Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Eletrônica e de Computação

Serpentina: desenvolvimento de um robô serpente telecomandado

Autor:
Flavia Correia Tovo

Orientador:
Prof. José Gabriel Rodriguez Carneiro Gomes, Ph.D.

Examinador:
Prof. Antonio Petraglia, Ph.D.

Examinador:
Prof. Antônio Cláudio Gomez de Sousa, M.Sc.

DEL
Junho de 2011

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

π

Agradecimentos

Primeiramente gostaria de agradecer a CAPES e a KAUST por terem financiado o meu intercâmbio, durante o qual tive a oportunidade de desenvolver esse projeto. Em segundo lugar, gostaria de agradecer ao Renato que trabalhou comigo no projeto e aos professores Alexis e Samuel que nos orientaram.

Igualmente, gostaria de agradecer aos professores do curso de Engenharia Eletrônica e de Computação pela Universidade Federal do Rio de Janeiro e aos professores da École d'Ingénieurs Télécom ParisTech por contribuir para a minha formação.

Agradeço aos meus colegas de trabalho do Laboratório de Processamento Analógico e Digital de Sinais (PADS - COPPE - UFRJ), onde tive a oportunidade de fazer iniciação cientifica antes do intercâmbio. Especialmente presto meu agradecimento a professora Mariane Rembold Petraglia por ter me orientado durante esse período. Agradeço também ao professor José Gabriel Rodriguez Carneiro Gomes por ter aceitado orientar esse trabalho.

Gostaria de agradecer também aos meus colegas da turma 2005/1 pelos anos que passamos juntos, por todas as dificuldades que fomos capazes de superar como um grupo, pelos trabalhos desenvolvidos em conjunto, pelo tempo passado juntos e pelos jogos de boliche.

Agradeço igualmente ao Discovery Channel que me serviu de inspiração para a vontade de trabalhar com robótica e especialmente ao meu tio Alexandre por ter sugerido o curso de eletrônica como um bom caminho para trabalhar nessa área: "já que se eu não gostasse, poderia facilmente trocar de área".

Finalmente, gostaria de agradecer à minha família, especialmente meus pais, que sempre me acompanharam, apoiaram e incentivaram. Um agradecimento especial para o meu marido Pietro por todo o apoio e ajuda que ele tem me dado desde o meu primeiro período de curso.

Resumo do Projeto Final apresentado Escola Politécnica - UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação

Serpentina: desenvolvimento de um robô serpente telecomandado

Flavia Correia Tovo

Junho/2011

Orientador: José Gabriel Rodriguez Carneiro Gomes

Departamento de Eletrônica e de Computação

Este trabalho apresenta o projeto e o desenvolvimento de um robô serpente telecomandado. O sistema é composto de duas partes: o robô e o sistema de controle. O robô é composto de dez vértebras que são conectadas mecanicamente umas as outras, compostas de duas partes móveis ligadas a dois servomotores e uma placa de circuito impresso. O sistema de controle é composto de uma interface gráfica e uma placa de controle. Para se locomover, o robô é mapeado numa senóide que se desloca.

Palavras Chave: robótica, robô serpente, computação distribuída, servomotor.

Abstract of Final Project presented to Escola Politécnica - UFRJ as a partial fulfillment of the requirements for the degree of Eletronic and Computer Engineer.

Serpentina: development of a remotely operated snake robot

Flavia Correia Tovo

June/2011

Advisor: José Gabriel Rodriguez Carneiro Gomes

Eletronic and Computer Departament

This work presents the design and development of a robot snake remotely operated. The system consists of two parts: the robot system and the control system. The robot is composed of ten vertebrae that are mechanically connected to each other, consisting of two movable parts connected to two servomotors and a PCB. The control system consists of a graphical interface and a control card. To get around, the robot is mapped into a sinusoidal wave that shifts.

Index Terms: robotics, snake robot, distributed computing, servomotor.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO I

INTRODUÇÃO

I.1 - MOTIVAÇÃO

Desde a atiguidade, robôs fazem parte do imaginário das pessoas. Com o passar do tempo, robôs começaram a ser desenvolvidos e a cada dia mais e mais robôs participam da nossa rotina. Inicialmente desenvolvidos para nos ajudar em tarefas como plantio de grãos e elevação de água, atualmente os encontramos nas mais diversas atividades, desde aplicações industriais a robôs que aspiram o chão ou ajudam idosos nas tarefas domésticas.

Um robô é uma máquina guiada automaticamente, capaz de executar tarefas por conta própria. Robótica é um ramo da tecnologia que lida com a concepção, construção, operação, disposição estrutural, fabricação e aplicação de robôs. Robótica está relacionada com as ciências da eletrônica, engenharia, mecânica e computação [1].

Robôs são compostos por vários sistemas embarcados. Um sistema embarcado é um sistema projetado para fazer uma ou algumas funções específicas e/ou dedicadas, frequentemente com restrições de computação em tempo real. Ele é incorporado como parte de um dispositivo completo, muitas vezes incluindo sistemas eletrônicos e mecânicos. Sistemas embarcados são controlados por um ou mais núcleos de processamento principal, que são tipicamente de microprocessadores ou processadores de sinais digitais [2]. A característica principal, entretanto, é ser dedicado, ou seja, executar uma tarefa particular.

Pequenos robôs têm o potencial para acessar espaços confinados, onde os humanos não podem ir. Entretanto, a mobilidade dos sistemas com rodas é limitada em ambientes desordenados. Robôs serpentes usando modos de locomoção biologicamente inspirados podem proporcionar um melhor acesso em muitas situações [3]. Eles podem usar seus muitos graus internos de liberdade para percorrer ambientes estreitos onde homens e outras maquinas não podem ir.

Além disso, esses dispositivos altamente articulados podem coordenar os seus graus internos de liberdade para executar uma variedade de movimentos, que vão além da

capacidade dos robôs convencionais com rodas e dos recentemente desenvolvidos robôs com pernas. O verdadeiro poder desses dispositivos está na sua versatilidade, alcançando comportamentos não limitados para rastejar, escalar e nadar. [4]

Robôs serpentes são mais úteis em situações onde as suas características únicas lhes dão uma vantagem sobre o ambiente. Esses ambientes tendem a ser longos e finos, como tubulações ou altamente desordenados como entulho [5]. Entre as diversas aplicações de robôs serpente estão: a busca e resgate em destroços [6] e microcirurgias cardíacas [7].

I.2 - Objetivo

O objetivo do projeto é o desenvolvimento de um robô serpente telecomandado e seu sistema de controle. Desta forma, tem-se como objetivos específicos: (1) desenvolvimento do sistema de controle e sua interface computacional; (2) projeto e fabricação dos circuitos impressos presentes nas vértebras do robô , e; (3) desenvolvimento de um sistema distribuído para o controle da trajetória do robô.

I.3 - Escolha do nome do projeto

O projeto é um robô serpente que foi desenvolvido por um grupo de brasileiros em intercâmbio na França. Para fazer uma referência ao Brasil, foi escolhido o nome Serpentina para o projeto do robô. Após a escolha do nome, uma fonte foi escolhida para lembrar o movimento das serpentinas quando jogadas. O resultado final é mostrado na Figura I.1.

Figura I.1: Logotipo criado para o robô

I.4 - Organização do Trabalho

Esse trabalho descreve o projeto e o desenvolvimento de um robô serpente telecomandado. O projeto foi desenvolvido entre 15 de fevereiro e 7 de maio de 2010, em dupla com o aluno Renato M. Honda (USP) sob orientação dos professores Alexis Polti e Samuel Tardieu durante o intercâmbio de duplo diploma com a TELECOM ParisTech.

O Capítulo II apresenta a descriçao completa do robô serpente. Na Seção II.1, a estrutura fisica do robô é descrita. Na Seção II.2, o projeto dos circuitos impressos presentes nas vértebras é apresentado. A Seções II.3 e II.4 são relacionadas ao software do

robô, sendo na Seção II.4 abordados somente os aspectos ligados ao movimento do robô. Na Seção II.5, são abordados os sistemas de comunicação empregados no robô.

No Capítulo III, são descritos os assuntos relacionados ao sistema de controle. Na Seção III.1, a arquitetura da placa de controle é descrita. A Seção III.2 diz respeito ao software do sistema de controle, descrevendo sua interface, Seção III.2.2, e os comandos disponíveis, Seção III.2.3. Na Seção III.3, são descritos os princípios usados no protocolo de comunicação usado na comunicação entre o sistema de controle e o robô.

As conclusões do trabalho são apresentadas no Capítulo IV.

