem definir a frente ou trás do robô, porém o importante é que o *byte* utilizado em uma das equações sejam antagônico ao outro. Desta forma, independente do referêncial adotado, todos os membros do robô irão responder igualmente aos mesmos comandos.

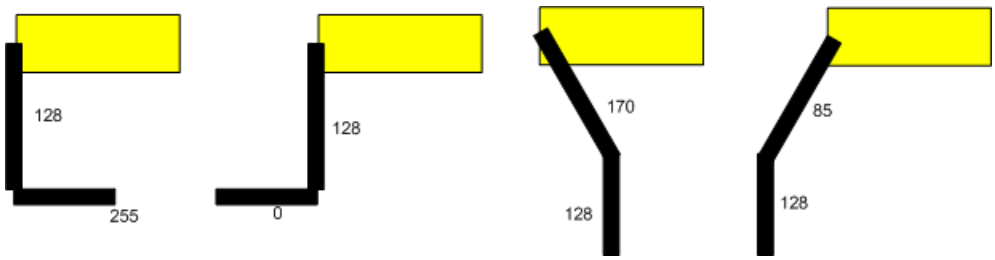


Figura 2.12: Exemplo bidimensional de posicionamento dos servomotores de uma pata, com seus respectivos valores de posição entre 0 e 255.

$$D_{MLP} = \frac{((P_{max} - P_{min}) * (255 - D_{mov}))}{255} + P_{min} \tag{2.2}$$

$$D_{MLP} = \frac{((P_{max} - P_{min}) * D_{mov})}{255} + P_{min} \tag{2.3}$$

Com o valor de D_{MLP} obtido para cada servo, basta utilizá-lo para atualizar o vetor $s(i)$ e consequentemente valor do comparador A, como mostra o fluxograma na Figura 2.5 da Seção 2.1.2.2. Assim a posição do servomotor no próximo período de 20ms irá se atualizar.

2.2.2.1 Sequênciamento de movimentos

Com o sistema calibrado e com o algoritmo de movimento implementado, inicia-se a fase de teste para o caminhar do quadrúpede. A geração do movimento artificial do quadrúpede tem dois problemas principais: qual é a sequência de posições dos servomotores durante o caminhar (dados enviados via serial com a posiçao de cada servomotor) e qual é a sincronia dos membros durante esse movimento. Por exemplo: não adianta nada os servomotores se moverem de maneira a proporcionar um movimento balístico perfeito se todas as patas iniciam o movimento ao mesmo tempo. Deve existir uma harmonia de entre cada membro do quadrúpede de forma a desenvolver um caminhar. Essa sequência se altera de acordo com a velocidade e a direção de deslocamento. Um dos objetivos desse projeto é o desenvolvimento de uma ordem de ativação das patas de modo que faça o quadrúpede se movimentar para frente e em linha reta da maneira mais rápida possível de forma a não se danificar.

Um exemplo de posições dos servomotores é ilustrado na Figura 2.12. Nessa Figura um dos graus de movimento está oculto, pois se trata de uma representação bidimensional.

Diversas pesquisas já foram feitas para esses tipos de problema em quandrúpedes: existe como se estimar matematicamente o movimento de cada membro [6]; como se obter a sequência de movimentos através da mimica de animais[11]; através de aprendizagem computacional (tanto em movimento de cada membro quanto na sequência de movimentos); ou até mesmo empíricamente por tentativa e erro. Para uma estrutura mecânica, a tentativa e erro pode ser desastrosa, pois o robô pode vir a se quebrar. Já as demais alternativas podem ser exploradas nesse trabalho, ou até mesmo a fusão de algumas delas.

No projeto desenvolvido por Souto [6], em paralelo a esse no LARA, foram geradas sequências de posições, de 0 a 255, para cada servomotor de cada membro, com o finalidade de realizar um movimento balístico. O modelo foi baseado na mesma plataforma mecânica utilizada nesse projeto. Por causa dessa

compatibilidade vários dados puderam ser trocados com o intuito de enriquecer ambos os resultados. O trabalho [6] utilizava modelos matemáticos que não levam em conta a ação da gravidade e portanto precisava de uma validação empírica. Já esse trabalho precisava de um ponto de partida para o aprimoramento do movimento. Diversas tentativas foram feitas até se encontrar uma sequência de movimento aceitável. Algumas dessas tentativas serão comentadas durante o capítulo de resultados experimentais (Capítulo 3).

Ao determinar uma sequência de movimentos aceitável, basta descobrir uma ordem de acionamento de cada pata para que o robô se movimente. A sequência de movimento gerada poderia ser idêntica para cada pata, já que o movimento é uniforme e o quadrúpede é simétrico. O movimento balístico em si não garante o movimento do robô, pois se as patas estiverem se movimentando na ordem errada, ele simplesmente não sairá do lugar ou até andará para trás. A sequência em que cada pata irá se mover é algo muito importante e pode ser estimada através de algoritmos de aprendizagem computacional. Alguns desses algoritmos são discutidos na Seção 2.5.

O programa que envia a sequência de movimentos para os servomotores também foi desenvolvido no sistema operacional *Linux*. Esse *software* é basicamente um algoritmo de repetição que lê uma sequência de posições geradas e as envia para cada servo do quadrúpede (isso se repete indefinidamente até que se pare o programa). O envio deve ser o mais rápido possível e de maneira que todos os servos recebam o seu próximo valor antes do fim do período de atualização, 20ms. Para averiguar se a velocidade de transmissão da porta serial é capaz de cumprir tal tarefa, deve-se estimar a quantidade de *bytes* necessários para se transmitir a posição de todos os servomotores.

A uma velocidade de 115200Bps é possivel se transmitir 14400 *bytes* a cada segundo (ou 288 bytes a cada 20ms), e a quantidade de dados para se atualizar a posição de um servomotor é: 1 *byte* para dado e 2 *bytes* para o cabeçalho, totalizando 3 *bytes*. Como são 12 servomotores: $3 * 12 = 36$ *bytes*. Portanto verifica-se que a velocidade de transmissão serial é de $\frac{288}{36} = 8$ vezes mais rápida do que o necessário. Se atrasos não forem inseridos no algoritmo do programa para sincronizar o envio de dados, o comparador responsável pelo servomotor irá se atualizar 8 vezes antes que o mesmo “perceba” a mudança, desperdiçando posições que podem ser excenciais ao movimento do quadrúpede. Os atrasos foram determinados empiricamente e implementados, mas podem ser ajustados de acordo com a necessidade do usuário.

2.3 SISTEMA DE SENSORIAMENTO

2.3.1 Introdução

O sistema de sensoriamento do robô-quadrúpede tem como objetivo estimar grandezas que possibilitem a avaliação do movimento do robô. Diversas são as grandezas que podem ser utilizadas com essa função. Uma delas é certamente o deslocamento do robô durante um movimento. Com essa grandeza, pode-se qualificar a eficiência das seqüências de passos utilizadas. Para a direção esperada em um movimento, são considerados passos ineficientes aqueles que implicam pequenos deslocamentos e passos eficientes aqueles que implicam grandes deslocamentos. Já para as direções inesperadas, considerações opostas às anteriores são feitas. Desse modo, a grandeza deslocamento, denominada posição - considera-se a posição inicial zero -, será uma das grandezas utilizadas para a avaliação. Dois eixos, *x* e *y*, serão tomados como direção esperada e inesperada do movimento respectivamente. Esses eixos coincidem com os correspondentes em

um sensor de aceleração. Tais sensores são descritos na seção 2.3.2.

Além do deslocamento, será utilizada para a avaliação de movimentos a grandeza inclinação. Por meio dela, será possível perceber o quanto que o robô trepita ao realizar dado deslocamento. Portanto, tal grandeza é capaz de verificar a suavidade dos passos utilizados em cada movimento. Essa grandeza será obtida a partir de medições realizadas sobre o eixo de atuação da gravidade, definido como o eixo z. Esse eixo também coincide com o correspondente em um sensor de aceleração.

Com o objetivo de obter as medições necessárias à estimativa das grandezas supracitadas, uma arquitetura eletrônica foi proposta para o sistema de sensoriamento do robô-quadrúpede. Essa arquitetura é composta por um microcontrolador ATMega8 da ATMEL, por um sensor de aceleração da Spark Fun Eletronics, por um transceptor DS-485 da National Semiconductor e por dois reguladores de tensão, um 7805 e um LM338K. O 7805 gera a tensão de alimentação do ATMega8 e do DS-485, cujo valor é de $5V$. O LM338K, por sua vez, gera a tensão de alimentação do sensor de aceleração, cujo valor escolhido é $3,3V$.

Esse projeto não objetiva explicar o funcionamento desses reguladores de tensão. Os detalhes sobre como obter as tensões supracitadas podem ser encontrados nos datasheets desses componentes [9] [10]. Por sua vez, descrição aprofundada do funcionamento do ATMega8 pode ser encontrada em seu manual[9].

O sensor de aceleração será descrito em seção própria (seção 2.3.2) , visto ser o componente-chave dessa arquitetura. O ATMega8, por sua vez, terá apenas os aspectos de funcionamento relacionados a sua integração com o sensor de aceleração discutidos. Essa discussão será feita durante a descrição do tipos de sensor de aceleração.

2.3.2 Sensor de aceleração

O sensor de aceleração utilizado é um acelerômetro de três eixos que obtém informações de um elemento sensor e as transmite ao mundo exterior através de uma interface de comunicação. Freqüentemente essa interface é o elemento que diferencia dois acelerômetros.

Durante a implementação do sistema de sensoriamento, dois modelos de interface foram utilizados. O primeiro é uma interface analógica; o segundo, uma interface IC. Relacionados a esses modelos de interface, foram empregados os acelerômetros MMA7260Q e LIS3LV02Q, respectivamente. Ambos sensores são produzidos pela Spark Fun Eletronics. Os detalhes pertinentes ao funcionamento desses componentes podem ser encontrados no *site* do fabricante [11].

As duas seções seguintes detalharão as características desses sensores de aceleração. Cabe ressaltar que a utilização desses sensores não foi simultânea. Fez-se necessária somente por problemas técnicos ocorridos durante a execução desse projeto.

Os acelerômetros MMA7260Q e LIS3LV02Q são apresentados soldados às suas respectivas placas na Figura 2.13.

2.13.

[9] http://www.datasheetcatalog.com/datasheets_pdf/7/8/0/5/7805.shtml
[10] http://www.datasheetcatalog.com/datasheets_pdf/L/M/3/3/LM338.shtml
[11] http://www.sparkfun.com

(a) mma7260Qbreakout (b) lis3lv02dqbreakout

Figura 2.13: Acelerômetros da Spak Fun Eletronics empregados no sistema de sensoriamento

2.3.3 Acelerômetro MMA7260Q

Inicialmente, utilizou-se o acelerômetro MMA7260Q (Figura 2.13 (a)). Esse sensor é um acelerômetro de três eixos que possui quatro seleções para o alcance da escala: $\pm$1,5g; $\pm$2g; $\pm$4g e $\pm$6g, em que g é a aceleração da gravidade. Para ajustar tais alcances, valores de tensão são aplicados aos pinos g-Select1 e g-Select2 do acelerômetro. Para cada combinação desses valores, diferentes sensibilidades são selecionadas. Os alcances e as sensibilidades correspondentes a essas combinações de tensão são colocados na Tabela 2.1. O alcance e a sensibilidade escolhidos foram $\pm$1,5g e 800mV/g, respectivamente.

Tabela 2.1: Descrição da sensibilidade do MMA7260Q em função de g-Select

g-Select2	g-Select1	alcance de escala	sensibilidade
0	0	$\pm$1,5g	800mV/g
0	1	$\pm$2g	600mV/g
1	0	$\pm$4g	300mV/g
1	1	$\pm$6g	200mV/g

O MMA7260Q deve ser suprido por uma tensão entre $2,2V$ e $2,6V$. A sua tensão de suprimento típica é $3,3V$. Quando suprido a $0g$ por essa tensão , um *offset* $1,65V$ é obtido nos pinos X_{out}, o Y_{out} e o Z_{out}. Tal *offset* será utilizado na obtenção da aceleração em cada eixo. A equação 2.4 relaciona, respectivamente, a tensão dos pinos X_{out}, Y_{out} ou Z_{out} às acelerações nos eixos x, y e z. Nessa equação V_{out} corresponde à tensão presente em um desses pinos, enquanto $Sens$ corresponde à sensibilidade selecionada. Por sua vez, a é a aceleração desenvolvida pelo quadrúpede no repectivo eixo.

$$a = \frac{(V_{out} - V_{ref})g}{Sens}. \tag{2.4}$$

Além dos pinos acima descritos, o MMA7260Q possui ainda o pino *Sleep Mode*. Seu acionamento diminui a potência consumida pelo acelerômetro, fazendo-o consumir uma corrente típica de 3μA . O circuito de recepção pelo ATMega8 da aceleração medida pelo MMA7260Q é mostrado na Figura 2.14.

Os pinos X_{out}, Y_{out} e Z_{out} são conectados a filtros passa-baixas, sendo dois para cada pino. Cada um desses filtros é construído pela conexão de um resistor a um capacitor e desse capacitor ao terra. Dois são os tipos de filtro implementados, sendo que todo pino é ligado a filtros desses dois diferentes tipos. O filtro é sempre obtido tomando-se a tensão entre o capacitor e o resistor.

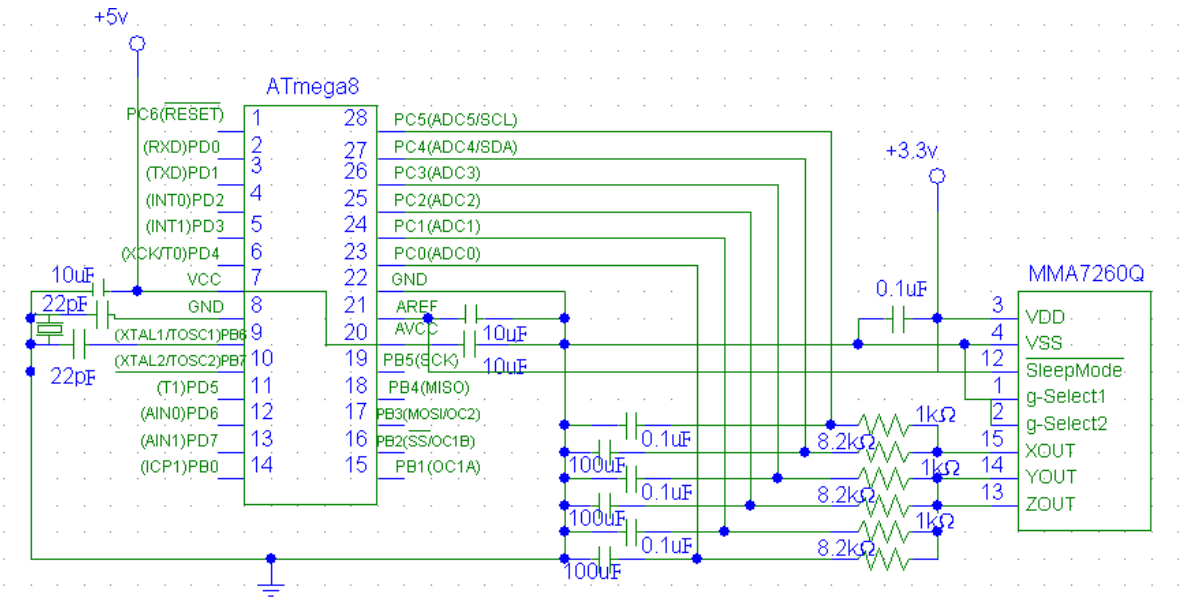


Figura 2.14: Circuito de recepção pelo ATMega8 da aceleração medida pelo MMA7260Q

Um desses tipos de filtro é construído, utilizando-se um resistor de 1,0kΩ e um capacitor de 0,1μF. Tal filtro tem como objetivo reduzir o ruído no relógio do acelerômetro. O outro tipo de filtro, porém, é construído, utilizando-se um resistor de 8,2kΩ e um capacitor de 100μF. Esse filtro tem como objetivo a obtenção do valor médio da saída a que está conectado: X_{out}, Y_{out} ou Z_{out}. As freqüências desses filtros são aproximadamente 1,6kHz e 0,2Hz, obedecida a ordem de citação.

A saída desses filtros é conectada aos pinos de conversão A/D(Analógico-Digital) do microcontrolador. Esses pinos, que pertencem à porta C do microcontrolador,são os pinos de AD0 a AD5. Os valores presentes nesses pinos são coletados a intervalos de tempo constante. Ou seja, o microcontrolador amostra os valores desses pinos, que são sinais analógicos de tensão.