CAPÍTULO II

Robô

II.1 - Estrutura física

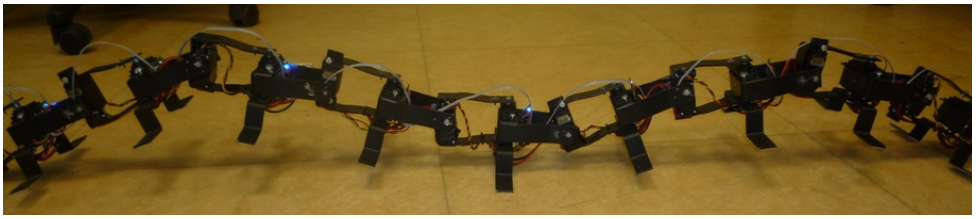


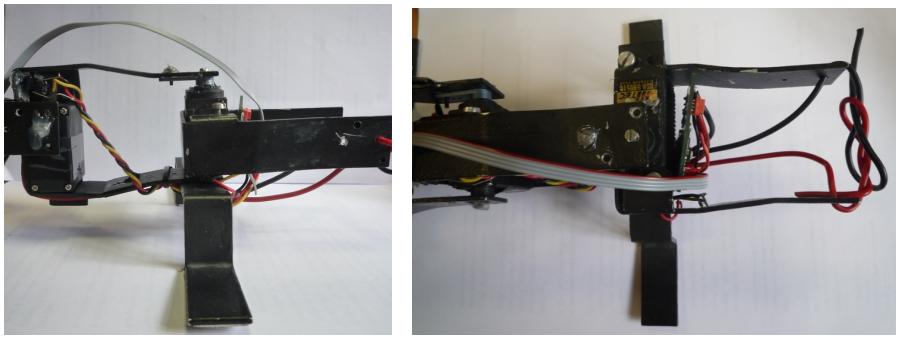
Figura II.1: Vista lateral do robô

O robô, como mostrado na Figura II.1, é composto de 10 vértebras que são conectadas mecanicamente umas às outras. Cada vértebra mede 15 cm de comprimento e é composta de duas partes móveis ligadas a dois servomotores[1] e uma placa de circuito impresso, descrito na Seção II.2, usada para o controle dos servomotores e comunicação entre as vértebras.

O servomotor que se encontra na extremidade da vértebra, visível na Figura II.2(a), é responsável pelos movimentos no plano vertical. A associação dos movimentos desse motor em todas as vértebras possibilita o deslocamento retilíneo do robô. O segundo servomotor que se encontra no meio da vértebra, visível na Figura II.2(b), faz a ligação entre duas partes moveis e é responsável pelos movimentos no plano horizontal, o que possibilita que o robô faça trajetórias curvilíneas.

Como o objetivo dos servomotores no robô é modificar o ângulo entre duas vértebras para compor o movimento da serpente, a última vértebra não possui motor para o movimento vertical e a primeira vértebra não possui motor para o movimento horizontal.

[1]Servomotor é uma máquina, mecânica ou eletromecânica, que apresenta movimento proporcional a um comando, em vez de girar ou se mover livremente sem um controle mais efetivo de posição como a maioria dos motores; servomotores são dispositivos de malha fechada, ou seja: recebem um sinal de controle; verificam a posição atual; atuam no sistema indo para a posição desejada. [8]



(a) Vista lateral de uma vértebra (b) Vista superior de uma vértebra

Figura II.2: Vistas lateral e superior de uma vértebra

Sendo para as 10 vértebras: 9 servomotores para movimentos verticais e 9 servomotores para movimentos horizontais.

Além da ligação mecânica entre as vértebras, um cabo de alimentação (5V) e um barramento CAN (fornecedor de uma segunda alimentação de 3,3V), tratado na Seção II.5.2, são ligados a cada um dos circuitos das vértebras.

II.2 - Arquitetura

Para definir a arquitetura do robô é necessário analisar que funções desejamos que ele seja capaz de executar e como. A tarefa mais importante do robô é: se locomover seguindo instruções de um operador. Para isso várias abordagens são possíveis, podemos ter um sistema de controle externo que, dada uma entrada do operador, calcula os movimentos que cada vértebra deve seguir para percorrer a trajetória determinada e os envia para a vértebra correspondente.

Outra abordagem possível é as ordens do operador serem transmitidas diretamente para o robô, e este, possuindo uma unidade de cálculo central, calcula os movimentos de cada vértebra distribuindo-os para as mesmas. Ou ainda, as ordens do operador são enviadas para o robô, que distribui essa informação entre suas vértebras e cada vértebra calcula seus próprios movimentos para que o robô como um todo possa executar o que foi determinado.

A primeira alternativa exige uma grande taxa de transmissão de informações entre o sistema de controle e o robô, o que pode atrasar a resposta do robô ou provocar uma resposta errônea se o ambiente onde este se encontra não for favorável à transmissão dessas mensagens. A segunda alternativa é mais adaptada para as situações descritas, mas o bom funcionamento do robô depende do bom funcionamento da sua unidade de controle centralizado, isso quer dizer, se essa parte apresentar um problema, todo o robô é inutilizado. A terceira abordagem é a mais robusta, se as vértebras forem indiferenciáveis, caso algumas vértebras apresentarem problemas ou o robô for dividido mecanicamente em

várias partes, essas partes podem continuar a funcionar como o robô original.

Por ser a solução mais robusta, a solução adotada foi a opção em que cada vértebra é responsável por calcular o seu próprio movimento e repassar as informações às demais vértebras. Definida essa parte essencial sobre a forma de funcionamento do robô, outras tarefas precisam ser levadas em conta. É necessário que o robô como um todo seja capaz de receber as mensagens de controle e que cada vértebra seja capaz de:

- se comunicar com as outras vértebras;
- calcular seus próprios movimentos;
- controlar os servomotores de sua propriedade;
- sinalizar quaisquer problemas de operação;
- permitir que um agente externo a reinicialize em caso de necessidade.

O modelo entrada/saída do circuito das vértebras é mostrado na Figura II.3.

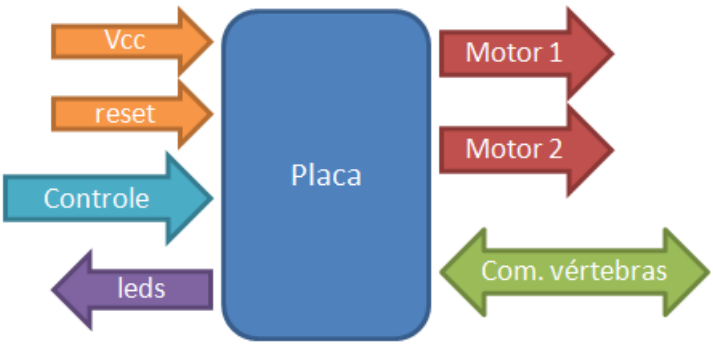


Figura II.3: Arquitetura definida

II.2.1 - Escolha dos componentes

O processador escolhido foi o STM32F103CBT6 [9]. O STM32 dispõe de vários manuais que ajudam o desenvolvimento de aplicações e o entendimento do seu funcionamento [10][11][12][13]. O processador apresenta várias saídas em PWM, conversores ADC com 12 bits de resolução e 1μs de tempo de conversão e apresenta modos de baixo consumo, o que faz dele um bom processador para sistemas embarcados que exigem controle de motores.

Como o sistema foi descrito, a melhor configuração de barramento para a comunicação entre as vértebras é uma que possua múltiplos mestres. Pela sua robustez, baixa latência e característica de vários mestres e vários escravos, foi escolhido o barramento CAN, Seção II.5.2, com o controlador MCP2551 [14] conectado a uma porta série pela qual é feita a conexão do barramento.

Para que o robô pudesse se comunicar com o sistema de controle, foi utilizado um módulo que implementa o protocolo ZigBee; para conectar o ZigBee ao circuito, uma segunda porta série foi colocada. Essa porta pode ser usada tanto para um *bootloader* como para o ZigBee.

Os servomotores são utilizados em modo standard [15], logo os sinais são modulados em PWM diretamente por um timer do processador, sendo necessária apenas a colocação de jumpers para a conexão dos cabos. O modo de controle do servomotor é descrito na Seção II.3.3, e o servomotor propriamente dito é descrito no Apêndice B. Ainda em relação com os motores, dois amplificadores diferenciais por resistência de shunt foram incluídos para a medição da corrente consumida pelos motores.

Dois conjuntos de leds coloridos programáveis tipo RGB foram adicionados para sinalizar o estado do circuito, um conjunto de leds é usado para sinalizar o tipo de trajetória do robô (para frente ou para trás, executando ou não uma curva) e o outro para sinalizar o estado de funcionamento do robô, sendo um led piscante a 10Hz o sinal de que a vértebra está funcionando.

Para a reinicialização do circuito foi colocado um botão, e para escolha do modo de operação do processador, um switch.

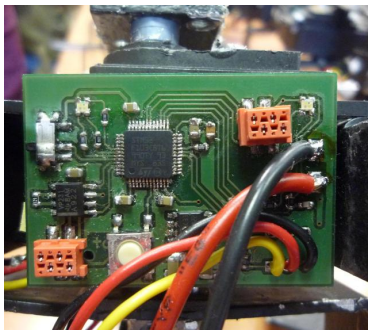


Figura II.4: Versão final da placa usada em cada vértebra.

A configuração final da placa das vértebras é mostrada na Figura II.4 e seu esquemático é mostrado no Apêndice I.1.

II.3 - Sistema e funcionamento

O sistema é composto de dez vértebras, cada uma possuindo um circuito, que trabalham em paralelo, cada uma controlando seus respectivos motores. Um módulo ZigBee é conectado a uma vértebra e esta tem o papel de líder do robô. A perda da vértebra conectada ao ZigBee significa a perda do controle do robô. Em uma versão final, cada vértebra deverá possuir um módulo ZigBee, o que a possibilita de assumir o papel de líder do robô.

II.3.1 - Estrutura do software

O sistema operacional FreeRTOS [16] é usado no projeto. FreeRTOS é um sistema operacional em tempo real para aplicações em sistemas embarcados, sendo compatível com o processador escolhido para o projeto. O software é dividido em várias tarefas principais e outras secundárias. Essas tarefas são principalmente responsáveis por:

- parar, avançar ou recuar o robô;
- executar movimento curvilíneo do robô ou fazê-lo voltar à posição inicial;
- identificar as placas presentes;
- receber comandos do computador;
- receber comandos da placa mestre ou de outras placas;
- executar um movimento automático para frente e para trás.

II.3.1.1 - Movimento retilíneo

Uma das tarefas principais é a que faz o controle do servomotor e que permite que o robô se desloque para frente e para trás (movimento vertical). Enquanto o operador não escolhe a opção de deslocamento, o robô não se move. Se uma mensagem de deslocamento é recebida, todas as vértebras, exceto a última, se colocam em espera de informações sobre a sua posição.

A última vértebra, que é uma das pontas do robô e muda de acordo com o sentido do movimento, começa o cálculo do algoritmo da trajetória (Seção II.4.1) e ao repassar os resultados da sua operação à vértebra vizinha, vai desencadear o cálculo da mesma e esta fará o mesmo para a seguinte. Assim que um circuito possui as informações necessárias para encontrar o ângulo para o qual o servomotor deve se mover, ele envia o sinal PWM correspondente. Portanto o movimento é feito a partir de uma propagação de posições e ele é executado do final para o inicio do robô.

II.3.1.2 - Movimento curvilíneo

Uma segunda tarefa principal é responsável pala execução de uma curva à direita ou à esquerda. Isso é feito por uma propagação da curva. Isso quer dizer que quando a placa líder recebe um comando de virar à direita, à esquerda, ou de voltar a um movimento retilíneo, ela envia esse comando à vértebra que está na primeira posição do robô. Em seguida essa mensagem é propagada para as outras placas.

A primeira placa vai executar a ação desejada e enviar o mesmo comando para a vértebra seguinte. Essa vértebra espera um número variável de passos (dependendo da velocidade de deslocamento do robô) antes de executar também a ação. Assim que ela o

executa, o comando é enviado à vértebra seguinte. Isso se repete até a última vértebra. Esse movimento se executa do início para o final do robô.

II.3.1.3 - Identificação dinâmica de presença

Uma outra tarefa principal é a identificação dinâmica da presença das vértebras. A placa mestre, aquela conectada ao ZigBee, envia mensagens de verificação de presença. Quando as outras vértebras recebem essas mensagens, elas enviam uma resposta. A placa mestre verifica quais vértebras ainda estão presentes e, escolhendo o maior conjunto de vértebras conectadas sequencialmente, as identificações das primeira e última vértebras ativas são enviadas a todas as vértebras.

O conhecimento das identificações das extremidades do robô é importante pois a vértebra de uma das extremidades é a responsável por desencadear todos os cálculos.

II.3.1.4 - Comunicação com o sistema de controle

A todo momento o operador pode enviar comandos para o robô. É sempre a mesma placa que vai recebê-los, pois ela é a única a possuir um módulo ZigBee configurado. Dependendo do estado do movimento do robô, assim como do tipo de comando, essa placa pode aceitá-lo ou recusá-lo. Por exemplo se o robô se desloca em um sentido e o operador envia um comando para que ele se desloque no outro sentido, ele tem que enviar primeiro o comando de parada e em seguida enviar o novo sentido de deslocamento.

Os comandos de deslocamento foram implementados como sendo:

- deslocamento para frente e para trás e ordem de parada;
- virar à direita, à esquerda ou voltar a um deslocamento retilíneo.

Existem também os comandos que modificam as configurações da execução:

- tempo de espera entre dois cálculos: isso modifica a velocidade de deslocamento do robô;
- a amplitude e o comprimento da senóide na qual o robô é mapeado;
- o ângulo da uma curva.

II.3.1.5 - Ordens vindas de outras placas

Duas tarefas são responsáveis por receber as mensagens vindas pelo barramento CAN. Uma primeira tarefa filtra as mensagens recebidas e trata aquelas na qual o destinatário é a própria placa. Uma segunda tarefa é responsável pelo tratamento das mensagens enviadas em *broadcast*.

Esses filtros são utilizados no momento em que uma mensagem chega. Só haverá interrupção se a mensagem se enquadrar em uma dessas possibilidades e somente a interrupção respectiva será ativada.

II.3.2 - Sistema distribuído

As dez vértebras que compõem o robô formam um sistema distribuído onde a única forma de comunicação entre elas é a troca de mensagens pelo CAN. Durante a execução, a placa conectada ao módulo ZigBee trabalha como placa mestre, sendo responsável pela comunicação do robô com o sistema de controle.

Como um sistema distribuído, a informação de quais células ainda compõem o sistema é uma informação fundamental. A placa mestre também é responsável por iniciar o sistema de identificação dinâmica e de enviar a identificação das vértebras das extremidades da parte ativa do robô para a totalidade do robô. Como o movimento sempre começa por uma das extremidades do robô, a vértebra que está nessa posição, tendo essa informação, pode iniciar o cálculo que fará com que o robô possa se locomover.

A última vértebra, sabendo que está na extremidade, calcula o movimento do seu motor. Ao terminar esse cálculo ela repassa a informação de posição calculada para a vértebra vizinha, que vai usar essa informação para poder calcular o seu próprio ângulo.

Com exceção das informações de mudança de parâmetros da senóide usada para mapear o movimento do robô e informações sobre o sentido do movimento, cada vértebra depende somente das informações fornecidas pelas vértebras vizinhas a ela.

II.3.3 - Controle dos motores

As principais informações do servomotor e o modo de funcionamento escolhido são descritas no Apêndice B, para maiores informações sobre outros modos de funcionamentos ver [15].

O modo de operação escolhido para o servomotor foi o standard, o que faz com que o sinal de controle enviado para o motor seja um sinal em PWM. Dentro do servomotor, esse sinal é usado através de uma ponte H.

II.3.3.1 - PWM

A modulação por largura de pulso, em inglês pulse-width modulation, é um método permitindo a geração de sinais analógicos contínuos a partir de saídas de circuitos digitais, como microprocessadores.

O interesse do PWM nesse trabalho é exatamente poder gerar um sinal contínuo de controle dos servomotores a partir da saída de um sistema digital. O princípio é utilizar o filtro passa baixa natural dos sistemas (como a inércia de um motor) para compor o

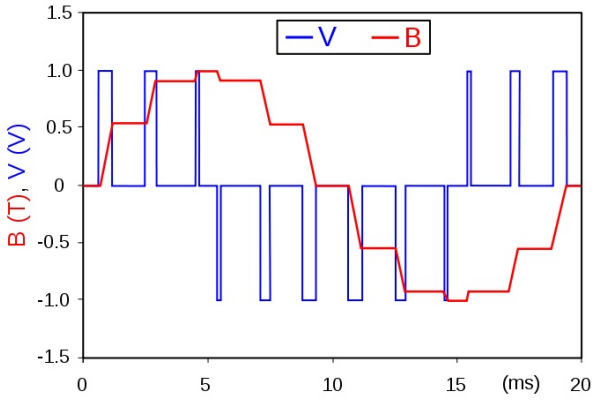


Figura II.5: Exemplo de PWM em um motor CA. FONTE [17]

sinal analógico. Assim, a partir da sucessão de estados discretos durante um tempo T, podemos obter um valor continuo médio.

Esse valor médio fornecido à carga é controlado pelo acionamento ou desligamento de um interruptor entre a carga e a alimentação. Quanto maior o tempo em que o interruptor permanece ligado comparado com o tempo em que permanece desligado, maior a potência fornecida à carga.

A frequência de acionamento do interruptor tem que ser muito maior do que a frequência do sinal analógico almejado. O termo *duty cycle* descreve a proporção de tempo no qual o interruptor está ligado com relação ao período. Um *duty cycle* pequeno corresponde à baixa potência, pois o interruptor fica desligado a maior parte do tempo , e um *duty cycle* grande à alta potência. *Duty cycle* é expresso em porcentagem.

A maior vantagem do PWM é que a perda de energia do interruptor é baixa. Quando o interruptor está desligado, a corrente é praticamente nula e quando ele está ligado, a queda de tensão no dispositivo é quase nula.

Um exemplo de PWM em um motor de corrente alternada é mostrado na Figura II.5. O sinal fase-fase, em azul, é modulado como uma série de pulsos que resultam numa forma de onda senoidal como a densidade de fluxo no circuito magnético do motor, em vermelho. A suavidade da onda resultante pode ser controlada pela largura e número de impulsos modulados por determinado ciclo.

II.4 - Deslocamento

Como mostrado na Figura II.6, as cobras possuem quatro diferentes métodos de deslocamento [18]:

- sanfonado (concertina),
- serpentino (serpentine),
- ondulação lateral (sidewinding), e

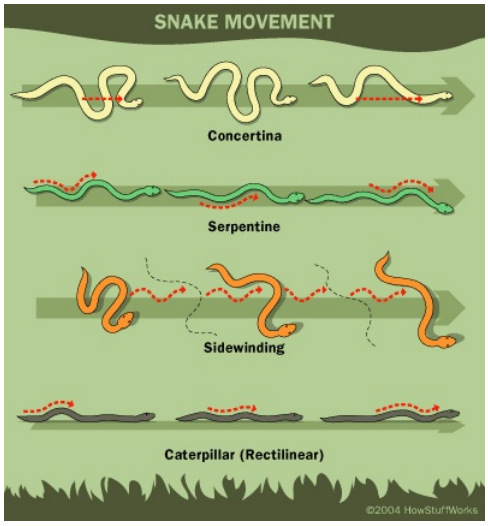


Figura II.6: Tipos de deslocamento de cobras. FONTE [18]

- retilíneo (caterpillar or rectilinear).

O movimento sanfonado é o movimento utilizado quando as cobras escalam, os outros movimentos não se adequam para essa função. A cobra estende a cabeça e a frente do corpo ao longo da superfície vertical e então encontra um lugar para agarrar com suas escamas ventrais.

O movimento serpentino é o movimento em forma de S. Ele é usado pela maioria das cobras. Começando no pescoço, a cobra contrai seus músculos, impulsionando seu corpo de um lado para o outro, criando uma série de curvas.