As amostras obtidas são convertidas em um valor digital (ADC) entre 0 e 1023. Tal valor é obtido pela comparação da tensão V_{in} presente no pino do conversor A/D com uma tensão de referência V_{ref} escolhida ao se programar o ATMega8.

A equação abaixo mostra como obter o valor digital da aceleração:

$$ADC = 1024\frac{V_{in}}{V_{ref}} \tag{2.5}$$

Para a arquitetura em questão, uma tensão de referência de 3,3V foi escolhida, visto ser a maior tensão alcançada pelo MMA7260Q ao medir uma aceleração. Para tal, programou-se V_{REF} como a tensão presente no pino A_{REF} e conectou-se A_{REF} a 3,3V.

Os pinos g-Select1 e g-Select2 do MMA7260Q, por sua vez, estão conectados à referência do circuito. Tais conexões são feitas para a operação a uma sensibilidade de 800mV/g. Assim, obtém-se maior precisão para os valores de aceleração obtidos do acelerômetro. Essa sensibilidade se torna possível pelos valores de aceleração obtidos em cada eixo pelo quadrúpede não superarem $\pm$ 1,5g .

Já o pino V_{CC} é conectado aos 3,3V gerados por um regulador de tensão LM338K. Reguladores do

mesmo tipo são utilizados para alimentar os servomotores empregados nesse projeto.

2.3.4 Descrição da arquitetura eletrônica

A arquitetura eletrônica para o acelerômetro MMA7260Q é mostrada em diagrama de blocos na Figura2.15. Dessa arquitetura serão destacadas algumas interfaces devido a sua importância.

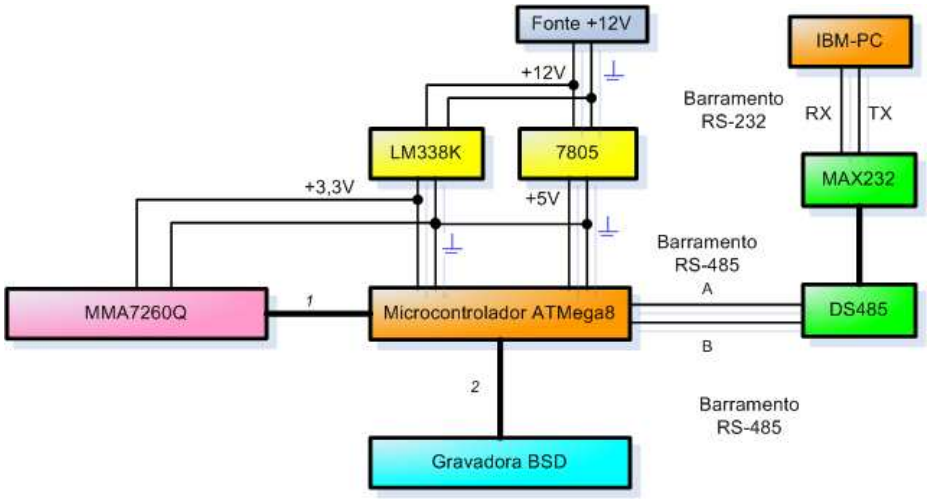


Figura 2.15: Diagrama de blocos do sistema de sensoriamento para o MMA7260Q

A interface *1* é aquela pela qual o microcontrolador ATMega8 obtém amostras de aceleração do MMA7260Q . Nessa interface, são obtidas tanto amostras que estimam o valor médio da aceleração quanto amostras que estimam seu valor instantâneo. A estimativa do valor médio da aceleração visava a previsão do *offset* de tensão correspondente à aceleração da gravidade. Entretanto, sabe-se que durante o movimento do robô a estimativa desse valor médio não é uma estimativa adequada da aceleração da gravidade (ver explicação na seção 3.3.1). A calibração do MMA7260Q pode vir a ser uma aplicação futura do valor médio obtido. Entretanto,atualmente utiliza-se outro algoritmo (ver seção 2.4.3).

Cabe ressaltar que tanto o valor médio quanto o valor instantâneo de aceleração medidos pelo MMA7260Q devem anteriormente digitalizados no conversor A/D do ATMega8. A equação 2.5 realiza essa conversão.

Quanto ao valor instantâneo de aceleração, ele é utilizado tanto no algoritmo de posicionamento do robô, descrito durante a seção 2.4, quanto no algoritmo do inclinômetro, descrito durante a seção 2.4.7.

A interface *2*, por sua vez, é aquela pela qual se programa o microcontrolador ATMega8. Tal programação é realizada através de uma Gravadora BSD. A descrição de como implementar e conectar tal gravadora ao ATMega8 não está no escopo desse projeto. Para tal, consulte a referência [12]. Já as interfaces RS-485 e RS-232 foram descritas nas Seções 2.1.3.1 e 2.1.3.2, respectivamente.

O diagrama esquemático referente ao diagrama de blocos apresentado nesta seção é mostrado na figura I.1 presente no Anexo desse projeto.

2.3.5 Acelerômetro LIS3LV02Q

Em um segundo momento, utilizamos o sensor de aceleração LIS3LV02Q (Figura 2.13 (b)). A utilização desse sensor aconteceu após a danificação do acelerômetro MMA7260Q utilizado no sistema de sensoriamento. Tal danificação deveu-se à aplicação de uma sobretensão de 8V ao pino V_{DD} desse acelerômetro. Como fez-se necessário adquirir outro acelerômetro, optou-se por um acelerômetro com saída digital e interface serial. Tal escolha visava a obtenção de um acelerômetro que proporcionasse uma arquitetura de implementação mais compacta e que possuísse maior flexibilidade quanto às configurações. Escolheu-se assim o acelerômetro LIS3LV02Q.

Esse sensor apresenta uma saída linear digital que utiliza a interface IC para a transmissão de dados em modo I^2C/SPI. Ele também apresenta um alcance de escala com duas possíveis seleções: $\pm$ 2g e $\pm$ 6g. A sua capacidade de medir a aceleração se dá em uma largura de banda de 640Hz para qualquer um de seus eixos. O valor dessa largura de banda é ajustável pelo usuário.

O circuito de recepção pelo ATMega8 da aceleração medida pelo LIS3LV02Q é mostrado na Figura 2.16. Nele, cada um dos pinos CS, SPC, SDI e SDO é conectado a um resistor de 1kΩ. Esses, por sua vez, são conectados, respectivamente, aos pinos PC5,PC4,PC3 e PC2 de um ATMega8.

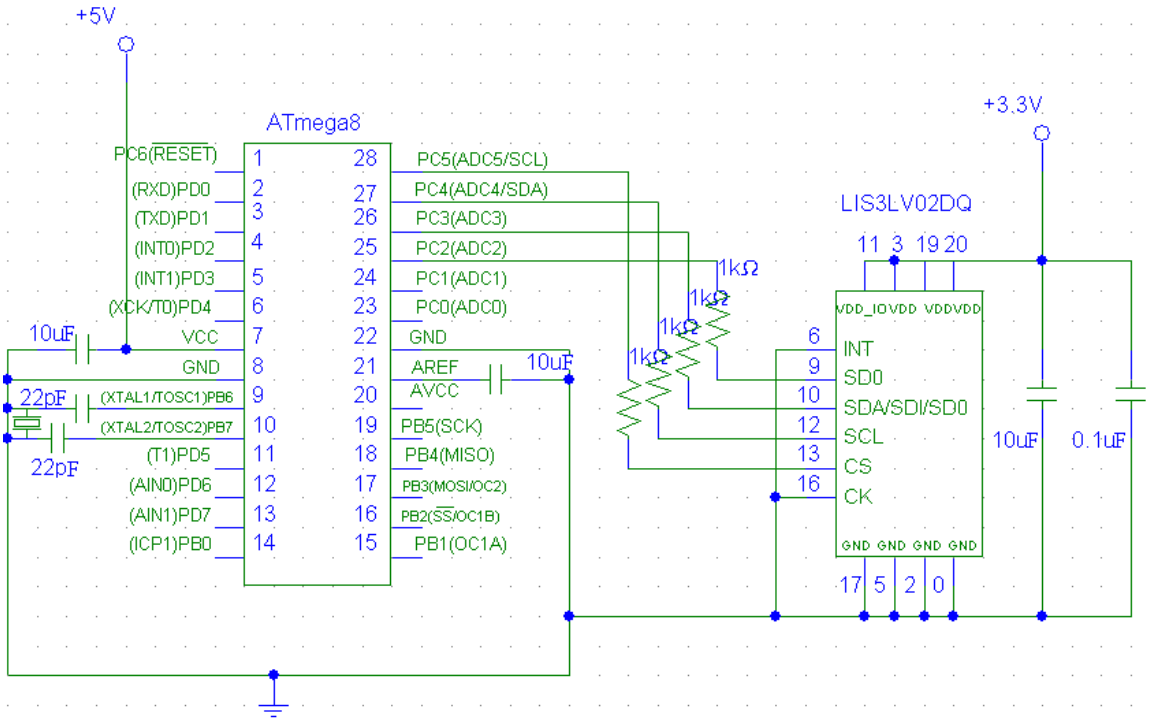


Figura 2.16: Circuito de recepção pelo ATMega8 da aceleração medida pelo LIS3LV02Q

Além desses pinos, existem os pinos V_{DD}, GND, CK e INT. O pino GND é conectado ao terra do sistema. O pino V_{DD}, por sua vez, deve ser conectado a uma tensão de suprimento entre $2,16V$ e $3,6V$, sendo que, nesse projeto, seu valor é 3,3V . O pino CK corresponde à entrada do relógio externo; o INT, à entrada de um sinal interrupção. Como ambos não são necessários, fez-se suas conexões ao terra.

O LIS3LV02Q contém vários registradores. Neles, armazenam-se tanto as configurações do modo de funcionamento, quanto os valores de aceleração obtidos para cada eixo. Para acessar esses registradores,

comandos de leitura ou de escrita de dados são necessários. Para acessá-los, comandos de leitura ou de escrita de dados são necessários. Tais comandos são enviados em uma das duas interfaces de comunicação do acelerômetro. A interface SPI é a interface utilizada neste projeto.

Antes de se deter à explicação desta interface, torna-se necessário o entendimento dos comandos supracitados .

Os comandos de leitura de dados são os comandos empregados na obtenção dos dados contidos no acelerômetro. Esses comandos enviam um sinal de informação que contém o endereço do registrador que armazena o dado requisitado. Os comandos de escrita, por sua vez, são os comandos empregados na atualização das configurações do acelerômetro. Tais comandos enviam um sinal de informação que contém, além do endereço do registrador que armazena a configuração, o dado de configuração a ser nele colocado.

2.3.5.1 Protocolo SPI

A comunicação entre o acelerômetro e o meio exterior via interface SPI se dá a quatro fios. Utilizam-se, para tal, os pinos CS, SPC, SDI e SDO do acelerômetro. Nessa comunicação, o acelerômetro atua como escravo, apenas correspondendo às solicitações do microcontrolador ATMega8, componente mestre da arquitetura.

A descrição dos pinos de comunicação anteriormente citados será feita com a explicação de suas atuações durante uma comunicação. A atuação dos pinos CS e SPC será descrita inicialmente, pois, tanto para a leitura quanto para a escrita dos dados, idênticos são os sinais aplicados a esses pinos. Serão explicados, em seguida, os pinos SDI e SDO. Primeiramente, dentro de uma comunicação de leitura de dados. Em seguida, dentro de uma comunicação de escrita de dados. As Figuras 2.17 e 2.18 ilustram os sinais presentes em todos esses pinos durante uma comunicação serial SPI de leitura e de escrita, respectivamente.

O pino CS é o pino que indica tanto o início quanto o fim de uma comunicação. Quando seu valor é resetado, a comunicação serial inicia, habilitando o sinal no pino SPC. Quando ele tem seu valor novamente setado, a comunicação termina, desabilitando o sinal no pino SPC.

O pino SPC é o pino do relógio utilizado na comunicação. Nele, 16 pulsos de relógio são aplicados a cada comunicação. O seu valor inicial é sempre em nível alto, sendo que os dados são capturados na transição de subida dos pulsos de relógio e transmitidos na transição de descida desses pulsos.

O pino SDI é sempre um pino de entrada para a interface SPI a 4 fios. Em uma comunicação de leitura de dados, ele captura dois *bytes* : o primeiro com informação, o segundo sem. O primeiro *byte* contém em seus 6 *bits* menos significativos o endereço do registrador cujo dado será obtido. Em seu bit mais significativo, ele contém o valor lógico 1, indicativo do modo leitura de comunicação. Já, no bit remanescente, o valor 1 ou 0, indicando ou não o incremento automático de endereços no modo de múltiplas leituras. O segundo *byte* é um *byte* irrelevante, apenas preenche o tempo necessário ao fim da comunicação.

O pino SDO, por sua vez, é sempre um pino de saída para interafce SPI a 4 fios. Em uma comunicação de leitura de dados, ele transmite dois *bytes* : o primeiro sem informação, o segundo com. O primeiro *byte* apenas preenche o tempo necessário à obtenção do *byte* de informação presente em SDI. Já o segundo

transmite ao mestre o dado armazenado no registrador escolhido. Esse registrador é aquele cujo endereço são os 6 *bits* menos significativos do primeiro *byte* em SDI.

Em uma comunicação de escrita de dados, o pino SDI recebe um sinal de entrada com dois *bytes* de informação. O primeiro desses *bytes* , a não ser pelo *bit* mais significativo, apresenta a mesma composição que em uma leitura de dados. Tal *bit* possui o valor lógico 0, indicativo do modo escrita de comunicação. Já o segundo transmite ao mestre o dado armazenado no registrador escolhido. Esse registrador é aquele cujo endereço são os 6 *bits* menos significativos do primeiro *byte* em SDI.

O pino SDO não envia nenhum *byte* de informação nesse tipo de comunicação. O sinal nele transmitido é irrelevante.

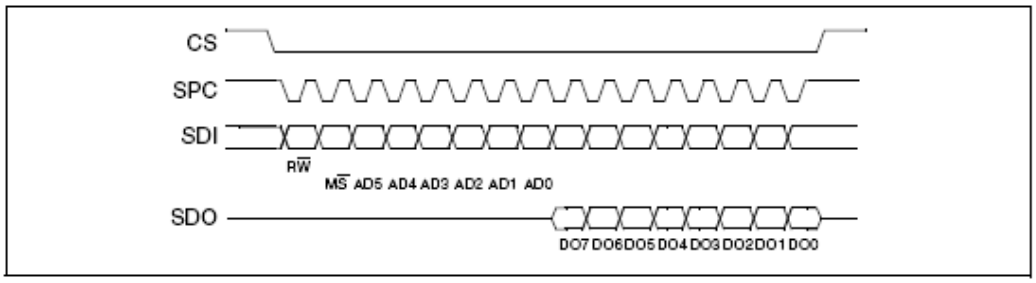


Figura 2.17: Protocolo SPI para a leitura de dados de um registrador

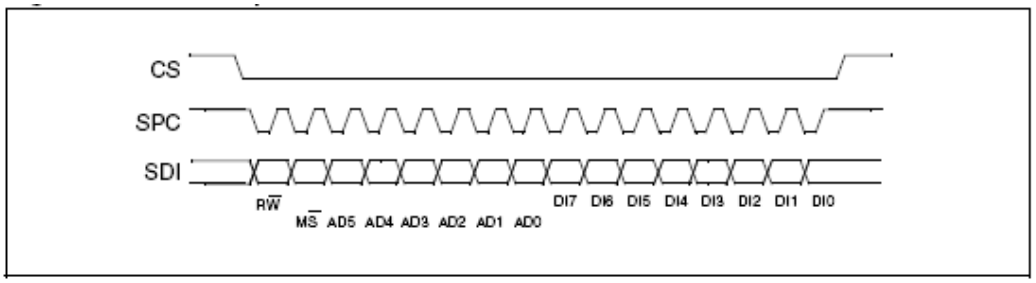


Figura 2.18: Protocolo SPI para a escrita de dados em um registrador

2.3.6 Registradores

Vários são os registradores existentes em um LIS3LV02Q. Dentre eles, 9 serão citados pela sua importante utilização nesse projeto.

O primeiro registrador, WHO_AM_I (endereço $0X0F$, número 0F em base hexadecimal), é o identificador do LIS3LV02Q, que contém o valor $0X3A$. Esse registrador é bastante útil no teste da comunicação serial. Tal teste é feito pela leitura do dado nele armazenado. O recebimento de um dado $0X3A$ indica um bom funcionamento da comunicação implementada.