A ondulação lateral é usada em ambientes com poucos pontos de resistência, esse movimento é uma variação do movimento serpentino. Contraindo seus músculos e arremessando o corpo, elas criam uma forma de S que tem apenas dois pontos de contato com o solo; quando se impulsionam, movem-se lateralmente.

O movimento retilíneo é um método muito mais lento para movimentar-se. Essa técnica também contrai o corpo em curvas mas essas ondas são bem menores e se curvam para cima e para baixo, ao invés de para os lados. Quando uma cobra usa o movimento retilíneo, os topos de cada curva levantam acima do solo enquanto as escamas ventrais da base empurram o chão, criando um efeito encrespado que pode ser matematicamente aproximado por uma senóide.

Pela simplicidade do modelo matemático, o modo de deslocamento escolhido para o robô foi o movimento retilíneo. Para a locomoção do robô, uma senóide é propagada pela sua estrutura, da parte de trás para a parte de frente do robô.

Como convenção adotada, a parte do robô conectada à alimentação é a parte de trás do mesmo.

II.4.1 - Cálculo da trajetória

Esse algoritmo é o responsável por que o robô possa se deslocar para frente e para trás. Para esse algoritmo, a primeira vértebra é a vértebra posicionada na parte de trás do robô, enquanto que a décima é a vértebra posicionada na frente (cabeça).

Em resumo, a idéia é mapear as vértebras do robô em uma senóide, calculando a posição de todas as vértebras nessa onda. Em seguida, o cálculo admite que um pequeno passo foi feito, ele assume que o robô vai se deslocar, e recalcula todas as posições. Se esse mapeamento é feito utilizando um pequeno intervalo de tempo entre as operações e se as vértebras respeitarem esse formato de onda, é possível observar o movimento do robô.

Para calcular as posições e chegar a mapear as vértebras, a primeira vértebra recebe uma posição fixa (por exemplo, posição zero antes do inicio do deslocamento) e a partir dessa informação as posições das duas próximas vértebras são calculadas. Para que uma vértebra possa calcular seu ângulo de movimento ela precisa de três posições. Quando essas posições são encontradas, elas são enviadas à segunda vértebra, que calcula a posição da quarta vértebra. Esse processo se repete até a última vértebra.

Como o servomotor que controla os ângulos verticais se encontra na extremidade e não no meio da vértebra, isso quer dizer que uma vértebra não é simétrica. Para fazer os cálculos para que o robô possa avançar ou recuar, é preciso usar posições diferentes de vértebras.

Assim para fazer o robô avançar, as posições necessárias para as vértebras são (x_i onde i indica a posição da vértebra na serpente):

- primeira vértebra: x_1, x_2 e x_3;
- segunda vértebra: x_2, x_3 e x_4;
- terceira vértebra: x_3, x_4 e x_5;
- ...

E para fazer o robô recuar, as posições necessárias para as vértebras são:

- décima vértebra: x_{10}, x_{10} e x_9;
- nona vértebra: x_{10}, x_9 e x_8;
- oitava vértebra: x_9, x_8 e x_7;
- ...

Para fazer o cálculo da posição da próxima vértebra, dois princípios são utilizados:

- as extremidades de cada vértebra são mapeadas na senóide;

- o comprimento de uma vértebra é constante (15cm).

Dessa forma, se admitirmos que o ponto de inicio da vértebra é (x_0,y_0) e o ponto (x_1,y_1) é sua outra extremidade, temos:

$$(x_1 - x_0)^2 + (y_1 - y_0)^2 = 15^2$$
$$(x_1 - x_0)^2 + (sin(x_1) - sin(x_0))^2 = 15^2$$

Se o ponto x_0, y_0 é conhecido, é possível encontrar o x_1, e consequentemente o y_1, de maneira interativa por aproximações.

Para realizar esse cálculo interativo, dois algoritmos foram implementados: o método da bisseção (*bisection method* [19]) e o método de Brent (*Brent's method* [20]). O método de Brent implementa o método da bisseção, das secantes (*secant method* [21]) e da interpolação quadrática inversa (*inverse quadratic interpolation* [22]). O método da bisseção é mais exato, mas ele pode precisar de muito tempo para convergir. Em contra partida, o método de Brent converge em um intervalo menor de tempo, mas ele é menos confiável. Quando o cálculo não é bom, ele é refeito pelo método da bisseção.

No final, o método da bisseção foi suficiente para a aplicação e foi o único a ser usado nos cálculos. Como o processador não possui cálculo em ponto fixo, as operações necessárias para a conversão dos cálculos também foram desenvolvidas. O algoritmo de mapeamento e o cálculo em ponto fixo foram testados em simulação, o resultado da simulação é mostrado na Figura II.7.

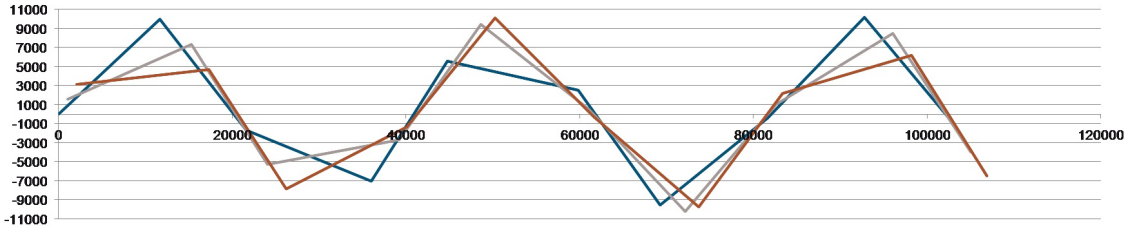


Figura II.7: Mapeamento do robô na senóide por simulação

II.4.2 - Curvas

Como explicado da Seção II.3.1.2, a placa mestre recebe a ordem de começar uma curva e repassa essa mensagem para a primeira vértebra. Essa vértebra, depois de um número fixo de passos, faz a mudança do seu ângulo horizontal e envia essa mensagem para a vértebra seguinte.

Um esquema da propagação de uma curva para quatro vértebras é mostrado na Figura II.8. Podemos observar na figura que depois que o ângulo entre duas vértebras é modificado, este permanece o mesmo até que uma ordem de execução de uma norva curva ou de voltar para uma posição retilínea seja recebida.

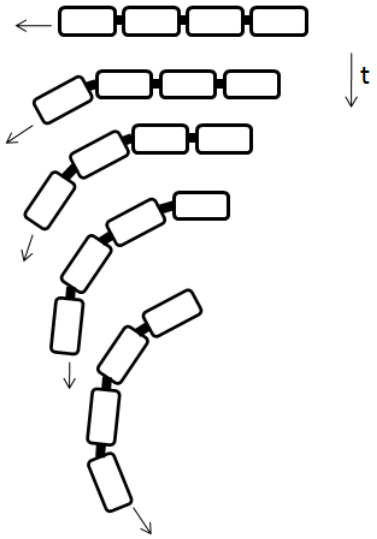


Figura II.8: Esquema de uma propagação de curva para 4 vértebras

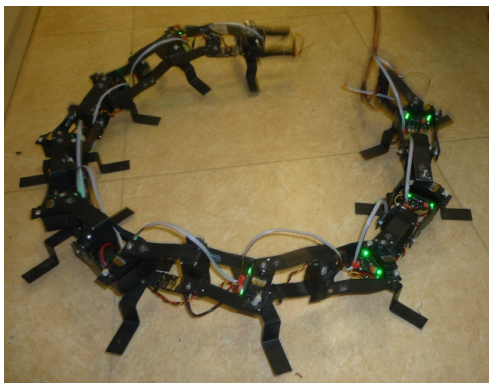


Figura II.9: Todas as vértebras executando uma curva

A característica do servomotor de manter o mesmo ângulo enquanto não for recebida a ordem de voltar para uma posição retilínea pode ser usada para execução de trajetórias circulares como mostrado na Figura II.9.

O intervalo de tempo esperado antes de executar uma curva é fixo e igual para todas as vértebras, exceto para a que se encontra na frente no robô. A ordem de grandeza do intervalo de tempo para a primeira vértebra é trinta vezes menor que o intervalo de tempo para as outras vértebras.

II.5 - Sistema de comunicação

O robô apresenta dois tipos de comunicação: CAN para a comunicação entre as vértebras e ZigBee para a comunicação com o sistema de controle. Esses dois sistemas são visíveis na vértebra da Figura II.10.

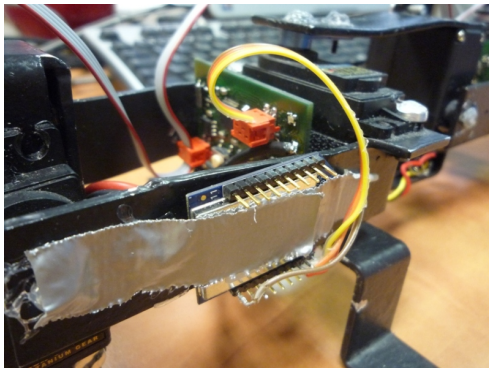


Figura II.10: Sistemas de comunicação na vértebra

II.5.1 - ZigBee

ZigBee é um protocolo de auto nível, com baixo consumo e baixo custo, permitindo a comunicação sem fio baseada na norma IEEE 802.15.4. Primeiramente o baixo custo permite que a tecnologia seja largamente empregada em controle sem fio e monitoramento de aplicações. Em segundo lugar, o baixo consumo de potência permite uma maior vida útil com baterias menores.

A tecnologia do ZigBee tem como objetivo a comunicação em curta distância como a proposta pela tecnologia Bluetooth, porém sendo mais barata e mais simples. Outra diferença entre essa tecnologia e as redes Wi-Fi e Bluetooth é desenvolver menor consumo, por um alcance reduzido (cerca de 10 metros) e as comunicações entre duas unidades poderem ser repetidas sucessivamente pelas unidades existentes na rede até atingir o destino final (comunicação multi-ponto).

Devido à característica de baixo consumo, encontramos esse protocolo em aplicações embarcadas onde o consumo é um critério de seleção. Assim, a automação residencial e os numerosos captores e telecomandos que ela implementa apreciam particularmente esse protocolo em plena expansão e cuja configuração da rede se faz automaticamente em função do acréscimo ou da supressão de nós.

Áreas típicas de aplicação incluem [23]:

- casa inteligente: controle avançado de temperatura, controle de iluminação, segurança e proteção doméstica;
- redes de sensores sem fio.

A Figura II.11 mostra o dispositivo, que implementa o protocolo ZigBee, usado no projeto.

II.5.2 - CAN

O barramento CAN (*Controller Area Network* [25]) é um barramento de tipo *broadcast* com múltiplos mestres. Isso quer dizer que todos os nós escutam todas as mensagens e

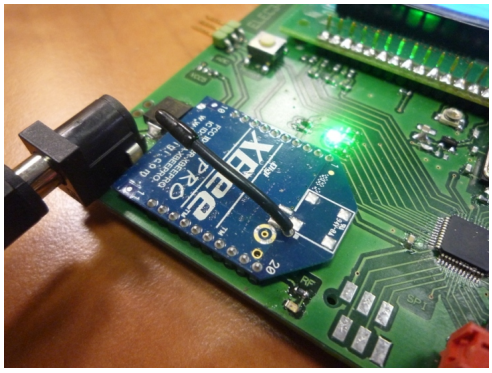


Figura II.11: ZigBee conectado na placa de controle

que não há um componente que controla o acesso à rede. A dimensão das redes pode variar de uma rede pessoal (PAN - *Personal Area Network*) à uma rede local (LAN - *Local Area Network*) dependendo do propósito da rede. Pelo fato das CANs basearem-se na aplicação de sistemas em tempo real é necessário um controle rígido de erros e garantia de recebimento de mensagens.

Cada nó é capaz de enviar e receber mensagens, mas não simultaneamente. A mensagem é composta primeiramente de uma ID, que representa a prioridade da mensagem, e então até oito bytes de dados. Ela é transmitida serialmente para o barramento. O padrão do sinal é codificado em NRZ (Não retorno à zero - *Non-return-to-zero* [24]).

CAN são bastante utilizados em automóveis e indústrias. Os dispositivos que são conectados por uma rede CAN são geralmente sensores, atuadores e outros dispositivos de controle. Esses dispositivos não são conectados diretamente ao barramento, mas através de um controlador CAN e de um processador host.

Se o barramento está livre, qualquer nó pode começar a transmitir. Se dois ou mais nós começam a enviar mensagens ao mesmo tempo, a mensagem com o ID mais dominante (mais bits zero) vai prevalecer e é recebida por todos os nós.

Como não existe endereçamento explicito em mensagens CAN, cada controlador vai pegar todo o trafego no barramento e, com uma combinação de filtros em hardware e software, ele determina se a mensagem é interessante ou não. Na verdade não existe a noção de endereçamento no CAN, em vez disso o conteúdo da mensagem possui um identificador presente em alguma parte da mensagem. Mensagens CAN são ditas como endereçadas por conteúdo (*contents-addressed*).

No momento da configuração do CAN, uma máscara é definida para que o controlador saiba em que parte da mensagem está o identificador. Também no momento da configuração, é dito com que número ele deve comparar a mensagem após aplicada a máscara para saber se a mensagem é relevante ou não. É possível definir vários pares máscara/número, para um mesmo controlador.

A velocidade de transmissão de um barramento CAN varia de acordo com o comprimento do barramento. Essa limitação é devida ao fato de o regime de arbitragem exigir

que a frente de onda do sinal deve ser propagada até o nó mais distante e voltar antes que o bit seja amostrado. Algumas velocidades máximas de transmissão por comprimentos de barramento [26] são mostrados na Tabela II.1.

Comprimento de barramento	Velocidade máxima de transmissão
100 m	500 kbit/s
200 m	250 kbit/s
500 m	125 kbit/s
6 km	10 kbit/s

Tabela II.1: Velocidade máxima de transmissão por comprimento de barramento

Um barramento CAN ISO 11898 precisa ser terminado. Para isso uma resistência de 120Ω é colocado em cada extremidade do cabo. Essa terminação serve para dois propósitos: remover reflexões na extremidade do barramento e assegurar um nível DC correto. Considerando as resistências nas extremidades do barramento, o CAN tem como topologia a mostrada na Figura II.12.

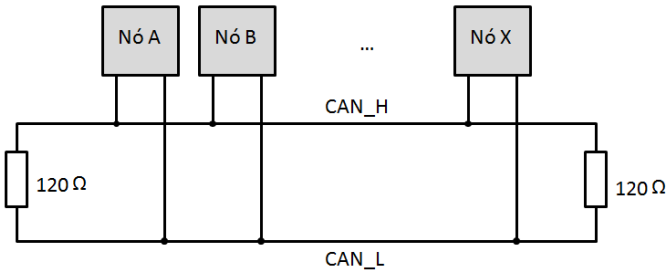


Figura II.12: Topologia do barramento CAN

O CAN apresenta 4 tipos de mensagens (frames) [27]:

- data frame (frame de dados): "aí está a mensagem N e seu conteúdo";
- request frame (frame de pedido): "eu gostaria que alguém me enviasse a mensagem N";
- error frame (frame de erro): "houve um erro";
- overload frame: "não consigo acompanhar o ritmo".

Figura II.13: Formato de uma mensagem de dados CAN

A configuração do frame de dados é mostrado na Figura II.13. SOF e EOF são os bits de inicio (*start of frame*) e final (*end of frame*) do frame. Control tem o número de bytes

de dado presentes em Data. Data pode carregar de 0 a 8 bytes. CRC (*cyclic redundancy check*) são 15 bits usados no código de detecção de erros. IFS (*intra frame space*) é o espaço entre dois frames, composto de pelo menos três bits a nível lócico 1. ID podem ser 11 ou 29 bits, nesse trabalho são usados 29 bits, separados da seguinte forma:

- destinatário: 8 bits com a identificação da vértebra para quem é a mensagem (igual a 0 se *broadcast*);
- remetente: 8 bits com a identificação da vértebra que emitiu a mensagem;
- comando: 5 bits usados para identificar o tipo de ordem enviada à vértebra, ex. ir para frente, mudar a velocidade, etc. ;
- parâmetro: 8 bits usado para os informar os parâmetros necessários às funções implementadas para executar os comandos determinados. Ex. o comando ADC é usado para pedir uma medida e para enviar a resposta ao pedido, o parâmetro é usado para definir se a mensagem é um pedido ou uma resposta.

CAPÍTULO III

SISTEMA DE CONTROLE

Figura III.1: Sistema de controle

O sistema de controle do robô, Figura III.1, é composto de uma interface gráfica rodando em um computador e uma placa de controle, conectada ao computador por um cabo USB. O operador escolhe o comando a enviar para o robô pela interface gráfica. Essa informação é enviada pela USB para a placa de controle que em seguida envia essa informação para o robô via ZigBee, Figura III.2.

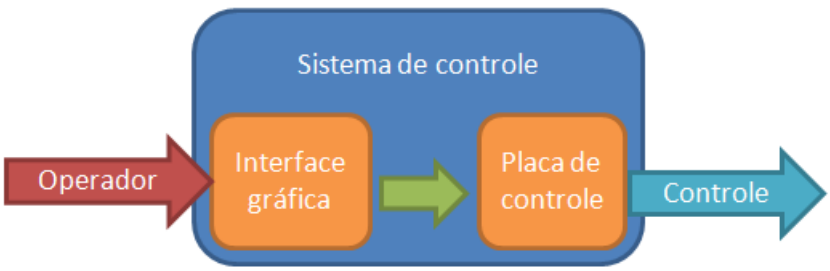


Figura III.2: Funcionamento do sistema de controle

Inicialmente a interface de controle era feita totalmente pela placa de controle, o operador escolhia o comando desejado por meio de dois botões e do lcd, e este comando era enviado para o robô. Essa interface era limitada, e com o desenvolvimento da interface via computador, foi abandonada.

III.1 - Arquitetura

A arquitetura da placa de controle foi desenvolvida para servir para aplicações diversas. Essa placa foi concebida não somente para aplicação no projeto como para ser usada como material de estudo dos componentes presentes. Os componentes incluídos na placa são:

- uma interface LCD;
- uma porta CAN;
- uma porta SPI;
- uma porta USB escrava;
- uma entrada analógica conectada a um potenciômetro;
- dois botões;
- uma porta série (RS232) para debug;
- um módulo ZigBee;
- um buzzer;
- leds;
- uma porta JTAG.

Neste projeto foram usados: o LCD e os botões (primeira versão da interface do operador); o CAN (para testes do bom uso do protocolo), a entrada analógica (para teste da configuração do ADC), a porta série (para conexão ao USB do computador), o ZigBee (para envio de comandos para o robô e para testes da configuração e do uso do protocolo) e a porta JTAG (usada para gravar uma nova versão do programa na placa).

A configuração final da placa de controle é mostrada na Figura III.3 e seus esquemáticos são mostrados no Apêndice I.2.