O segundo é o registrador CTRL_REG1. Tal registrador contém as principais configurações do acelerômetro. Dentre elas, podem ser citadas o controle do consumo de potência (PD1,PD0), a escolha da taxa de amostragem para a obtenção dados de aceleração (DF1,DF0), a habilitação do modo teste para o acelerômetro (ST) e a habilitação dos canais de medida nos eixos X (Xen), Y (Yen) e Z (Zen). Os termos entre parênteses representam os *bits* correspondentes às configurações do registrador CTRL_REG1, dipos

do mais significativo para o menos.

A configuração escolhida para esse registrador corresponde à escrita do *byte* $0XF7$. Para tal valor, o dispositivo encontra-se ligado, com taxa de amostragem igual a 2560Hz, modo teste desligado e eixos habilitados.

O terceiro é o registrador CTRL_REG2. Tal registrador contém, dentre outras, as configurações de alcance da escala (FS), de habilitação de interrupções (IEN), da interface SPI utilizada (SIM) e do controle de reinicalização sobre o conteúdo da memória (BOOT).

A Tabela 2.2 apresenta os valores de sensibilidade correspondentes à seleção do alcance de escala por meio de FS. A configuração escolhida nesse projeto para esse registrador corresponde a um alcance da escala de 2-g ($FS = 0$). A sensibilidade correspondente à essa configuração é 1024LSb/g, em que g é a aceleração da gravidade e 1 LSb é 1g/1024 em 12 *bits* de representação a uma escala de 2-g.

Tabela 2.2: Descrição da sensibilidade do LIS3LV02Q em função de g-Select

FS	alcance de escala	sensibilidade
0	$\pm$2g	1024LSb/g
1	$\pm$6g	340LSb/g

Outras configurações empregadas para o registrador CTRL_REG2 são a desabilitação das interrupções do acelerômetro ($IEN = 0$) e o uso de uma interface SPI a 4 fios ($SIM = 0$). Quanto ao *bit* BOOT, ele é setado inicialmente a fim de se retomar as configurações iniciais todos os registradores.

Os outros seis registradores são o OUTX_L ($0X28$), o OUTX_H ($0X29$), o OUTY_L ($0X2A$), o OUTY_H ($0X2B$), o OUTZ_L ($0X2C$) e o OUTZ_H ($0X2D$). Eles armazenam os valores de aceleração obtidos pelo elemento sensor. A aceleração obtida é armazenada nesses registradores em 12 *bits* a complemento de 2. Os registradores cujo nome termina em "_L" armazenam os 8 *bits* menos significativos de aceleração. Já os registradores cujo nome termina em "_H" armazenam os 4 *bits* mais significativos da aceleração em seus 4 *bits* menos significativos. Os *bits* remanescentes apenas repetem o *bit* mais significativo da aceleração.

2.3.7 Descrição da arquitetura eletrônica

A arquitetura eletrônica para o LIS3LV02Q é mostrada em diagrama de blocos na Figura2.19. Dessa arquitetura serão destacadas algumas interfaces devido a sua importância.

A interface *1* é aquela pela qual o microcontrolador ATMega8 obtém amostras de aceleração do LIS3LV02Q . Nessa interface, são obtidas as amostras que estimam o valor instantâneo da aceleração. Esse valor é utilizado tanto no algoritmo de posicionamento do robô, quanto no algoritmo do inclinômetro, descritos durante a seção 2.4. O protocolo utilizado nessa comunicação é o protocolo SPI, cuja explicação é colocada na seção 2.3.5.1

A interface *2*, por sua vez, é aquela pela qual se programa o microcontrolador ATMega8. Tal programação é realizada através de uma Gravadora BSD. A descrição de como implementar e conectar tal

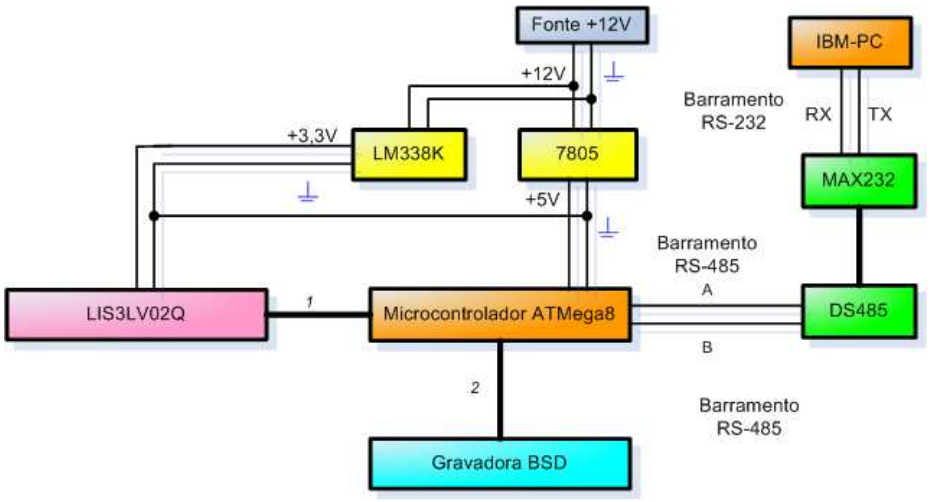


Figura 2.19: Diagrama de blocos do sistema de sensoriamento para o LIS3LV02Q

gravadora ao ATMega8 não está no escopo desse projeto. Para tal, consulte a referência []. Já as interfaces RS-485 e RS-232 foram descritas nas Seções 2.1.3.1 e 2.1.3.2, respectivamente.

O diagrama esquemático referente ao diagrama de blocos apresentado nesta seção é mostrado na figura I.2 presente no Anexo desse projeto.

2.4 ALGORITMOS DE PROCESSAMENTO DAS LEITURAS DO ACELERÔMETRO

2.4.1 Introdução

Os algoritmos implementados nessa seção baseiam-se em duas notas técnicas da *Freescale Semiconductor*: o *Implementing Positioning Algorithms Using Accelerometers*[3] e o *Measuring Tilt with Low-g Accelerometers*[13]. Tais artigos versam sobre a obtenção da posição e da inclinação de um objeto através do uso de um acelerômetro.

Os algoritmos descritos nessa seção são muito úteis à qualificação dos movimentos feitos pelo robô quadrúpede. Por meio deles, estimam-se a aceleração, a velocidade, a posição e a inclinação do robô em certos instantes de tempo. A obtenção desses valores é feita pelo processamento da aceleração proveniente de um acelerômetro. Em nosso projeto, o microcontrolador ATMega8 desempenha essa função.

2.4.2 Teoria e algoritmo de integração

No microcontrolador ATMega8, a aceleração, integrada duas vezes no tempo, fornece o deslocamento realizado pelo robô.

Com o objetivo de integrar duplamente a aceleração no tempo, uma integração simples deve ser feita duas vezes. A velocidade é obtida pela primeira dessas integrações. Para melhor entendimento, o algoritmo de integração é revisado na seção 2.4.2.

A derivada de uma varíavel corresponde à obtenção de sua taxa de variação. Desse modo, a velocidade

(v) é a derivada da posição (s) e a aceleração (a) é a derivada da velocidade:

$$a = \frac{d\vec{v}}{dt} \;\; e \;\; v = \frac{d\vec{s}}{dt} \therefore a = \frac{d(d\vec{s})}{dt^2}; \tag{2.6}$$

A integração é a operação matemática inversa da derivação, logo:

$$v = \int (\vec{a})dt \;\; e \;\; s = \int (\vec{v})dt \therefore s = \int \left(\int (\vec{a})dt \right) dt \tag{2.7}$$

Tal operação pode ser entendida graficamente como a área entre o eixo de integração e a curva da grandeza a ser integrada. Se a área estiver acima do eixo de integração, será considerada positiva; caso contrário, negativa. Uma maneira de obter o valor dessa área consiste em particioná-la em pequenos intervalos e em seguida somar as áreas resultantes da partição. Em uma situação real, a partição de tal área é seguida pela estimativa de um valor para a curva dentro do intervalo de partição.

A integral de uma curva somente se torna exata quando o comprimento do intervalo de partição tende a zero. Entretanto, estimativas acuradas do valor de uma integral podem ser obtidas se um comprimento adequado de partição for utilizado. A Figura 2.20 ilustra o algoritmo.

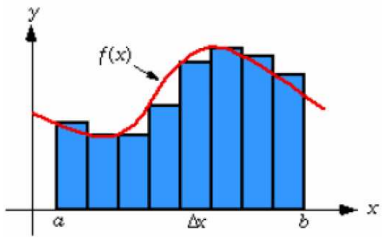


Figura 2.20: Integração de um sinal amostrado (*Ver equação 2.8*). Adaptado de [3]

A equação abaixo formula esse algortimo:

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x \tag{2.8}$$

em que:

$$\Delta x = \frac{b-a}{n} \quad \text{e} \quad x_i \text{ é a estimativa do valor da curva na partição i}$$

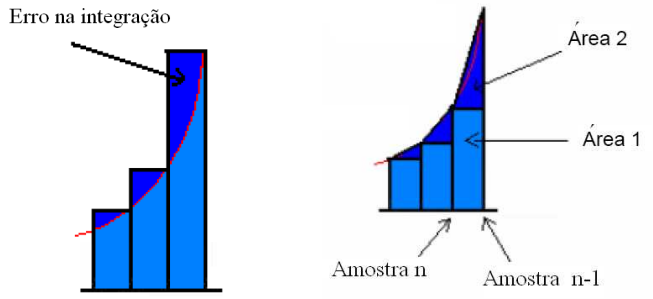
A equação 2.8 resume o princípio matemático da integração. A função a ser integrada é amostrada a intervalos constantes. As amostras assim obtidas são ponderadas pelo intervalo de amostragem e, em seguida, somadas.

Tal princípio é empregado também para a obtenção da velocidade e da posição do robô. Ou seja, a soma das amostras ponderadas de sua aceleração corresponde à sua velocidade. E a soma das amostras ponderadas de sua velocidade corresponde à sua posição. Tal ponderação é feita pelo tempo de amostragem. Por simplicidade, esse tempo será considerado igual 1. Desse modo, a consideração de seu real valor somente

acontecerá ao fim dos cálculos. Se assim não fosse, operações com aritmética de ponto flutuante seriam necessárias. A limitação da memória flash do ATMega8 em 8196 *bytes* justifica essa decisão.

Observa-se, pelo princípio da integração, que a soma das áreas entre o eixo de integração e a curva da grandeza a ser integrada. Desse modo, tendo 1 como período de amostragem, poder-se-ia afirmar que a integral de uma curva correponde à soma de suas amostras. Entretanto, comete-se um erro de aproximação ao assim se anunciar uma integração. Tal erro é observado na Figura 2.21(a).

Esses erros podem ser reduzidos se métodos de integração numérica mais eficientes forem utilizados. Um deles é o Método Trapezoidal. Nele, a área sob a curva é aproximada pela soma de duas áreas menores. Uma corresponde à área de um retângulo, cuja altura é a amostra n-1. A outra, à área de um triângulo, cuja altura é o módulo da diferença entre os valores amostra n e amostra n-1. A Figura 2.21(b) demonstra a integração por esse método.



(a) Integração por retângulos

(b) Método Trapezoidal

Figura 2.21: Erros gerados durante uma integração. Adaptado de [3]

O Método Trapezoidal é um método de aproximação linear de uma integral, portanto, seu modelo de primeira ordem. A equação (2.10) mostra como obter, em uma partição, a área (com o devido sinal) entre a curva a ser integrada e o eixo de integração, utilizando esse método.

$$Area_n = \left(Amostra_n + \frac{Amostra_n - Amostra_{n-1}}{2} \right) \times T \tag{2.10}$$

Tendo em vista a obtenção de valores coerentes, a calibração do acelerômetro fez-se necessária. A calibração de um acelerômetro consiste em obter a aceleração que esse apresenta estático sob a aceleração da gravidade. A aceleração deverá ser obtida para cada eixo do acelerômetro. Ela será a referência utilizada para se obter a aceleração real do robô quadrúpede.

Tanto a aceleração obtida pelo MMA7260Q quanto a obtida pelo LIS3LV02DQ devem passar pela etapa de calibração. A calibração tem a função de se fazer considerar a orientação normal do robô. No caso do MMA7260Q, a calibração também tem a função de considerar o *offset* a 0g que lhe é característico, cujo valor é $V_{dd}/2$, em que V_{dd} é sua tensão de suprimento. Tal consideração é feita ao se retirar da aceleração o valor de referência obtido na calibração A Figura 2.22 mostra o resultado de uma rotina de calibração aplicada ao MMA7260Q. Nele, o valor de referência corresponde ao resultado da calibração.

O valor da aceleração após uma calibração corresponde à subtração do sinal amostrado pelo valor de referência. A Figura 2.22 mostra as áreas obtidas pela integração de um sinal amostrado já calibrado. A

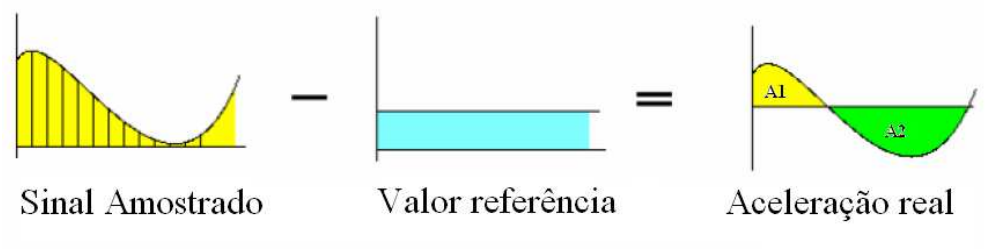
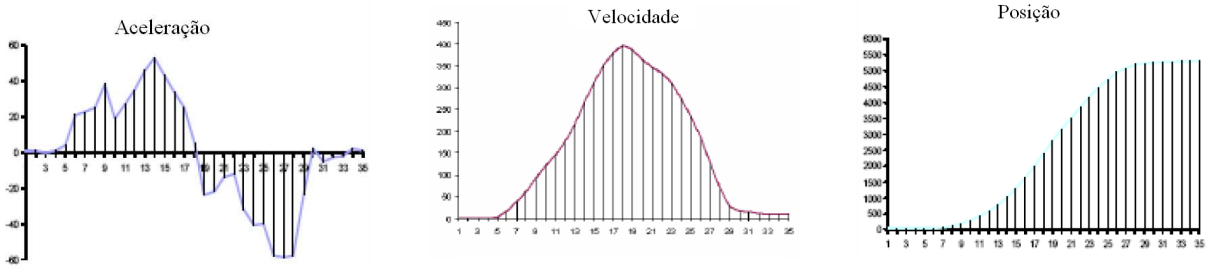


Figura 2.22: Aceleração após a calibração. Adaptado de [3]

área A1 corresponde a uma aceleração positiva. A área A2, a uma aceleração negativa.

Um último lembrete deve ser feito sobre os métodos de integração numérica. Sinais devem ser empregados ao se somar áreas que resultem no valor de uma integral. Somente assim o valor de uma integração numérica tem o significado físico esperado. Desse modo, um sinal positivo deve ser utilizado, se a área estiver acima do eixo de referência e um sinal negativo deve ser utilizado, caso contrário. A aplicação dessa análise à Figura 2.22 promove o seguinte resultado: a aceleração negativa manifesta em A1 diminui a velocidade do robô, enquanto a aceleração positiva manifesta em A2 lhe aumenta a velocidade.

O método trapeziodal será utilizado para o obtenção das integrais numéricas desse projeto. A Figura 2.23 mostra os valores de aceleração, de velocidade e de posição esperados ao se empregar esse método. Vale lembrar que, como o período de amostragem foi considerado 1, tais gráficos são apenas proporcionais aos esperados.



(a) Aceleração real (b) Velocidade real (c) Posição real

Figura 2.23: Aproximação proporcional da aceleração, da velocidade e da posição. Método Trapezoidal. Adaptado de [3]

As seções a seguir descrevem outros algoritmos utilizados na obtenção da posição do robô quadrúpede. Imperfeições características da amostragem serão melhor compreendidas nessas seções.

2.4.3 Calibração

Calibrar a aceleração consiste em compensar a influência que a orientação e o *offset* a 0g têm na medição realizada pelo acelerômetro. Tal compensação implica o reconhecimento e a retirada do valor de referência obtido. A acurácia desse valor somente é conseguida na ausência de movimento e com muitas amostras.

A calibração implementada nesse projeto consiste em obter a média de 4096 amostras de aceleração. A obtenção dessa média é feita para os eixos x e y do acelerômetro. Portanto, seu valor é armazenado em duas variáveis de referência, uma para cada eixo. Tais variáveis terão seu valor requerido toda vez que se for obter um novo valor real de aceleração. Esse novo valor será a diferença entre o valor obtido pelo acelerômetro e a variável de referência.