Figura III.3: Arquitetura da placa de controle

III.2 - Software

No sistema de controle, o software é dividido em duas partes. Uma das partes é a interface gráfica escrita em Python, e a outra é a aplicação que roda na placa de controle.

Através da interface gráfica o operador escolhe a operação desejada. Essa informação é codificada e transmitida por meio da porta USB e do cabo USB/série para a placa de controle. A placa de controle deve traduzir a mensagem recebida para a linguagem usada na comunicação ZigBee e então enviá-la para o robô.

O software da placa de controle, espera que alguma mensagem seja transmitida pela sua porta série, quando isso acontece, uma interrupção é gerada. Essa interrupção desperta a função que vai "interpretar"a mensagem recebida e traduzi-la para ser enviada via ZigBee. As mensagens recebidas pela porta série, assim como o que deve ser enviado pelo ZigBee, são colocados em filas para evitar que informações sejam perdidas caso momentaneamente o fluxo de informações seja grande.

A aplicação da placa de controle, assim como das placas das vértebras é escrito em linguagem C.

III.2.1 - Python

Python [28] é uma linguagem de programação de alto nível, interpretada, dinâmica e orientada a objetos. Ela tem interface para várias operações e bibliotecas do sistema operacional. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada.

A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Ela prioriza a legibilidade do código e combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros. Entre os módulos e frameworks disponíveis para Python está uma biblioteca estilo MATLAB, assim como banco de dados

e módulos para desenvolvimento de sistemas distribuídos.

Outra vantagem da linguagem é ser compatível com diversos sistemas operacionais como várias versões de Unix e Windows.

Para essa aplicação, Python foi a linguagem escolhida pela facilidade para acessar as portas USB do computador e pela simplicidade ao desenvolver interfaces gráficas.

III.2.2 - Interface

A interface nos permite controlar o robô e escrever na flash de todas as vértebras ao mesmo tempo, ou cada uma delas independentemente. O código é dividido em duas partes, a da escritura na flash das placas e a que controla o robô.

Com respeito a escritura na flash, ela é ligada ao *bootloader* presente nas placas das vértebras do robô. Essa parte nos permite entrar em modo bootload. Na inicialização, as placas esperam dez segundos antes de saltar para o programa principal. Durante esses dez segundos, se uma mensagem do *bootloader* chega, as placas entram em modo *bootloader* e não continuam a execução. Nesse modo, é possível escrever um programa a partir de um endereço que pode ser especificado ou verificar que o programa foi escrito corretamente. Saltar para um endereço especifico na flash também é possível.

Todas as mensagens enviadas por essa parte da interface são frames aceitos pelo barramento CAN. Dessa forma a placa de controle se ocupa de receber os bytes e de os organizar no frame antes de enviá-los. Para execução dessa parte da interface, é necessário que o robô esteja ligado à placa de controle, para isso o barramento CAN presente no robô e responsável por conectar as vértebras umas as outras possui onze conectores em vez de dez.

A segunda parte da interface é a que nos permite controlar o robô. Ela não envia diretamente mensagens aceitas pelo barramento CAN, mas apenas mensagens que serão interpretadas pela vértebra mestre do robô. Dessa forma, a placa de controle se ocupa em receber os bits pela porta série e os enviar por ZigBee. Como o ZigBee é conectado a uma UART [29], e esta possui uma fila FIFO, todos os bits recebidos pela porta série são colocados na fila de saída da UART, e consequentemente do ZigBee.

Para facilitar a utilização, foram colocados na interface botões, um para cada ação executada pelo robô:

- avançar, recuar e parar;
- reset;
- virar à direita, à esquerda ou voltar para uma movimento retilíneo;
- aumentar/diminuir a velocidade;
- aumentar/diminuir a amplitude da senóide;

- aumentar/diminuir o período da senóide.

Digitando um valor, o ângulo de uma curva também pode ser modificado.

Para fazer os testes, era possível enviar um frame CAN totalmente configurável. Mas essa opção foi retirada da interface na versão final para que ela tivesse uma aparência mais limpa. A versão final da interface é mostrada na Figura III.4.

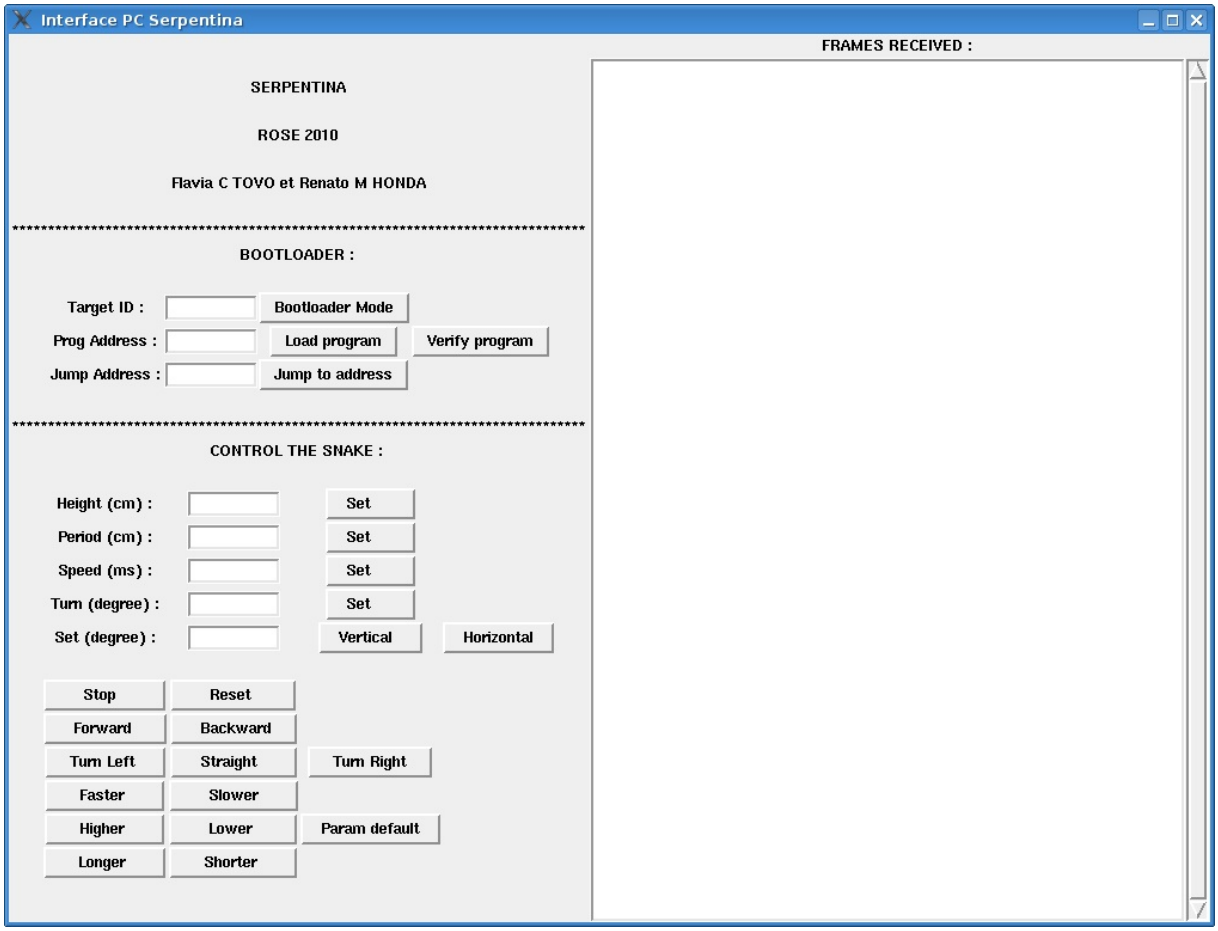


Figura III.4: Interface gráfica

III.2.3 - Comandos

A lista completa de comandos do robô e suas funções são mostrados na tabela III.1.

III.3 - Protocolo de comunicação

O protocolo de comunicação usado nas mensagens enviadas através da interface tem como objetivo ter mensagens curtas mas ao mesmo tempo ter mensagens identificáveis por um observador.

Possuir mensagens curtas tem como função reduzir a quantidade de informação a ser enviada por ZigBee, evitando quaisquer problemas com a fila e diminuindo o tempo de resposta do robô.

Comando	Função
Set Height (cm)	modifica a amplitude da senóide na qual o robô é mapeado, o parâmetro presente nas vértebras será substituído pelo valor colocado no campo
Set Period (cm)	modifica o período da senóide na qual o robô é mapeado, funciona da mesma forma que *Set Height*
Set Speed (ms)	modifica o valor do tempo de espera entre dois cálculos de posição, funciona da mesma forma que *Set Height*
Set Turn (degree)	modifica o ângulo de uma curva, funciona da mesma forma que *Set Height*
Set (degree) Vertical	envia à vértebra identificada no campo *Target ID* da parte do *bootloader* a ordem de modificar sua posição vertical pelo ângulo colocado no campo
Set (degree) Horizontal	funciona da mesma forma que *Set (degree) Vertical* para a posição horizontal
Stop	interrompe o movimento do robô. Com esse comando, o robô permanece na posição que estava anteriormente.
Reset	não só interrompe o movimento do robô como força a posição inicial para os motores.
Forward	determina que o robô deve se movimentar de trás para frente, seguindo convenção da Seção II.4, não apresenta efeito caso o robô esteja no modo *Backward*. Para mudança de sentindo é necessário que o robô seja parado.
Backward	determina que o robô deve se movimentar da frente para trás, semelhante ao *Forward* não apresenta efeito caso o robô esteja neste modo
Turn Left	inicia uma curva para a esquerda. A curva é continuada indefinitamente, até que *Straight* seja acionado.
Straight	volta para um movimento retilíneo, propaga o ângulo horizontal nulo
Turn Right	inicia uma curva para a direita. Funciona da mesma forma que *Turn Left*.
Faster	diminui de um passo fixo o tempo entre dois cálculos de posição
Slower	aumenta de um passo fixo o tempo entre dois cálculos de posição
Higher	aumenta de um passo fixo a amplitude da senóide na qual o robô é mapeado
Lower	diminui de um passo fixo a amplitude da senóide na qual o robô é mapeado
Param default	retorna todos os valores modificados para os valores por default
Longer	aumenta de um passo fixo o período da senóide na qual o robô é mapeado
Shorter	diminui de um passo fixo o período da senóide na qual o robô é mapeado

Tabela III.1: Lista completa de comandos do robô

Mensagens cujo comando é identificável facilita o desenvolvimento do código de tratamento das mensagens ZigBee do lado receptor.