Nesse projeto, destina-se um período de $40ms$ à calibração do robô. Tal calibração é realizada automaticamente toda vez que se reseta o ATMega8 conectado ao acelerômetro.

2.4.4 Filtro digital

O filtro digital passa-baixas é um bom atenuador de ruído de alta freqüencia e média nula. Desse modo, ele pode ser utilizado para reduzir o ruído existente na aceleração fornecida pelo acelerômetro. Tal redução é importante para que se obtenha valores adequados de velocidade e de posição. Caso ela não acontecesse, a integração do ruído existente na aceleração produziria valores de espaço e de velocidade incoerentes.

Um bom modo de remover esse ruído é aplicar uma média ao sinal amostrado. Ou seja, obter a aceleração em um instante de tempo como a média de um conjunto de amostras recém-chegadas. Esse conjunto deve ter um número grande de amostras, de modo a obter uma boa estimativa para a aceleração do robô. Entretanto, tal número deve ser suficientemente pequeno, a fim de garantir que a perda de amostras, decorrente do número de amostras utilizado na média, não estime com má qualidade as variações da aceleração que não são provenientes do ruído.

O número escolhido para essas amostras foi de 64. Para esse número, um período de $10ms$ foi empregado, para cada média.

2.4.5 Filtro janela

O filtro janela consiste em eliminar pequenas variações da aceleração em relação ao valor de referência. Essas variações, normalmente provenientes de um ruído, geram erro na estimativa da velocidade. Tal velocidade, por ser diferente zero, indica um equivocado deslocamento do robô. A posição se torna uma medida incoerente.

A maneira encontrada para minimizar tais variações foi definir uma limiar para o módulo aceleração. Valores de aceleração que não ultrapassem em módulo esse limiar serão considerados nulos. Nesse projeto, utiliza-se o valor 10LSb como limiar para o LIS3LV02DQ. Como esse sensor é configurado para que 1024LSb corresponda à gravidade, todos os valores de aceleração, cujo módulo seja inferior à $4,8cm/s^2$, são desconsiderados.

A Figura 2.24(a) demonstra o mecanismo utilizado para discriminar esses valores de aceleração. Já a Figura 2.24(b) mostra a saída do filtro janela implementado com esse mecanismo de discriminação.

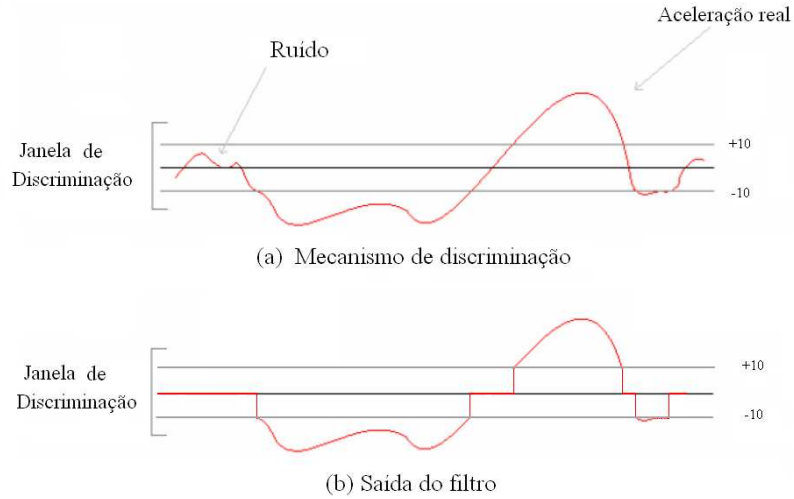


Figura 2.24: Filtro Janela. Adaptado de [3]

2.4.6 Verificação do fim do movimento

Em sistemas contínuos, a mudança de posição de um objeto promove uma aceleração cuja integral é nula ao fim do movimento. Entretanto, sistemas discretos de aceleração são apenas aproximações de sistemas contínuos. Tais aproximações conduzem a um erro na estimativa da aceleração, que ao ser integrado em um deslocamento, produz uma velocidade não-nula, embora constante.

Tal velocidade não corresponde à realidade do fim de um movimento de deslocamento, portanto deve ter seu valor zerado. Um bom modo de identificar o momento adequado para zerar essa velocidade é acompanhar os valores assumidos pela aceleração (real) do objeto. Caso a aceleração seja nula durante um certo intervalo de tempo, a velocidade também é considerada nula. Nesse projeto, utilizou-se 250 ms como o tempo de verificação dessa aceleração. Esse tempo corresponde a 25 estimativas do valor real da aceleração.

A Figura 2.25 mostra valores típicos assumidos pela aceleração durante um delocamento. O objeto, acelerado durante certo intervalo de tempo em uma direção, é desacelerado até parar. Nesse movimento, áreas são desenhadas indicando acréscimos e decréscimos de velocidade. Entretanto, as áreas correspondentes ao acréscimos não possuem o mesmo valor da dos decréscimos. Alguma velocidade, positiva ou negativa, ainda permanece após o movimento. Faz-se necessário o uso de um algoritmo de verificação.

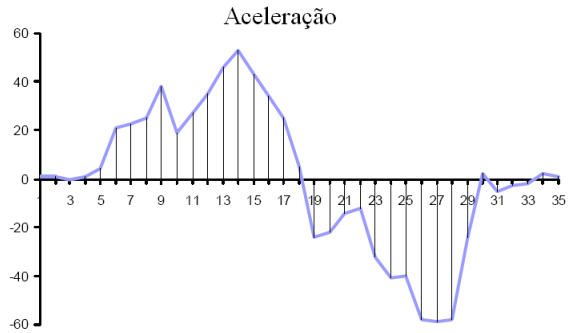


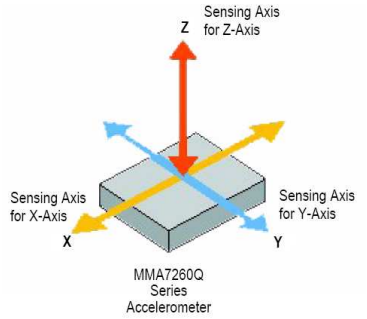
Figura 2.25: Aceleração típica de um movimento de deslocamento. Adaptado de [3]

2.4.7 Inclinômetro

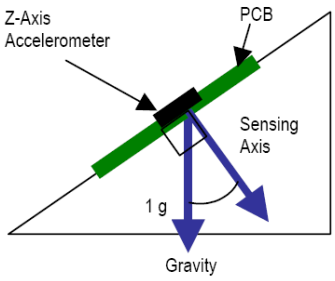
A inclinação do robô é uma medida estática, cujo valor é obtido pela comparação da aceleração presente em um ou mais eixos do acelerômetro com a aceleração da gravidade. Essa comparação freqüentemente se depara com as não-linearidades da sensibilidade, a qual é utilizada para obter a aceleração. Portanto, aconselha-se geralmente o emprego de tabelas para a obtenção de uma melhor estimativa para essa aceleração.

Como a obtenção de medidas de inclinação extremamente precisas não é necessária a esse projeto, uma única equação, que relaciona as acelerações supracitadas, foi utilizada. Tal equação tem como base a aplicação de relações trigonométricas em um triângulo retângulo. Para o entendimento de como obter tal relação, é necessário antes observar a disposição dos eixos de medição de um acelerômetro. A Figura 2.4.7 (a) mostra a disposição desses eixos para o MMA7260Q. A disposição dos eixos do MMA7260Q é semelhante à do LIS3LV02DQ, a única diferença reside no sentido desses eixos. O único efeito dessa diferença consiste na mudança do eixo de referência utilizado na medição da inclinação.

Nesse projeto, somente a inclinação do robô em relação ao eixo z (inc_z) fez-se necessária, tornando-o o melhor eixo de referência. A Figura 2.4.7 (b) relaciona a inclinação à aceleração da gravidade (g) e à aceleração medida no eixo z (a_z). Por essa figura, observa-se que o ângulo de inclinação do robô é congruente ao existente entre o eixo z e a aceleração da gravidade. A equação 2.11 relaciona essas grandezas.



(a) Eixos de medição da aceleração de um MMA7260Q. Fonte: [13]



(b) Componente gravidade no eixo z de um MMA7260Q. Fonte: [13] inclinado

$$\cos\left(inc_z\right) = \frac{a_z}{g} \quad \therefore \quad inc_z = \arccos\left(\frac{a_z}{g}\right) \tag{2.11}$$

Nessa equação, os valores g e a_z devem ser adequados ao *offset* e à sensibilidade do acelerômetro utilizado. Desse modo, se uma sensibilidade de $800mV/g$ for escolhida para o acelerômetro MMA7260Q, $a_z = a_{amostra} - \frac{V_{dd}}{2}$ e $g = 800$. Já para o acelerômetro LIS3LV02DQ , como a aceleração é armazenada em complemento de 2, se uma sensibilidade de $1024LSb/g$ for escolhida, $a_z = a_{amostra}$ e $g = 1024$. A única observação a ser feita é que a referência para a medição da inclinação no LIS3LV02DQ têm sentido de dentro para fora da superfície, sentido contrário ao do MMA7260Q.

2.4.8 Atualização das Variáveis de Integração

As estimativas para os valores de aceleração, de velocidade e de posicão em determinado instante de tempo devem ser armazenadas a fim de serem empregadas como variáveis de integração. Tal armazenamento consiste na atualização das variáveis que estimaram a aceleração, a velocidade e a posicão no instante de tempo imediatamente anterior, pelas variáveis do instante de tempo presente correspondentes.

2.4.9 Composição dos algoritmos

Os algoritmos descritos seguem uma seqüencia de aplicação bem definida. A apresentação dessa seqüencia é feita na Figura 2.26, através de um fluxograma.

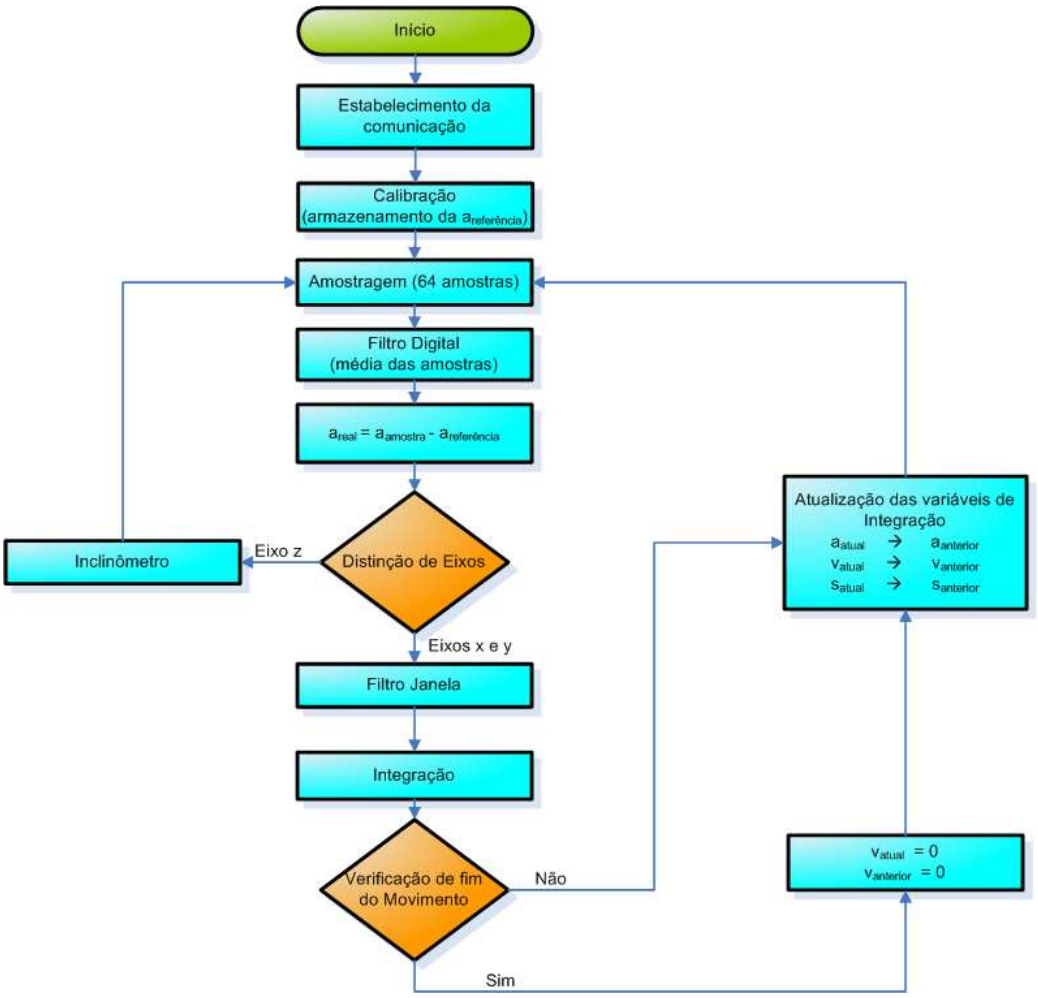


Figura 2.26: Fluxograma dos algoritmos de posicionamento

2.5 APRENDIZAGEM

O conceito de aprendizagem de máquina é algo muito estudado e ao mesmo tempo muito polêmico. Apenas recentemente, com o surgimento do computador moderno, é que a inteligência artificial gan-

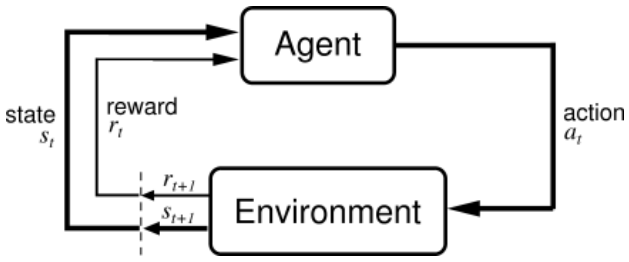


Figura 2.27: Interação entre o Agente e o Ambiente em um algoritmo de aprendizagem pore reforço. Fonte: [4]

hou meios e massa crítica para se estabelecer como ciência integral, com problemáticas e metodologias próprias. Desde então, seu desenvolvimento tem extrapolado os clássicos programas de xadrez ou de conversão e envolvido áreas como visão computacional, análise e síntese da voz, lógica difusa, redes neurais artificiais, aprendizagem por reforço e muitas outras[12].

O projeto tem como um dos objetivos a implementação de um algoritmo de aprendizagem para estimar os tempos de ativação de cada membro do robô quadrúpede. Dentre os algoritmos conhecidos, o de aprendizagem por reforço foi que proporcionou uma melhor adequação ao sistema [4]. Esse tipo de algoritmo utiliza o conceito de ação, estado e recompensa para o aprendizado. A Figura 2.27 mostra de forma resumida o conceito por trás desse algoritmo.

O Agente é quem toma as decisões (ou ações) destinadas à aprimorar o aprendizado. Tudo aquilo que não é o Agente, é o Ambiente. O Ambiente é quem "reage" às ações tomadas pelo Agente e retorna a ele o seu novo estado e uma recompensa pela ação tomada. Se o resultado não foi o esperado, a recompensa deve ser baixa, e caso contrário alta. A maior dificuldade na implementação desse tipo de algoritmo é na definição do que é ação, estado e quais as recompensas para as diversas possibilidades que se possa encontrar durante o aprendizado.

Porém, devido à falta de tempo hábil não foi possível a implementação do algoritmo de aprendizagem por reforço nesse projeto. Optou-se por uma abordagem mais simples e rápida, porém mais rudimentar, chamada de Passeio Aleatório Adaptativo (ou PAA). Deseja-se compensar essa rudimentalidade com os dados obtidos pelo acelerômetro, ou seja, utilizando o *feedback* do ambiente para se determinar a qualidade do processo.

2.5.1 Passeio Aleatório Adaptativo

Um passeio aleatório ou caminhada aleatória, algumas vezes chamado de "caminhada do bêbado", é uma formalização intuitiva da tomada de vários passos consecutivos, cada qual em um direção aleatória. Por exemplo, o caminho percorrido por uma molécula ou por um líquido ou gás é um passeio aleatório. Num passeio aleatório a direção de um ponto no caminho para o próximo é escolhido aleatoriamente.[13].

Porém o que se deseja fazer nesse projeto é algo mais refinado do que o passeio aleatório puro e simples. A idéia é utilizar como base uma ordem de acionamento no qual o quadrúpede consiga andar para frente, mesmo que de maneira rudimentar, e a partir dela utilizar o método de passeio aleatório para

[12] http://pt.wikipedia.org/wiki/Intelig%C3%AAncia_artificial
[13] http://pt.wikipedia.org/wiki/Passeio_aleat%C3%B3rio

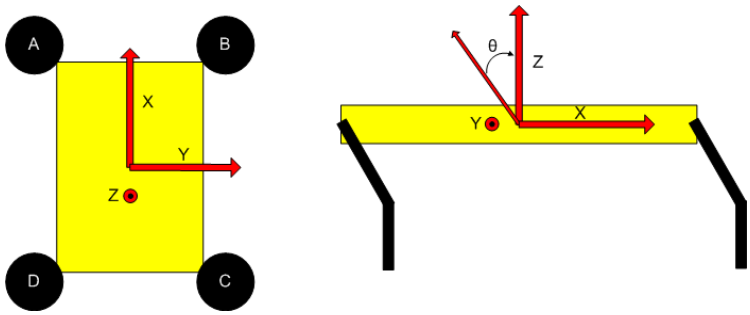


Figura 2.28: Representação do quadrúpede visto de cima (imagem à esquerda) e em perfil (imagem à direita) com os respectivos eixos de referência e inclinação usados para o acelerômetro.

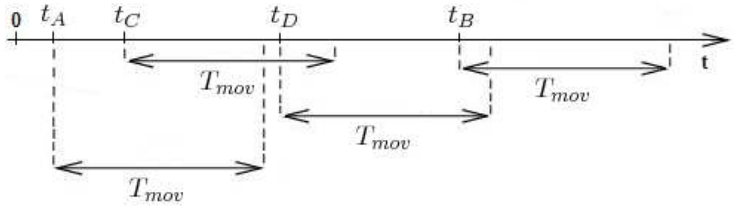


Figura 2.29: Exemplo de tempos de ativação representados na linha do tempo.

gerar uma nova ordem de acionamento. A nova ordem é testada e aprovada ou reprovada de acordo com os dados coletados pelo acelerômetro durante o caminhar.

Como mencionado na Seção 2.2.2, não adianta uma sequência de movimento para cada servomotor se os membros não estão em harmonia. É necessário definir como cada membro será chamado e como cada eixo direcional foi adotado antes de discutir o algoritmo. A Figura 2.28 ilustra uma representação do quadrúpede visto de cima e em perfil. Para melhor distinção durante o projeto, foi designado a cada membro uma letra, sendo A o membro frontal esquerdo, B o frontal direito, C o dianteiro direito e D o dianteiro esquerdo. Na Figura também é ilustrado os eixos de referência utilizados pelo acelerômetro, incluindo o eixo Z e o ângulo θ de inclinação.

Como todas as patas possuem a mesma sequência de posições dos servomotores, isso faz com que elas executem toda a sequência de movimento em um mesmo período de tempo T_{mov}. Desta forma, para se gerar uma ordem de acionamento diferente, deve-se diferenciar os seus tempos de início de movimento. O algoritmo desenvolvido tem como objetivo de determinar esse conjunto de tempo, dado por $U = \{t_A, t_B, t_C, t_D\}$, que representam o início das patas A, B, C e D respectivamente. A Figura 2.29 ilustra um exemplo de tempos de acionamento de cada pata representados na linha do tempo. Nessa figura, pode-se observar que a sequência de acionamento é dada por A, C, D e B. Sendo que C inicia seu movimento após A ter iniciado, mas antes de finalizado. E D inicia o movimento momentos antes de C finalizar. O mesmo ocorre com B, iniciando momentos antes de D finalizar. Para essa configuração, A só irá iniciar seu movimento novamente após B ter finalizado, ou seja, após $t = t_B + T_{mov}$.

A sequência reflete diretamente no caminhar do quadrúpede, principalmente quando duas ou mais patas

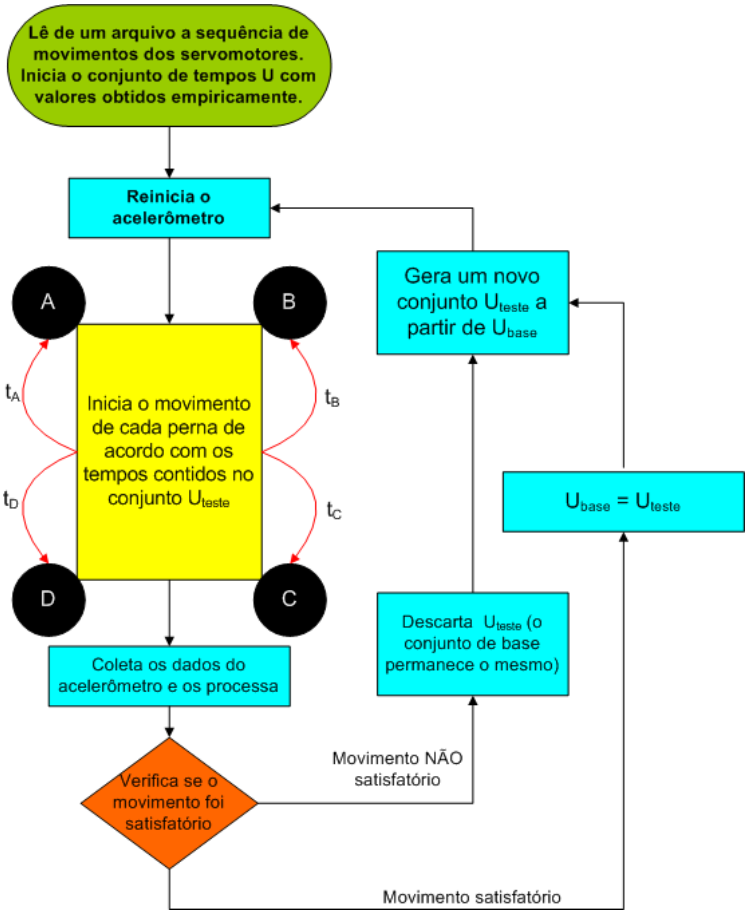


Figura 2.30: Fluxograma do algoritmo de treinamento baseado em Passeio Aleatório Adaptativo.

estão em movimento ao mesmo tempo, pois o problema se torna em obter um equilíbrio dinâmico[14]. Se o algoritmo proposto for capaz de determinar qual o melhor conjunto U para que o quadrúpede desempenhe um movimento satisfatório, o objetivo principal do trabalho será alcançado. Esse algoritmo é descrito no fluxograma da Figura 2.30.

O critério de um movimento satisfatório adotado para esse projeto é o seguinte: o robô deverá se movimentar para frente o mais rápido possível e obter ao longo de seu caminhar a menor inclinação máxima(θ_{max}) com relação ao eizo Z possível. Para quantizar esses critérios, utiliza-se dos dados obtidos pelo acelerômetro durante o caminhar. Ele é capaz de medir o espaço percorrido no eixo X e Y, e a inclinação com relação ao eixo Z durante o seu movimento. Desta forma, para verificar se o robô movimenta para frente, verifica-se a razão $R = \frac{s_x}{s_y}$ (epaço percorrido em X com relação ao espaço percorrido em Y). Quanto maior e positiva (pois não se deseja que o robô movimente-se para trás) a razão R, mais o robô terá movimentado para frente do que para o lado. Idealmente essa razão tenderá ao infinito, indicando um movimento exclusivo na direção do eixo X.

O valor obtido pelo inclinômetro é utilizado para determinar o equilíbrio do robô durante o movimento. Se ao longo do caminhar o robô se desequilibrar de forma excessiva, a inclinação máxima obtida será

[14]O equilíbrio dinâmico é obtido utilizando-se do momento de inércia do corpo em movimento. Para um quadrúpede parado, retirar duas patas do solo ao mesmo tempo certamente irá fazê-lo desequilibrar e tombar. Já em movimento, esse desequilíbrio pode ser compensado pela inércia do corpo.

maior do que se o mesmo não desequilibrasse. Esse método foi proposto para tentar quantizar esse tipo de ocorrência, porém essa informação pode levar a erros de julgamento. Propõe-se que futuramente sejam implementados outros métodos para aprimorar a obtenção de tal informação.

O programa tem seu início com a leitura de um arquivo texto *(.txt)* com a sequência de movimento dos servomotores obtidas empiricamente. Como mensionado, essa sequência é idêntica para cada pata. Após o carregamento dessa sequência na memória RAM do IBM-PC, carrega-se um conjunto inicial de tempos U_{base}, obtidos empiricamente, para a base dos cálculos do algoritmo aleatório. O algoritmo entra então em uma rotina de temporização dos disparos de sequência. Na primeira vez, $U_{base} = U_{teste}$, ou seja, a primeira sequência de testes é a própria sequência inicial adotada. Toda vez que o temporizador atinge um dos tempos contidos no conjunto U_{teste}, a respectiva pata tem sua sequência de movimentos disparada. Uma vez iniciado o movimento de uma pata, o algotimo retorna novamente para a rotina de temporização para aguardar o próximo disparo. Os dados da pata que já iniciou o movimento são enviados pelo IBM-PC em segundo plano, utilizando-se dos recursos fornecidos pelas *POSIX Threads* (ou **PThreads**). Desta forma é possível obter o movimento de todos os membros ao mesmo tempo, não sendo necessário o aguardo da finalização de uma sequência para iniciar a próxima.

Para obter um valor mais acurado pelo acelerômetro, decidiu-se disparar pelo menos dez vezes cada pata. Os dados coletados durante essa fase são então processados e julgados de acordo com a razão R e o θ_{max}, descritos anteriormente. Se o conjunto U_{teste} foi aprovado, então $U_{base} = U_{teste}$, ou seja, o conjunto utilizado anteriormente será substituido pelo conjunto aprovado. Caso seja reprovado, o conjunto de base mantém-se o mesmo. A sequência de base é aquela cujo valor será usado para os cálculos da nova sequência de testes. É nesse cálculo que o algoritmo de passeio aleatório irá atuar.

Seja um intervalo de tempo entre $-t_{max}$ e t_{max}, o algoritmo aleatório gera um valor r nesse intervalo para cada pata, formando portanto um conjunto de valores dado por $L = \{r_A, r_B, r_C, r_D\}$. O novo conjunto U_{teste} é formado somando-se os termos de cada pata do conjunto U_{base} com o conjunto L. Representando essa soma matricialmente, temos:

$$\mathbf{U_{teste}} = \mathbf{U_{base}} + \mathbf{L}, \tag{2.12}$$

sendo,

$$\mathbf{U_{teste}} = \begin{bmatrix} t_{A_{teste}} \\ t_{B_{teste}} \\ t_{C_{teste}} \\ t_{D_{teste}} \end{bmatrix}, \mathbf{U_{base}} = \begin{bmatrix} t_{A_{base}} \\ t_{B_{base}} \\ t_{C_{base}} \\ t_{D_{base}} \end{bmatrix}, \mathbf{L} = \begin{bmatrix} r_A \\ r_B \\ r_C \\ r_D \end{bmatrix} \tag{2.13}$$

Deve-se levar em consideração que os tempos são sempre positivos, portanto se alguma dessas somas resultarem em valores negativos, o tempo é considerado como zero. Em nosso projeto o valor de t_{max} foi especificado sem nenhum critério em específico. Foi levado em consideração apenas o tempo total (T_{mov}) que a pata leva para executar todo o movimento. Portanto os valores numéricos para essas variáveis depende do tipo de sequência utilizada, e serão especificados no capítulo de resultados experimentais (Capítulo 3).

Inicialmente o critério de aprovação ou reprovação deve ser bastante flexível, aprovando quase todos

os movimentos executados. Conforme o número de aprovações naquele critério aumenta, o critério deve se tornar mais rígido. Desta forma a tendência é de que ao final de vários treinamentos o critério esteja bastante seletivo e o conjunto de tempo U_{base} resulte em um caminhar suave e satisfatório.

3 RESULTADOS EXPERIMENTAIS

Neste capítulo está presente o resultado dos experimentos feitos com os algoritmos descritos no capítulo anterior.

3.1 INTRODUÇÃO

O desenvolvimento desse projeto envolveu várias etapas. A maior parte delas foi para a montagem da plataforma quadrúpede e padronização do sistema de comunicação para que o mesmo pudesse ser usado em trabalhos futuros. A padronização do sistema favorece o trabalho em equipe, já que os trabalhos podem ser modularizados e trabalhados independentemente sem se preocupar com adaptabilidade das informações ou protocolos.

Os resultados experimentais foram obtidos apenas em seu último mês de desenvolvimento e estão em sua maioria em forma de vídeos, pois o caminhar de um robô é algo difícil de se representar em figuras, mesmo quando apresentadas em formas de slides. Aqui tentaremos mostrar parte destes resultados em slides, porém tentando ao máximo descrever o ocorrido. Os resultados referêntes ao acelerômetro serão mostrados em forma de gráficos obtidos através de resultados coletados com o robô quadrúpede parado e andando.

3.2 ALGORITMOS DE MOVIMENTO

Em conjunto com o projeto desenvolvido por Souto[6], foram desenvolvidas diversas sequências de movimento para os servomotores. Souto criou uma função em seu programa para que as posições dos servomotores fossem geradas no intervalo de 0 a 255, assim como utilizado nesse projeto. Inicialmente foram geradas sequências para apenas um membro, e a partir dela foram feitas análises do movimento balístico. As seções seguintes referem-se aos experimentos na ordem em que foram executados. Após cada um, uma breve análise dos resultados é feita.

3.2.1 Experimento 1

Primeiramente foi testado o movimento de uma perna de cada vez, sendo que enquanto uma estava em movimento, as outras três permaneciam paradas. Porém, ao se retirar um dos membros inferiores do solo, a plataforma se desequilibrava e não conseguia voltar à sua posição de origem. Isso acontece porque os servomotores traseiros são mais exigidos com relação aos dianteiros, pois a projeção do centro de massa não é coincidente com o centro geométrico dos pontos de apoio das patas, fazendo surgir um torque gerado pela gravidade no sentido anti-horário (Figura 3.1).

Qualquer movimento que envolvia a retirada de uma das patas traseiras, enquanto as demais permaneciam no solo, faziam com que a plataforma tombasse, impossibilitando o seu movimento. Para tentar contornar esse problema decidiu-se mover as três patas que estavam no solo para trás, como se estivessem

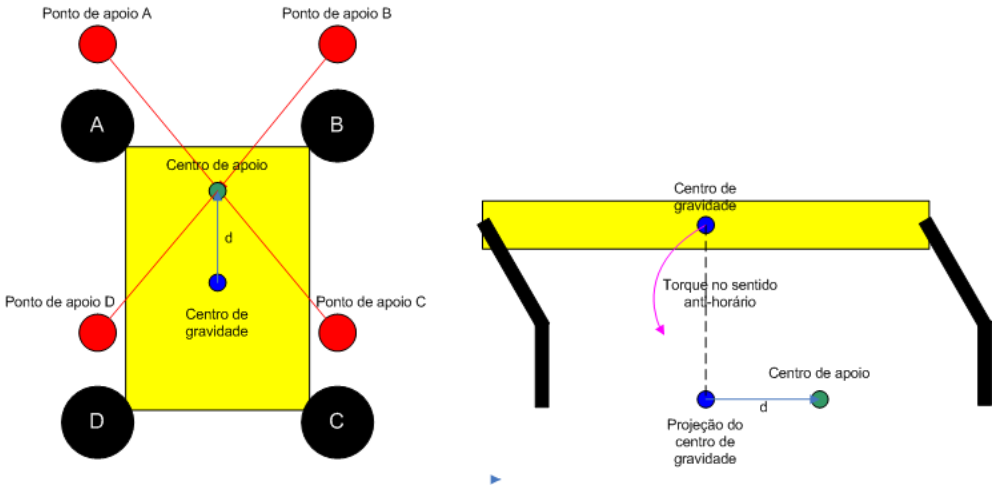


Figura 3.1: Ilustração dos pontos de apoio e do torque gerado pelo centro de massa.

impulsionando o robô para frente. Infelizmente isso também não adiantou. Essa fase experimental tentou obter o movimento do robô através do equilíbrio estático, pois os movimentos balísticos foram realizados em apenas uma pata por vez e enquanto a plataforma permanecia parada.

3.2.2 Experimento 2

Como o equilíbrio não pôde ser obtido utilizando-se técnicas estáticas, decidiu-se partir para o equilíbrio dinâmico, ou seja, utilizando da própria inércia para manter o equilíbrio da plataforma. O equilíbrio dinâmico não é uma tarefa simples de se obter, e foge do escobo desse trabalho, porém poder-se-ia tentar obtê-lo de maneira empírica. Através de diversas tentativas feitas com um membro em movimento, concluiu-se que esse tipo de movimento não proporcionava um equilíbrio dinâmico. Então partiu-se para tentativas com duas ou mais patas em movimento simultâneo.