Não há diferença de prioridade entre as mensagens, elas devem então ser tratadas em ordem de chegada.

Outra definição do protocolo é: se não está definido para quem a mensagem deve ser enviada e a mensagem não é relacionada a curvas, logo ela é *broadcast*. Se a mensagem é relacionada a curva, o destinatário também não precisa estar listado, pois ele será a vértebra na posição de cabeça da serpente.

Alguns exemplos dessas mensagens são: HI + (comando *Higher*); HI - (comando *Lower*); SP - (comando *Slower*); FOR (comando *Forward*); LEFT (comando *Turn Left*); STOP (comando *Stop*).

CAPÍTULO IV

Conclusão

O desenvolvimento desse projeto nos permitiu um aprimoramento de diversas competências, como o projeto de circuitos impressos e a programação para microprocessadores. Mais do que o aprimoramento de competências isoladas, a oportunidade de associar diversas áreas em um mesmo projeto também proporciona um grande crescimento como engenheiro, pois em situações reais frequentemente somos confrontados com problemas que englobam mais de uma área de conhecimento.

Durante esse projeto, nós desenvolvemos um sistema de controle e sua interface computacional. A interface computacional foi desenvolvida em Python e a aplicação embarcada na placa de controle foi escrita em linguagem C. Nós projetamos e fabricamos os circuitos impressos presentes nas vértebras do robô. Nós desenvolvemos um sistema distribuído para o controle da trajetória do robô.

Sugestões para trabalhos futuros são apresentadas nos parágrafos abaixo.

Visando a melhoria do projeto, nós poderíamos desenvolver a auto identificação de vizinhança, uma idéia para isso é na inicialização mover um motor vertical de cada vez e medir, pelo ADC, o consumo de corrente nas outras vértebras. Quanto menor a distância entre as vértebras, maior o consumo de corrente.

Outra melhoria seria o uso do ADC para a calibragem automática do robô, isso quer dizer achar o ângulo zero do motor. Isso pode ser feito alterando levemente o ângulo em torno do zero ideal em busca do menor consumo de corrente.

Para tornar o sistema mais robusto a falhas e realmente modular, nós poderíamos conectar um módulo que implementa o protocolo ZigBee em cada uma das vértebras.

Para aumentar a versatilidade do robô, outros tipos de movimentos naturais de cobras poderiam ser estudados e implementados. Isso faria com que o robô se adaptasse melhor a ambientes diversos.

APÊNDICE A

Esquemáticos

I.1 - Vértebra

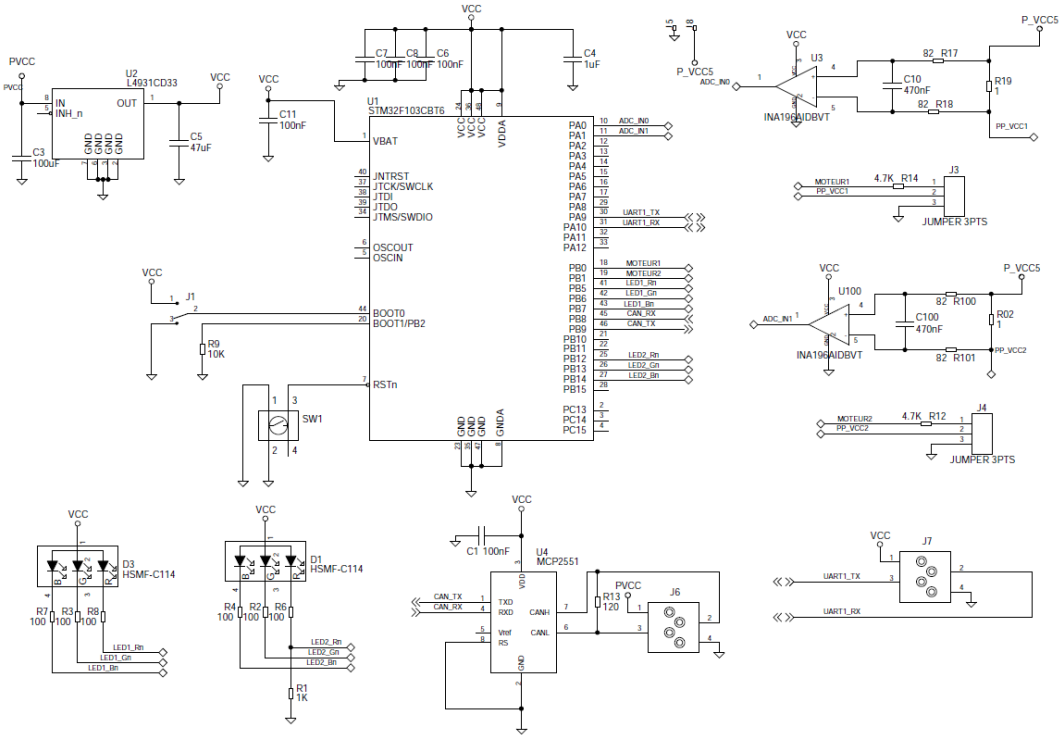


Figura A.1: Esquemático da placa usada em cada vértebra

I.2 - Placa de comando

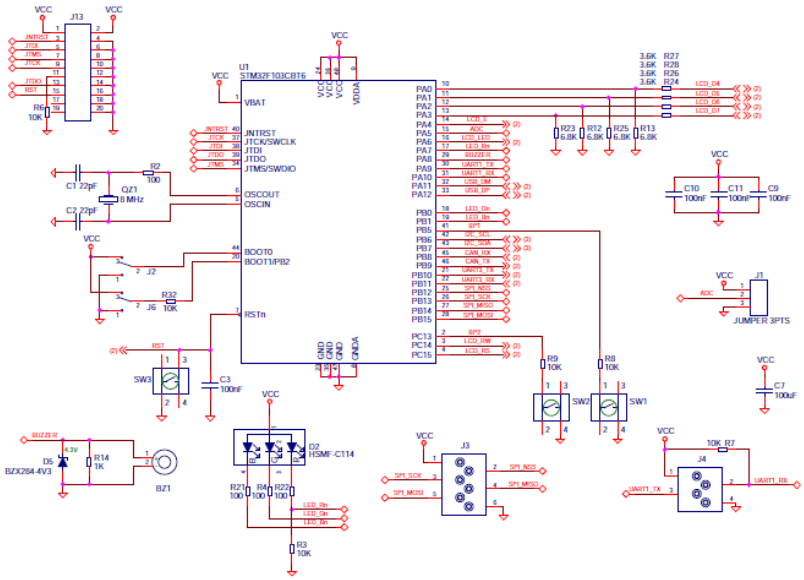


Figura A.2: Esquemático da placa de comando - parte 1

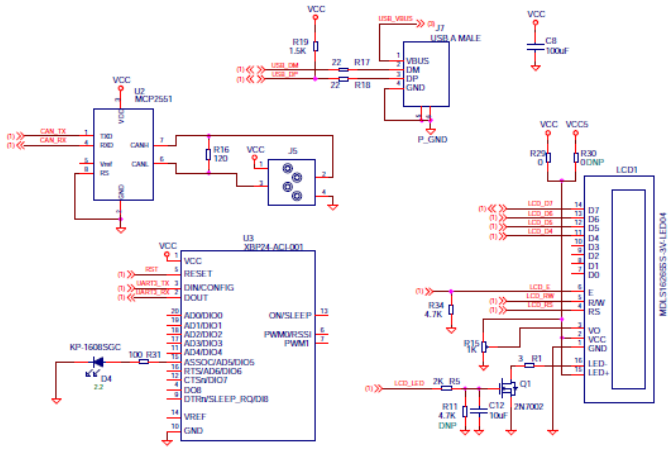


Figura A.3: Esquemático da placa de comando - parte 2

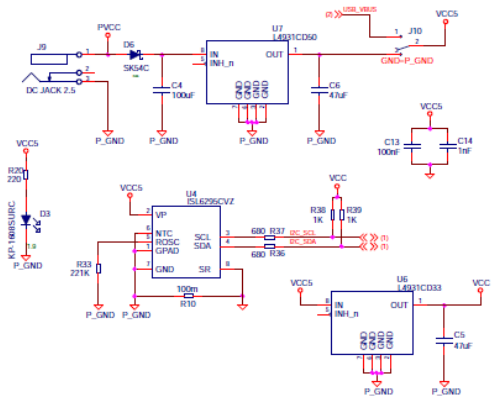


Figura A.4: Esquemático da placa de comando - parte 3

APÊNDICE B

Servomotor

Os servomotores utilizados no projeto são do tipo HSR-5995TG. As informações presentes nesse apêndice são baseadas nos documentos [15] e [30].