Observando o movimento dos animais, reparamos que os mesmos movimentavam todos os membros ao mesmo tempo, porém em uma determinada sincronia específica. Essa sincronia muda conforme o tipo de passada, ou seja, para passos lentos, para trotes, para galopes e etc. Adotamos inicialmente para o robô quadrúpede os passos semelhantes ao trote de cavalos. A Figura 3.2 obtida no site[1] da Wikipedia ilustra como esse caminhar se caracteriza.

A Figura 3.3 mostra como o trote é implementado no caso do robô quadrúpede: enquanto as patas A e C se movimentam simultaneamente no solo, no sentido de norte-sul, impulsionando o quadrúpede para frente, as patas B e D se movimentam balísticamente(portanto sem contato com o solo) no sentido sul-norte. Após o término dessa sequência, as patas A e C executam o mesmo movimento feito pelas patas B e D, e vice-versa. Apesar de a princípio parecer um movimento caótico, essa sequência foi gerada pelos algoritmos de [6] e testada. A Figura 3.4 apresenta slides do movimento obtido com a implementação dessa caminhada, porém com a plataforma quadrúpede suspensa.

Ao se testar essa sequência com o robô no solo, obteve-se um resultado surpreendente: ele movimentou-se para frente a uma velocidade satisfatória e só apresentou apenas pequenos erros de passada. Os slides

[1] http://pt.wikipedia.org/wiki/Trote_(equita%C3%A7%C3%A3o)

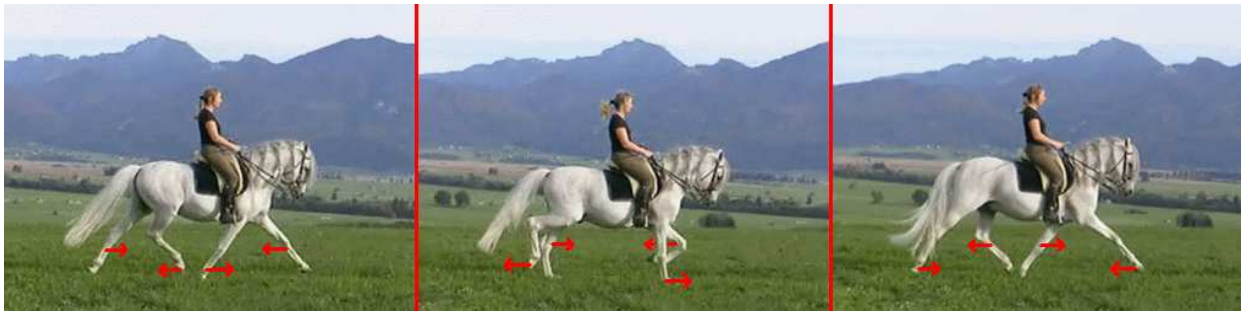


Figura 3.2: Sequência de movimento baseada em animais e utilizada para o robô quadrúpede (trote de cavalo). Fonte: $Wikipedia$[1]

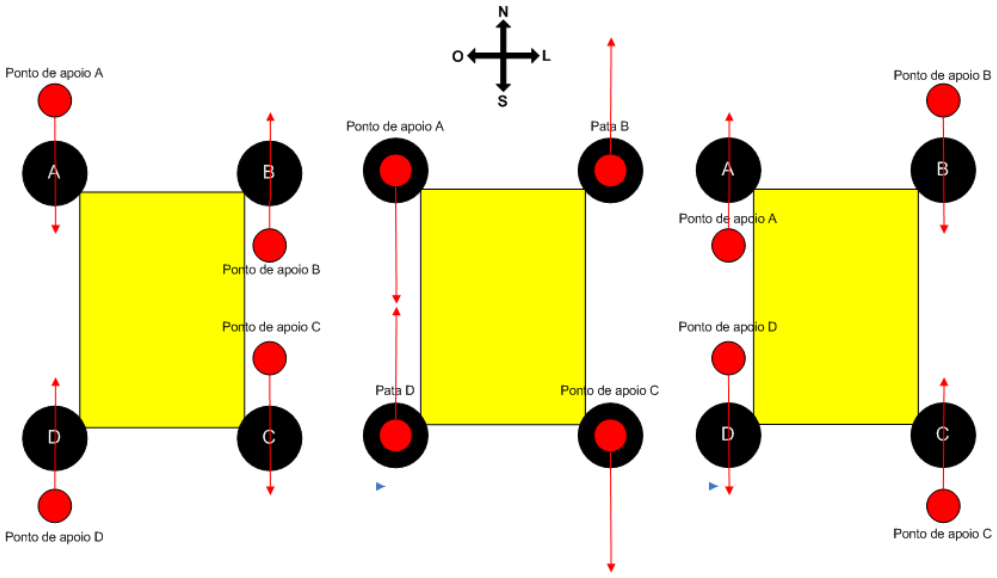


Figura 3.3: Representação da sequência de movimentos de trote na plataforma quadrúpede.

apresentados na Figura 3.5 mostram como o robô quadrúpede se comportou durante esse movimento.

Foi adotado assim esse tipo de caminhar como base do desenvolvimento para o treinamento de aprendizagem. Apesar de não ser um caminhar perfeito, ele apresenta características boas para um início de treinamento de aprendizagem, evitando tentativas desnecessárias que podem danificar o robô.

A sequência de movimentos dos servomotores foi idêntica para cada pata, sendo os seus respectivos valores mostrados na tabela 3.1. Nessa tabela o “Servo 1” representa o servomotor responsável pela junta entre a perna e a plataforma, o “Servo 2” o central e o “Servo 3” o servo mais próximo do solo.

3.2.3 Experimento 3

Independente do treinamento, foi desenvolvido em paralelo, com base na sequência utilizada no experimento 2, uma nova sequência de trote ajustada empiricamente. O movimento para frente do robô foi mais lento, porém não apresentou erros de desequilíbrio nem de passadas como no experimento passado. Foi constatado que o trote é bastante satisfatório e só apresentou problemas nos contatos das patas com o solo. A borracha utilizada nas patas deslisou, dificultando o impulsionamento do robô. Como o contato foi

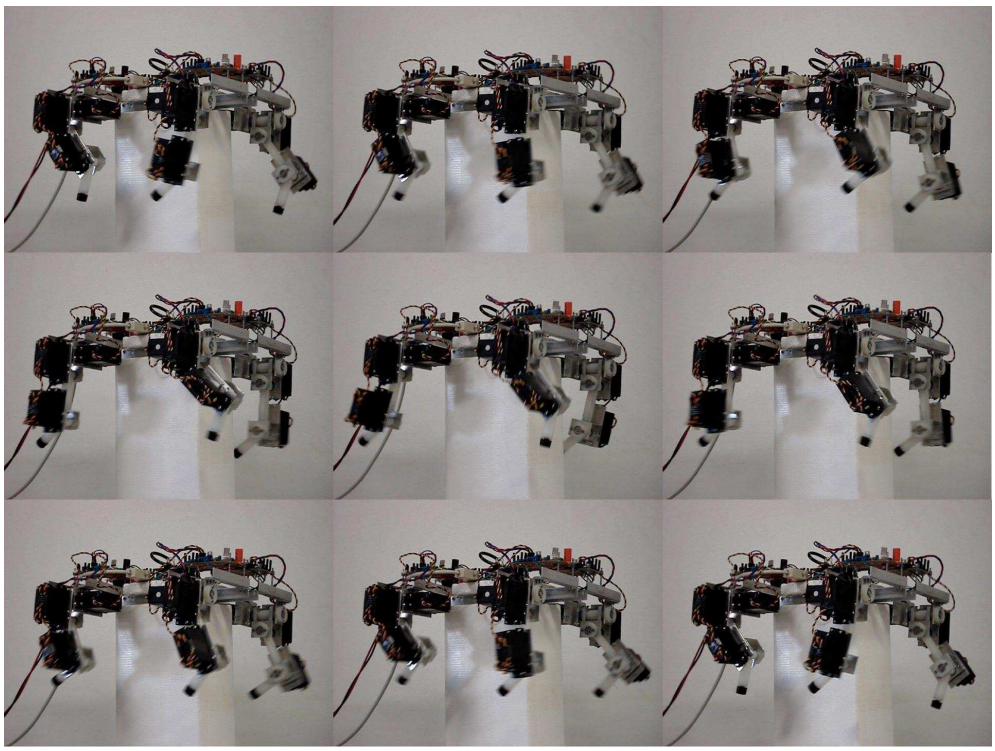


Figura 3.4: Sequência de slides do trote implementado ao robô quadrúpede com ele suspenso.

prejudicado, não foi possível uma análise mais conclusiva da sequência. Esse trote é mostrado no vídeo "Trote Final.mpg" presente no CD em anexo.

3.3 SISTEMA DE SENSORIAMENTO

3.3.1 Avaliação dos algoritmos de posicionamento

Os algoritmos de posicionamento foram testados em duas situações. Primeiramente, manteve-se o robô em repouso durante um período de seis segundos. Nesse período, mediu-se a sua aceleração, estimando-se não só o seu real valor, como também a velocidade alcançada pelo robô.

Em seguida, movimentou-se o robô segundo o algoritmo de locomoção implementado por Souto [6]. Durante seis segundos de locomoção, estimaram-se novamente a aceleração e a velocidade do robô. Empregou-se para tal a seqüência de algoritmos de posicionamento descrita na seção 2.4.9.

O IBM-PC solicitou 500 estimativas de velocidade e 500 estimativas de aceleração durante o repouso do robô. Um número idêntico de solicitações aconteceu durante a sua locomoção. Os dados obtidos nessas solicitações apresentaram bastante ruído, sendo necessário passá-los por um filtro passa-baixas para melhor visualização. Com essa finalidade, um filtro Butterworth de segunda ordem com freqüência de corte igual a 10 Hz foi utilizado. A escolha desse filtro objetivou a suavização das curvas de aceleração e de velocidade, possibilitando uma melhor análise dos dados obtidos. As Figuras 3.6 e 3.7 mostram a aceleração real(calibrada) antes e depois do filtro Butterworth. A Figura 3.6 apresenta o valor da aceleração em x (a_x); a Figura 3.7 , o valor da aceleração em y (a_y).

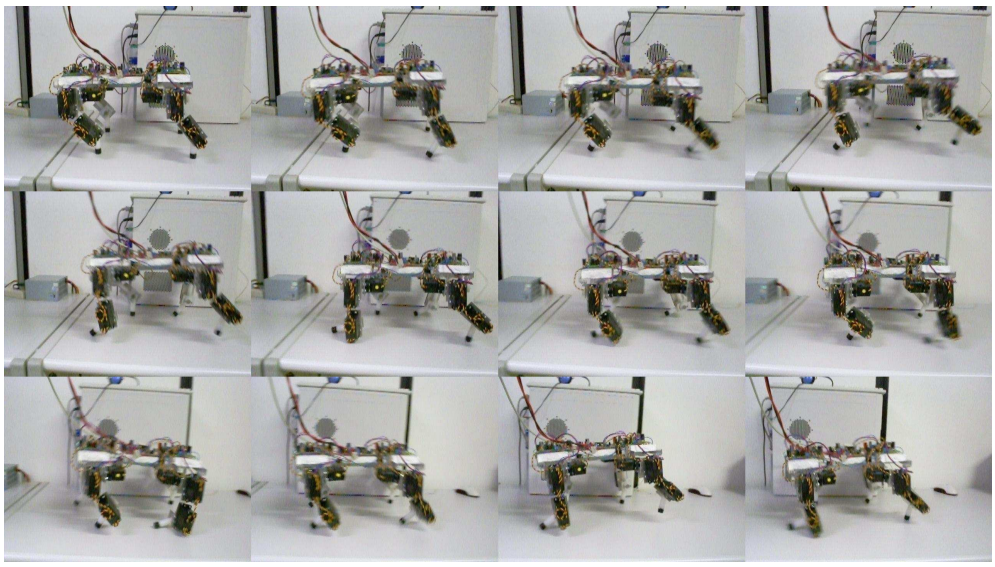


Figura 3.5: Sequência de slides do robô quadrúpede executando o movimento de trote no solo.

Nota-se pelas figuras que a aceleração manteve valores pequenos (menores que 10LSB, que são aproximadamente 24cm/s^2) durante quase todo o repouso do quadrúpede. Portanto, os algoritmos de calibração e de filtro janela são eficientes. Ou seja, não só as variações na aceleração do robô ocorrem em torno do valor zero como também são em módulo superiores ao limiar de 10LSB. Logo, as estimativas de aceleração obtidas durante o repouso fornecem resultados coerentes.

Durante o movimento foram obtidos valores inadequados de aceleração. Em ambos os eixos de medição, a aceleração estabilizou-se em valores não-nulos. Em x, a aceleração tendeu a assumir valores negativos; em y, valores positivos. Portanto, a ausência de algoritmos de compensação do efeito da gravidade proporcionou estimativas de aceleração que não são compatíveis com o movimento do robô.

A integração da aceleração é uma estimativa da velocidade do robô. Durante uma situação de repouso, a velocidade do robô é idealmente zero. Desse modo, uma boa estimativa de velocidade é aquela que, caso oscile, o faça em torno de zero. Já, durante o movimento do robô, considera-se uma boa estimativa da velocidade aquela cujo valor cresça num primeiro momento, se mantenha constante num segundo e zere ao final.

Apesar do resultado satisfatório obtido durante o repouso, o mesmo não aconteceu durante o movimento. Como mencionado anteriormente, a aceleração em movimento apresenta um valor médio não nulo. Desta forma, a velocidade correspondente a essa aceleração irá crescer/decrescer indefinidamente. O resultado obtido experimentalmente é apresentado nas Figuras 3.8 e 3.9. Como pode-se observar, a velocidade no eixo x (v_x) decresce e a velocidade no eixo y (v_y) cresce indefinidamente, conforme o esperado.

Demonstra-se assim que o algoritmo de posicionamento utilizado nesse projeto ainda não está preparado para aplicações 3D. Tal algoritmo desconsidera as mudanças de orientação que ocorrem enquanto o robô se movimenta. Como essa orientação não pode ser relevada, os algoritmos citados nesse projeto devem ser alterados a fim de se estimar corretamente a posição do robô. A principal alteração consiste em considerar a presença dinâmica do efeito da gravidade na locomoção do robô.

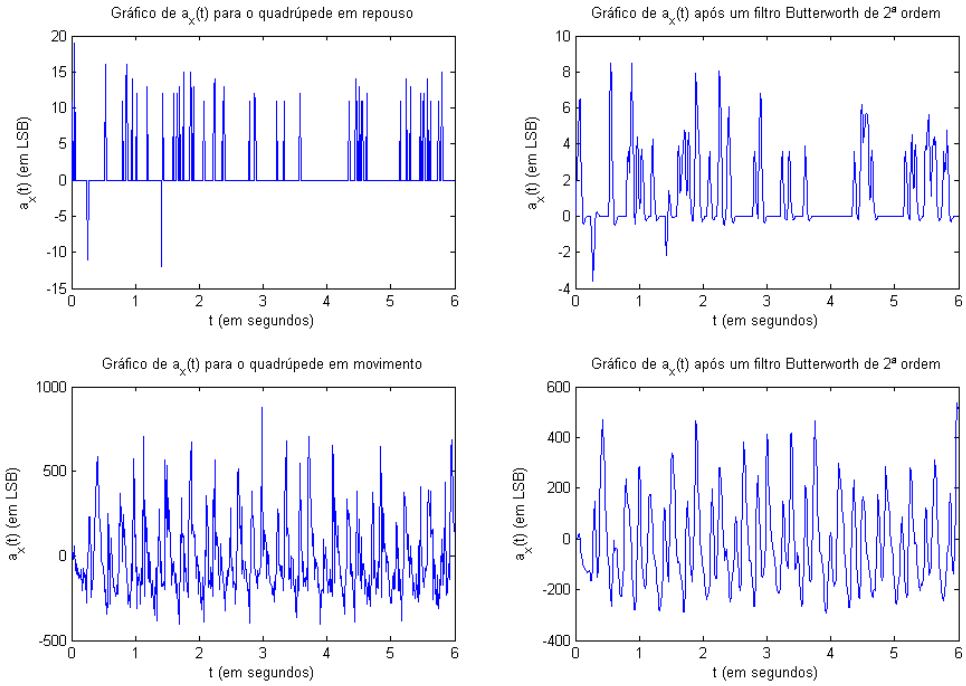


Figura 3.6: Gráfico da aceleração em x durante o repouso e durante a locomoção do robô quadrúpede

3.3.2 Avaliação do inclinômetro

A avaliação do inclinômetro foi feita durante a aplicação de uma seqüência de movimentos de rotação. Ou seja, à medida que se girava o robô, um programa executado pelo IBM-PC solicitava do ATMega8 os valores de aceleração e de inclinação referentes ao eixo z . A seqüencia de movimentos executada consiste em 5 giros. Algumas etapas desses giros são mostradas na Figura 3.10 .