II.1 - Controle do motor

O servomotor, possui um circuito impresso com um processador. O controle do motor, como mostrado na Figura B.1, é feito através de MOSFETS configurados em ponte H. As pontes são controladas pelos transistores conectados juntos. A consequência é que a ponte não pode ser completamente desligada. Então ou o motor está acionado ou o motor está travado enquanto o servomotor está alimentado. Nenhum monitoramento de temperatura ou da corrente é feito no motor.

II.2 - Interface de controle

Existem dois modos de controle do motor: modo PWM (ou modo Standard) e modo série. Quando controlado em modo PWM, o motor aceita um sinal de controle variando de 0 a 4,8 Volts (ou seja, a saída do STM32 em 3,3 V se encaixa perfeitamente). Em modo série, uma resistência de pull-down é necessária.

O modo PWM é mais simples, a largura do pulso do sinal de entrada indica o ângulo do motor. Como o servomotor é digital, um único pulso é suficiente para mover o motor.

O modo série permite um retorno de sinal. Dessa forma, pode-se fazer leitura do ângulo atual do motor e também do consumo de corrente. Entretanto um protocolo de comunicação mais complexo é necessário.

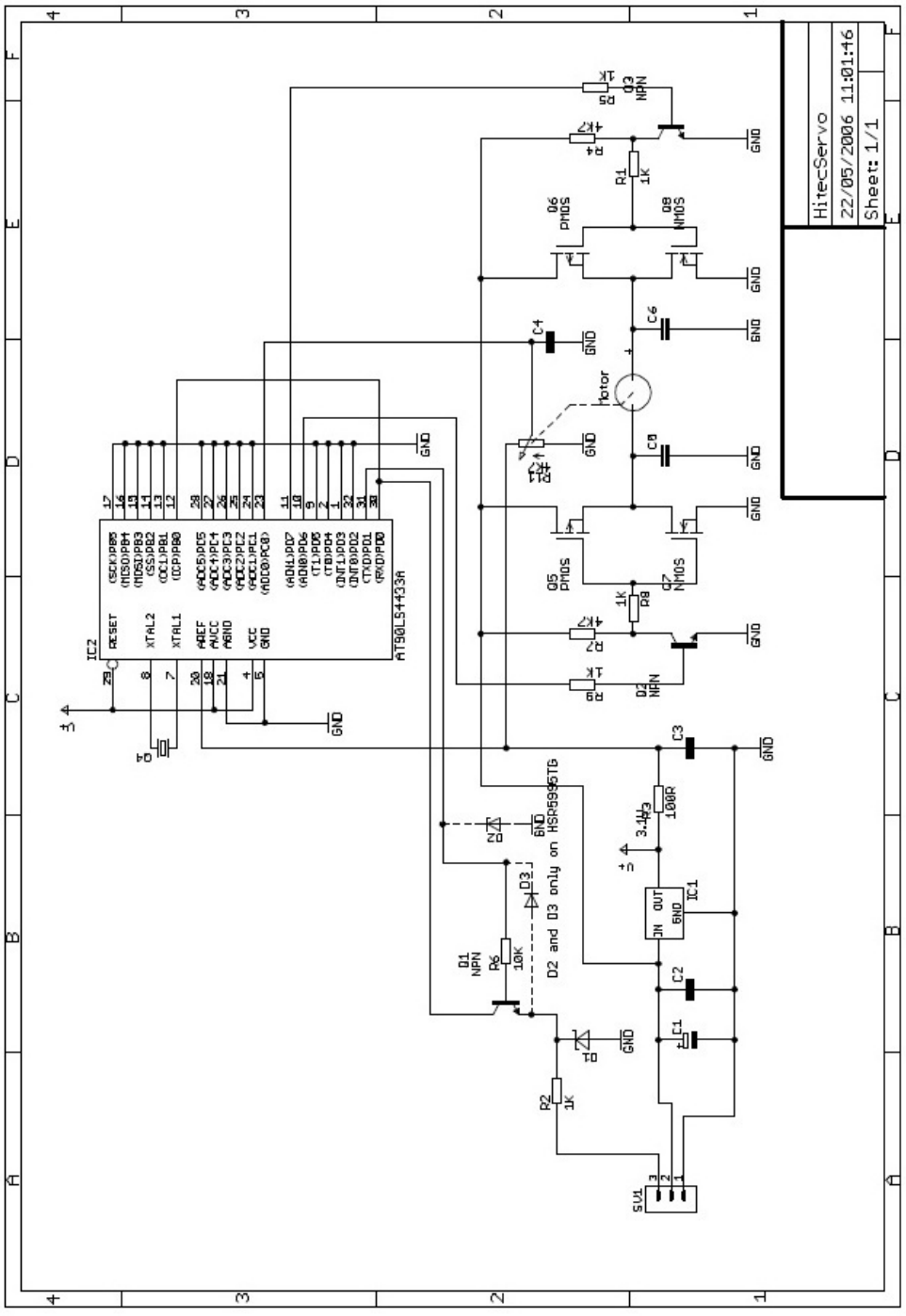


Figura B.1: Esquemático do controle interno do servomotor.

II.2.1 - Servomotor no modo Standard

Para mover o motor, basta apenas enviar o sinal adequado em PWM. Para obter esse sinal, o PWM deve ser configurado para um período de 20ms. Como mostrado na Figura B.2, o ponto neutro é obtido quando o tempo em valor alto da saída vale $1500\mu s$ (duty cycle de 7,5%). Para valores variando de $1500\mu ms$ até $2450\mu s$, um ângulo positivo é obtido. Para valores variando de $550\mu s$ até $1500\mu s$, um ângulo negativo.

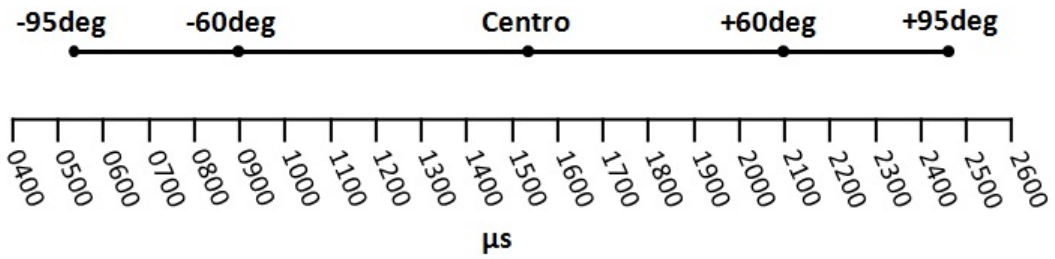


Figura B.2: Ângulo de giro do motor para parâmetro do PWM. FONTE [30]

A partir de um ângulo desejado, o valor em alto do período é obtido pela formula:

$$x = \alpha * 10 + 1500,$$

onde α é o ângulo desejado.

Referências Bibliográficas

[1] "http://en.wikipedia.org/wiki/Robotics".

[2] "http://en.wikipedia.org/wiki/Embedded_system".

[3] MCKENNA, J. C., ANHALT, D. J., BRONSON, F. M., *et al.*, "Toroidal skin drive for snake robot locomotion." In: *ICRA*, pp. 1150–1155, 2008.

[4] "http://www.cs.cmu.edu/ biorobotics/projects/modsnake/".

[5] "http://en.wikipedia.org/wiki/Snakebot".

[6] ERKMEN, I., ERKMEN, A. M., MATSUNO, F., *et al.*, "Snake robots to the rescue", *IEEE Robotics and Automation Magazine*, v. 9, n. 3, pp. 17–25, Set. 2002.

[7] OTA, T., DEGANI, A., SCHWARTZMAN, D., *et al.*, "A Highly Articulated Robotic Surgical System for Minimally Invasive Surgery", *The Annals of Thoracic Surgery*, v. 87, pp. 1253–1256, April 2009.

[8] "http://pt.wikipedia.org/wiki/Servomotor".

[9] *STM32F103xB Datasheet*. STMicroelectronics.

[10] *Application Note. STM32F10xxx hardware development: getting started*. STMicroelectronics.

[11] *Application Note. STM32F101xx, STM32F102xx and STM32F103xx system memory boot mode*. STMicroelectronics.

[12] *Reference manual. STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs*. STMicroelectronics.

[13] *The Insider's Guide To The STM32 ARM Based Microcontroller. An Engineer's Introduction To The STM32 Series*. Hitex Development Tools.

[14] *MCP2551 Datasheet. High-Speed CAN Transceiver*. Microship.

[15] *Hitec Digital Servos Operation and Interface*.

[16] “http://www.freertos.org/”.

[17] “http://en.wikipedia.org/wiki/Pulse-width_modulation”.

[18] “http://animals.howstuffworks.com/snakes/snake3.htm”.

[19] “http://en.wikipedia.org/wiki/Bisection_method”.

[20] “http://en.wikipedia.org/wiki/Brent%27s_method”.

[21] “http://en.wikipedia.org/wiki/Secant_method”.

[22] “http://en.wikipedia.org/wiki/Inverse_quadratic_interpolation”.

[23] *What's so good about ZigBee networks?* Daintree Networks.

[24] “http://en.wikipedia.org/wiki/Non-return-to-zero”.

[25] “http://en.wikipedia.org/wiki/Controller_area_network”.

[26] “http://www.kvaser.com/about-can/the-can-protocol/”.

[27] POLTI, A., TARDIEU, S., *ELECINF344: Bus*. TELECOM ParisTech, 2010.

[28] “http://www.python.org”.

[29] “http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter”.

[30] *Hitec HSR-8498HB Digital Servo Operation and Interface.*