A descrição do movimento do robô pode ser feita da seguinte maneira. Parte-se do eixo z (eixo de referência) em sentido anti-horário até se rotacionar o robô em $90°$. Em seguida, gira-se o robô em sentido horário em $180°$. O robô fica em uma posição diametralmente oposta à posição anterior. Dessa posição, retorna-se o robô ao eixo de referência. Aplica-se novamente um giro de $90°$ ao robô, desta vez em sentido anti-horário. Com tal giro, o robô fica de “cabeça para baixo”. Para terminar, o robô é rotacionado novamente à sua posição inicial, permanecendo aí até o término da execução do programa.

Durante a execução do movimento, foram obtidos os gráficos da inclinação e da aceleração no eixo z. Tais gráficos são apresentados em sua forma original e em sua forma modificada na Figura 3.11 . Nessa figura, as curvas originariamente obtidas pelo IBM-PC são mostradas à esquerda, enquanto o resultado da passagem de tais curvas por um filtro passa-baixas é mostrada à direita. Um filtro Butterworth idêntico ao utilizado na seção 3.3.1 é empregado com essa função. Os instantes em que o robô apresenta as posições mostradas na figura são destacados no gráfico 3.11 que apresenta a inclinação já filtrada. Marcadores em forma de círculo são utilizados com esse intuito.

A inclinação esboçada na Figura 3.11 corresponde fielmente ao menor ângulo existente entre o vetor aceleração da gravidade e o plano que contém o corpo do robô. A coerência desse resultado valida o algoritmo utilizado nessa seção. Tal algoritmo consiste na equação (2.11). Em tal equação, a inclinação é

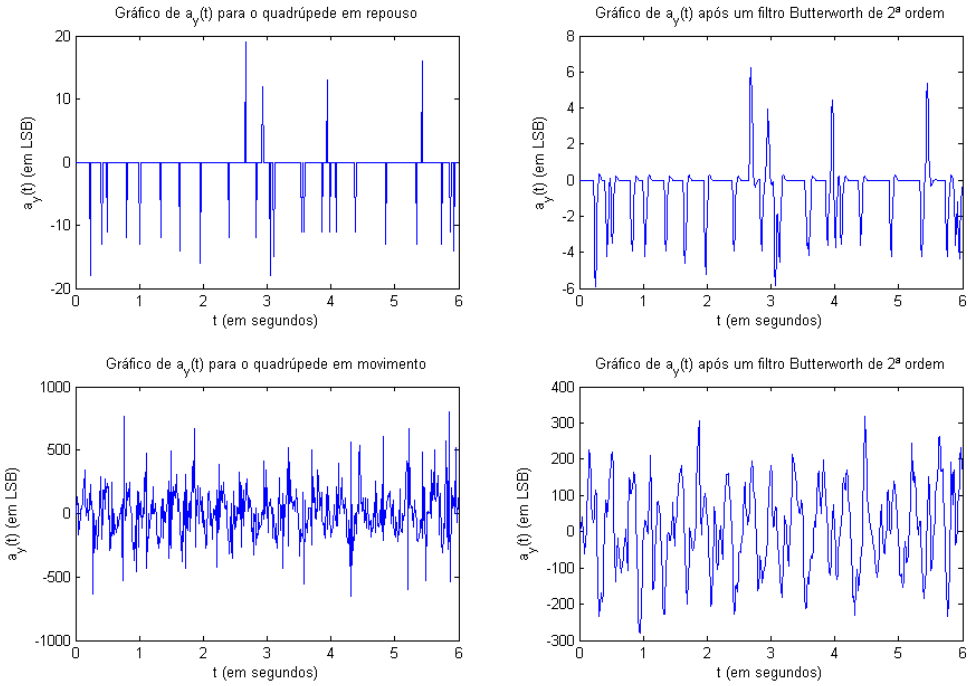


Figura 3.7: Gráfico da aceleração em y durante o repouso e durante a locomoção do robô quadrúpede

obtida através da aplicação de relações trigonométricas. Tais relações são aplicadas na aceleração presente no eixo z, única variável desta equação.

3.4 ALGORITMOS DE APRENDIZAGEM

Para a segurança da plataforma, os primeiros testes foram feitos com a plataforma suspensa. Durante esses testes, foi constatado que o barramento de comunicação ficou sobrecarregado ao utilizá-lo ao mesmo tempo para enviar os dados necessários para os cálculos do acelerômetro e os comandos de posições dos servomotores. Isso se tornou necessário porque os cálculos do acelerômetro tinham que ser feitos no IBM-PC, pois o ATMega8 não tinha espaço suficiente em sua memória **Flash** para o algoritmo. Essa limitação interferiu gravemente na temporização dos dados enviados aos servomotores, portanto comprometendo o funcionamento de ambos algoritmos.

Foi decidido então que, apesar de implementado, o algoritmo de aprendizagem não iria ser testado com a plataforma no chão, pois além do problema do barramento de comunicação, os dados coletados pelo acelerômetro ainda precisam ser tratados com uma matemática mais aprofundada, principalmente quando tratando da influência da gravidade na aceleração final. Desta forma, os dados obtidos pelo acelerômetro não representaram ser uma fonte fiél de informação. Os testes feitos com tais informações poderiam fazer a plataforma se quebrar com tentativas absurdas e impróprias para sua arquitetura.

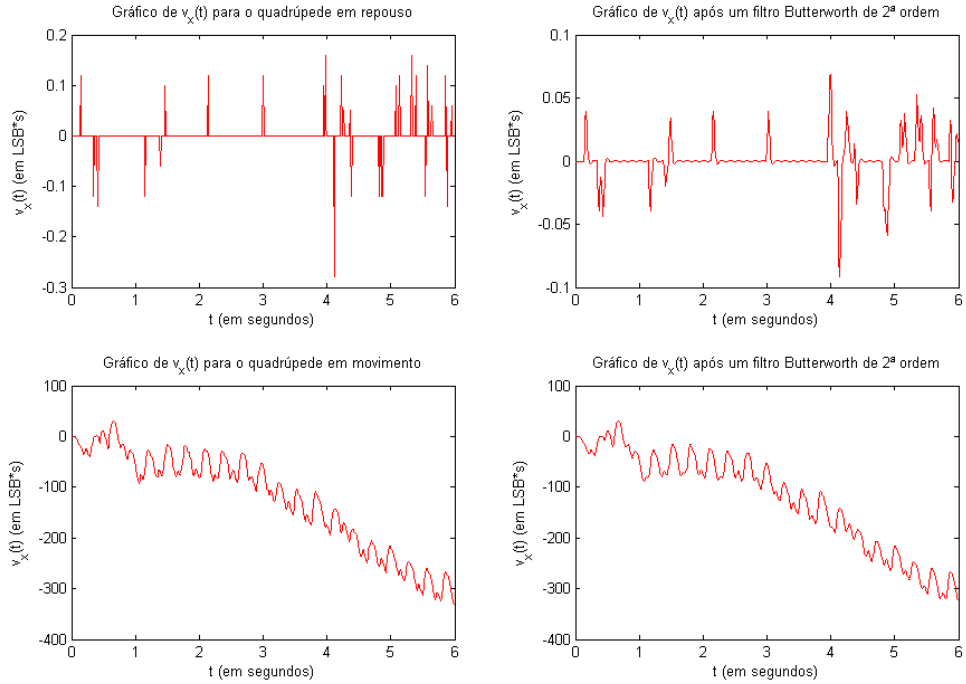


Figura 3.8: Gráfico da velocidade em x durante o repouso e durante a locomoção do robô quadrúpede

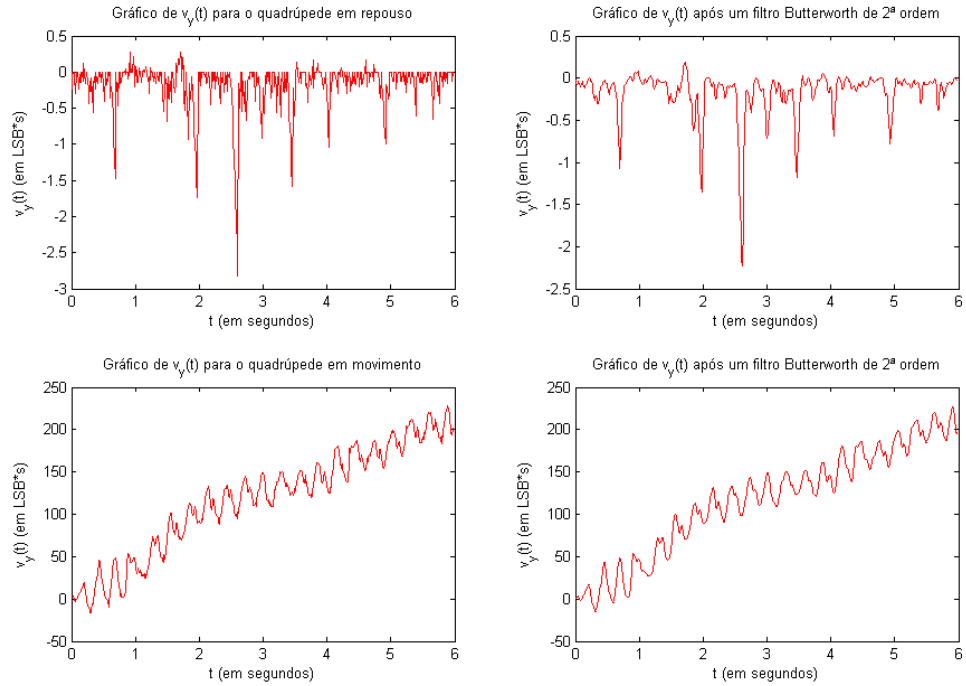


Figura 3.9: Gráfico da velocidade em y durante o repouso e durante a locomoção do robô quadrúpede

Tabela 3.1: Sequência de movimento dos servomotores para o trote

Sequência	Servo 1	Servo 2	Servo 3
0	128	191	64
1	128	190	54
2	128	188	47
3	128	185	41
4	128	181	37
5	128	177	35
6	128	171	34
7	128	165	34
8	128	159	36
9	128	152	38
10	128	144	43
11	128	136	49
12	128	127	57
13	128	118	67
14	128	107	82
15	128	107	82
16	128	124	51
17	128	137	31
18	128	149	15
19	128	160	3
20	128	170	0
21	128	179	0
22	128	188	0
23	128	196	0
24	128	202	0
25	128	205	3
26	128	206	15
27	128	204	31
28	128	199	51
29	128	189	82

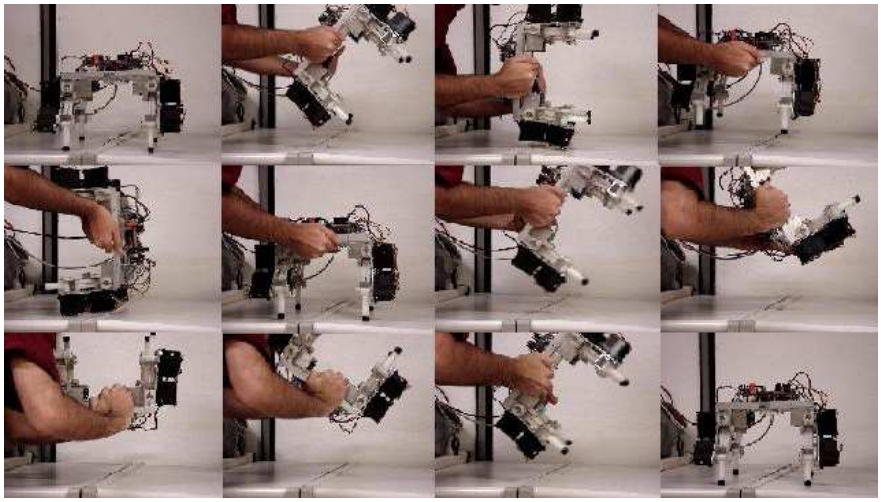


Figura 3.10: Seqüencia de movimentos realizada pelo robô durante a rotação.

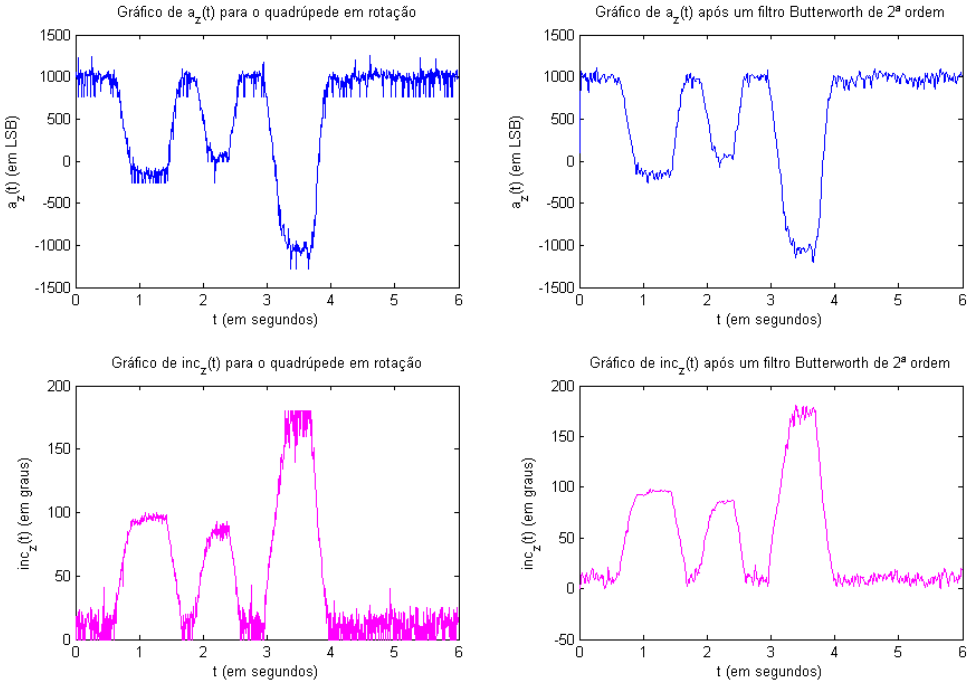


Figura 3.11: Gráfico da inclinação e da aceleração no eixo z para um movimento de rotação

4 CONCLUSÕES

Esse trabalho é a continuidade do desenvolvimento de uma plataforma robô quadrúpede capaz de se movimentar com a ajuda de três servomotores presentes em cada pata. Cada servomotor proporciona um grau de liberdade, portanto são três graus de liberdade em cada membro. O sistema de locomoção precisou ser reformulado com relação ao utilizado em trabalhos passados, pois os servomotores foram substituidos por um novo modelo, o HS-755HB. Um nova nova maneira de se gerar os sinais de MLP foi estabelecida. Esse novo método proporciona um aumento de capacidade de controle de três, para até oito servomotores.

O sistema de comunicação foi padronizado em um protocolo formulado em trabalhos anteriores. Esse protocolo proporcionou uma melhor performance na comunicação, economizando dados a serem transmitidos e possibilitando o controle da plataforma quadrúpede de maneira mais eficiente e robusta.

Com o sistema de geração de sinais de MLP e comunicação protocolada, foi possível a criação de um sistema de calibração. O sistema de calibração foi feito para que a plataforma do robô quadrúpede se tornasse mais flexível a alterações, isto é, na troca de servomotores, carcassa mecânica ou microcontrolador. Sob qualquer dessas mudanças é possível utilizar o programa de calibração para se reestabelecer o controle dos servomotores. Esse programa foi desenvolvido de maneira a se tornar o mais “universal” possível, tornando-se útil até em outros sistemas que se utilizam de servomotores.

O sistema de posicionamento, por sua vez, teve como objetivo estimar grandezas que pudessem qualificar os movimentos executados pelo sistema de locomoção. A aceleração, a velocidade, a posição e a inclinação foram as grandezas estimadas para esse fim. Para tal, uma série de algoritmos foram implementados. Tais algoritmos podem ser divididos em dois tipos principais: algoritmos de posicionamento e algoritmos de orientação.

Os algoritmos de posicionamento foram previstos pensando-se que a aceleração da gravidade não influenciaria as medições do robô, se seu acelerômetro estivesse calibrado. O erro nessa previsão custou a obtenção de medidas inacuradas de aceleração, que, por sua vez, originaram medidas incoerentes de velocidade e de posição: todos eles decorrentes do efeito da aceleração da gravidade.

Desse modo, algoritmos de posicionamento que atuem sobre objetos que percorrem trajetórias 3d devem possuir um meio de estimar e remover instantaneamente a influência da gravidade nas medições de aceleração. Esse meio não foi encontrado pela aplicação do sistema de sensoriamento desse projeto. Cabe ressaltar que um dos motivos que certamente proporcionou o insucesso na construção desse sistema é a escassez de trabalhos que versam sobre o posicionamento de objetos em ambientes 3d através de acelerômetros.

Quanto aos algoritmos de inclinação, eles se mostram satisfatórios. Estimativas de inclinação podem ser obtidas e, a partir delas, validar sequências de movimentos feitos pela plataforma quadrúpede.

Com relação ao algoritmo de aprendizagem, deve-se refinar os dados obtidos pelo acelerômetro para que um treinamento adequado possa ser feito. Apesar do algoritmo já ter sido implementado, se for utilizado os dados do acelerômetro na condição atual, os resultados serão inconsistentes com a realidade. Foi observado que o barramento de comunicação ficou sobrecarregado ao utilizá-lo para enviar os dados

referêntes ao acelerômetro e os comandos de posições dos servomotores ao mesmo tempo, limitando o funcionamento de ambos algoritmos.

4.1 PERSPECTIVAS

Novos algoritmos de controle e processamento dos dados do acelerômetro devem ser feitos. Principalmente quando se diz respeito à eliminação/suavização do efeito da aceleração gravitacional. Lembrando que para a validação dos dados do algoritmo de treinamento proposto, não é necessário um valor preciso, bastando apenas que sejam coerêntes com a realidade.

Outros algoritmos de treinamento devem ser explorados, já que a plataforma está adaptada e padronizada. Para tal, deve-se verificar se não existe algum algoritmo novo que se aplique de maneira mais eficiente para o caso de aprendizagem com valores contínuos (no caso dos tempos de ativação de cada pata) e para sistemas não supervisionados.

O sistema de comunicação apresenta uma ineficiência que só se tornou a tona ao final do projeto. Ao tentar implementar o algoritmo de movimento junto ao acelerômetro, percebeu-se que ele interferia na temporização do sistema de locomoção, pois ocupava parte da banda de transmissão serial. Para solucionar tal problema, sugere-se gravar a sequência de movimentos dentro da EEPROM do ATMega8 e a carregar na RAM interna do microcontrolador toda vez que for iniciado. Desta forma, para iniciar o movimento é necessário apenas o envio de um comando de INICIO. Isso faria com que o sistema de locomoção ficasse independente da porta serial e portanto liberando para um maior fluxo e dados e controle do sistema.

O algoritmo desenvolvido para gerar a sequência de movimentos utilizou-se de uma taxa de envio de dados constante (a amostragem de posicionamento dos servomotores era constante). Talvez para um movimento mais efiente seja necessário aumentar ou dimunir a taxa de envio durante o movimento. Por exemplo: aumentar a taxa de envio de dados quando o membro do quadrúpede encontra-se no ar, sem contato com o solo. Desta forma diminuindo a instabilidade dinâmica gerada quando o corpo do quadrúpede está em queda e possibilitando um caminhar mais suave e eficiente. Se possível aprimorar também a sequência base de movimento individual da pata e a sequência em que os membros se movimentam.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] RANDAZZO, A. *ST485: AN RS-485 BASED INTERFACE WITH LOWER DATA BIT ERRORS*. Disponível em <http://www.st.com/stonline/products/literature/an/7628.pdf>: Application Note 134, 2004.

[2] CALMON, A. du P.; PINHEIRO, N. C.; FERREIRA, R. U. *Desenvolvimento de um robô-cachorro comportamental: percepção e modelagem comportamental*. Trabalho de conclusão de curso de graduação em Engenharia Elétrica: [s.n.], 2006.

[3] SEIFERT, K.; CAMACHO, O. Implementing positioning algorithms using accelerometers. *Freescale Semiconductor*, AN3397 Rev 0, 2007.

[4] SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998. ISBN 0262193981. Disponível em: <http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html>.

[5] BEKEY, G. A. *Autonomous Robots: From Biological Inspiration to Implementation and Control (Intelligent Robotics and Autonomous Agents)*. [S.l.]: The MIT Press, 2005. ISBN 0262025787.

[6] SOUTO, R. F. *Modelagem cinemática de um robô quadrúpede e geração de seus movimentos usando filtragem estocástica*. Trabalho de conclusão de curso de graduação em Engenharia Elétrica: [s.n.], 2007.

[7] KOHL, N.; STONE, P. Policy gradient reinforcement learning for fast quadrupedal locomotion. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2004.

[8] COTTA, G. H.; NETO, L. R. *Realização de uma plataforma para estudo de robótica comportamental baseada em quadrúpedes*. Trabalho de conclusão de curso de graduação em Engenharia Elétrica: [s.n.], 2006.

[9] ATMEL. *ATMEGA8 Complete Reference Manual*. [S.l.], 2003.

[10] BORGES, G. A.; MARTINS, A. S. *Introdução ao padrão físico RS-485 para comunicação serial*. Disponível em: <http://www.ene.unb.br/ gaborges/recursos/notas/index.htm>: [s.n.], Junho 2006.

[11] ZHIFENG, C. et al. The cpg-based bionic quadruped system. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '03)*. [S.l.: s.n.], 2003. p. 1828–1833.

[12] BORGES, G. A. et al. *Desenvolvimento com microcontroladores Atmel AVR*. Disponível em: <http://www.ene.unb.br/ gaborges/recursos/embarcados/index.htm>: [s.n.], Junho 2006.

[13] CLIFFORD, M.; GOMEZ, L. Measuring tilt with low-g accelerometers. *Freescale Semiconductor*, AN3107 Rev 0, 2005.

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

Os circuitos desenvolvidos para o sistema de sensoriamento do robô-quadrúpede são apresentados nesse Anexo. A Figura I.1 mostra o diagrama esquemático desenvolvido com o emprego do acelerômetro MMA7260Q. Tal figura é referente ao diagrama de blocos implementado na Seção 2.3.7. Já a Figura I.2 mostra o diagrama esquemático desenvolvido com o emprego do acelerômetro LIS3LV02Q. Tal figura é referente ao diagrama de blocos implementado na Seção 2.3.4

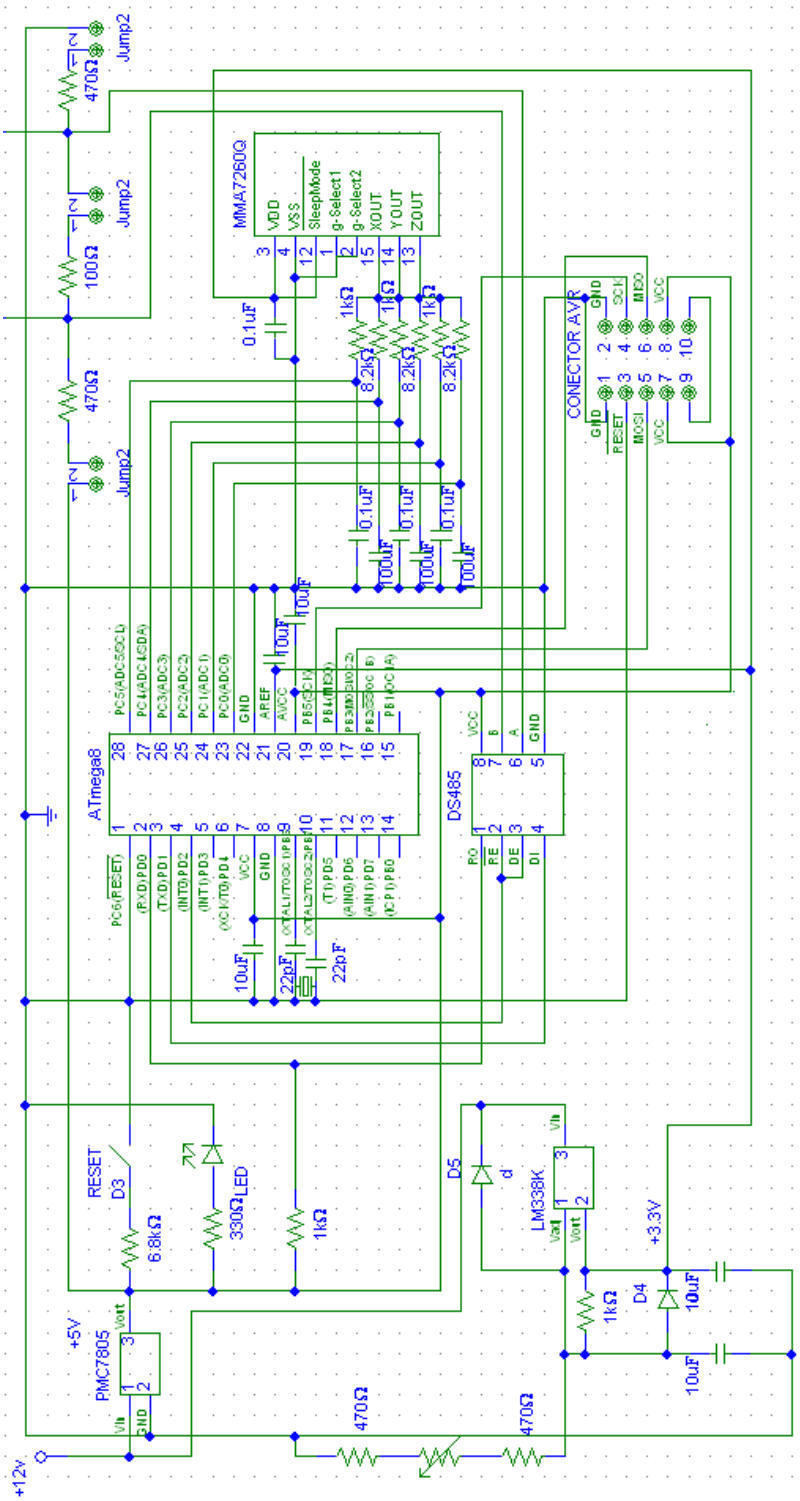


Figura I.1: Diagrama esquemático do sistema de sensoriamento para o MMA7260Q

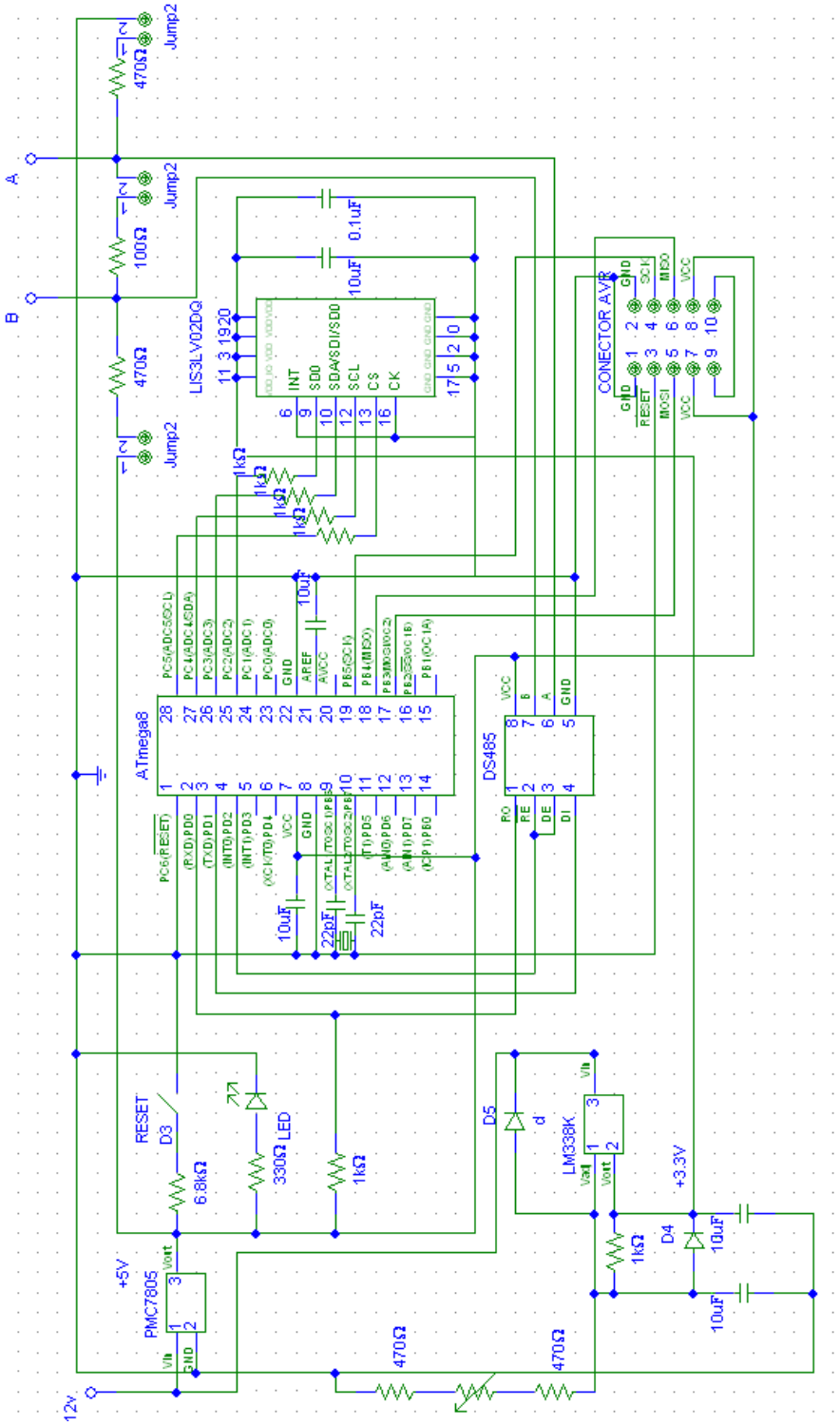


Figura I.2: Diagrama esquemático do sistema de sensoriamento para o LIS3LV02Q

II. DESCRIÇÃO DO CONTEÚDO DO CD

No CD estão presentes três diretórios principais: Vídeos, Fotos e Programas.

Em Vídeos está contido os vídeos do robô em seu treinamento no Experimento 2 e 3 e o utilizado para se testar o inclinômetro. O nome de cada arquivo descreve o processo filmado.

Em Fotos estão as fotos tiradas da plataforma quadrúpede recentemente e no início do projeto. No início do projeto não havia o acelerômetro instalado e o suporte das placas das microcontroladoras eram de *isopor*. Nas fotos recentes pode-se observar que os circuitos estão suspensos. Estão presentes também fotos do posicionamento do servomotor durante a calibração.

No diretório dos programas estão presentes todos os programas desenvolvidos nesse projeto, sendo: ATMega8 Servos o diretório que contém os programas referentes ao controle dos servomotores para o ATMega8, IBM-PC o diretório com os programas que controlam o quadrúpede no IBM-PC e ATMega8 Acelerômetro os programas que controlam o Acelerômetro através do ATMega8.

Programas contidos no diretório ATMega8 Servos:

- O único programa contido no diretório é o algoritmo que controla os servomotores através dos sinais de MLP gerados pelo ATMega8 e que codifica/decodifica o protocolo de comunicação estabelecido em [2].

Existem duas versões de programas contidos no diretório ATMega8 Acelerômetro:

- O primeiro é o Acelerômetro Completo que contém uma versão que não compila por falta de espaço na memória *Flash* do ATMega8. Essa versão contém o cálculo de todas as variáveis necessárias ao algoritmo de aprendizagem: espaço percorrido em x e y, velocidade no eixo x e y, e a inclinação no eixo z.
- O segundo é o Acelerômetro Compacto que contém a versão do programa que é compilável e portanto pode ser gravada no ATMega8. Essa versão processa os dados recebidos do acelerômetro e estima a velocidade no eixo x e y do robô.

Os programas contidos no IBM-PC são os seguintes:

- calibrador.c : Programa que efetua a calibração de todos os servomotores presentes no quadrúpede. Esse programa também grava na EEPROM do ATMega8 a posição inicial de cada servomotor ao se ligar a plataforma.
- testador.c : Programa que lê de um arquivo texto a sequência de movimentos dos servomotores. Os dados lidos são enviados continuamente através do barramento serial até que o usuário interrompa o programa.
- treinador.c : Algoritmo de treinamento que utiliza do passeio aleatório e dos dados recebidos do acelerômetro para qualificar o movimento do robô quadrúpede.

- teste_acelerometro.c : Programa que solicita as estimativas de espaço, velocidade e inclinação do acelerômetro. Os dados são recebidos via barramento serial e então armazenados em arquivos tipo texto (*.txt*). Esses dados são analisados em programas de processamento matemático para averiguar a qualidade da informação recebida, possibilitando uma análise mais detalhada dessa e o desenvolvimento de técnicas que aprimorem os algoritmos de posicionamento e inclinação existentes.