Universidade de Brasília - UnB
Faculdade de Tecnologia - FT
Departamento de Engenharia Elétrica - EnE

PROJETO E IMPLEMENTAÇÃO DE UM SISTEMA DE CONTROLE PARA BRAÇO ROBÓTICO BASEADO EM CINEMÁTICA PARALELA

Luzia Lopes Almeida

Supervisor Acadêmico: Prof. José Camargo da Costa
Universidade de Brasília - UnB

Supervisor técnico: Prof. Anders Stengaard Sørensen
University of Southern Denmark - SDU

Julho 2007

Resumo

Robôs paralelos, quando comparados com manipuladores seriais clássicos, apresentam, em geral, melhor desempenho em termos de precisão, velocidade, rigidez e habilidade para manipular cargas elevadas. Entretanto, a complexidade em ambos os aspectos mecânico e estrutural origina modelos cinemáticos complicados ou específicos, de forma que não são atraentes para a maior parte das aplicações industriais.

Neste projeto, um sistema completo de controle de posição de um braço robótico paralelo é projetado, desenvolvido e implementado. Este sistema permite um posicionamento preciso do braço através do movimento simultâneo de todos os atuadores.

Ademais, um método geral para o estudo da cinmática, ambas direta e inversa, de estruturas paralelas, é aplicado ao manipulador em estudo. Além disso, testes serão executados para determinar algumas caracteríticas do robô.

Os resultados obtidos e dificuldades enfrentadas serão apresentados.

Abstract

Parallel robots, in comparison with classic serial manipulators, usually achieve a better performance in terms of accuracy, velocity, rigidity and ability to manipulate large loads. However, its complexity both in the mechanical and structural aspects leads to particular and complicated kinematics models, making them unattractive for most industrial applications.

In this thesis, a complete position control system of a parallel based robotic arm is designed, developed and implemented. This system permits a precise positioning of the arm through a simultaneous movement of all actuators.

Also, a general method to study both the forward and inverse kinematics of parallel structures is applied to the manipulator used in this project. In addition, a few tests are executed to determine some of the robot's characterics.

The obtained results and difficulties are presented.

Agradecimentos

Ao supervisor técnico, Prof. Anders Sørensen por sugerir este projeto e a todo momento, contribuir com idéias, tornando possível alcançar todos os objetivos.

Ao supervisor acadêmico, Prof. José Camargo da Costa, por concordar com o projeto à distância e tomar todas as medidas possíveis e necessárias para a realização do mesmo.

Aos professores da Faculdade de Engenharia da Universidade do Sul da Dinamarca, especialmente Prof. Henrik Gordon Petersen, por sua paciência em ensinar o método de análise da cinemática e pela ajuda em encontrar erros nos códigos fonte, e Prof. Jørgen Jeppe Madsen, por se colocar sempre à disposição quando era necessária ajuda com a programação do FPGA.

A Arne e Lars, funcionários da Sala de Componentes Eletrônicos da Universidade do Sul da Dinamarca, por atender a todos os pedidos e requisitos, da melhor maneira possível.

A todo o pessoal técnico do Laboratório de Mecânica e do Laboratório de Robótica da Universidade do Sul da Dinamarca, pela ajuda com os equipamentos e ferramentas necessários ao projeto.

Um agradecimento especial a todos os colegas no Laboratório de Robótica da Universidade do Sul da Dinamarca, principalmente Anders Bøgild e aos pesquisadores do grupo "Universal Robots".

À amiga Lívia Batista Maciel, por toda ajuda prestada como minha representante no Brasil.

No entanto, o agradecimento mais importante é para os meus pais, por ter feito este esforço para me manter na Dinamarca e por acreditar que esta oportunidade contribuiria para meu crescimento profissional e pessoal.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Robôs paralelos

Robôs paralelos são mecanismos de malha-fechada que apresentam desempenho superior em termos de precisão, velocidade, rigidez e habilidade para manipular cargas elevadas, quando comparados com robôs seriais comuns. Uma estrutura paralela pode ser definida como contendo duas extremidades, uma base fixa e uma móvel ("base" e "end-effector"), conectados por múltiplas cadeias seriais, em que nem todas as juntas são atuadas[1].

A plataforma de Stewart é reconhecida como um dos primeiros robôs paralelos desenvolvidos e foi publicada pela primeira vez num artigo científico por V. E. Gough em 1956 [2]. É um tipo de manipulador paralelo com seis graus de liberdade. Consiste de seis pernas independentemente atuadas, cada uma contendo uma junta deslizante, e movimentos nas pernas resultam em mudanças na posição e orientação da extremidade móvel [2] (Figura 1.1). Como ocorre com a maior parte dos robôs paralelos, o problema da cinemática direta é extremamente complexo, apesar de, neste caso, o problema da cinemática inversa ter uma solução única e muito simples.

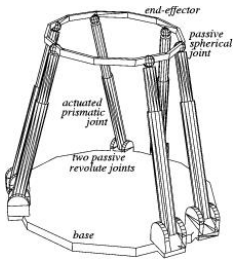


Figura 1.1: Plataforma de Stewart (Fonte:[1]).

Uma configuração muito comum para a plataforma de Stewart é o formato triangular de oito faces, ou octogonal. Esta também é conhecida por plataforma de Sterwart tipo

3-3, pois contém três conexões coincidentes, tanto na base como na extremidade.[1] A Figura 1.2 ilustra esta estrutura. Este tipo de manipulador é bastante eficaz devido à sua rigidez (proporcionada pela distribuição de peso por toda a estrutura) e precisão no posicionamento [3]. Outra vantagem é a possibilidade de mover os seis eixos simultaneamente, em vez dos movimentos seqüenciados exigido pelas máquinas tradicionais. Sendo assim, com uma plataforma Sterwart 3-3 é possível obter seis vezes mais rigidez e cinco vezes mais precisão de movimentos em relação a uma máquina tradicional [3].

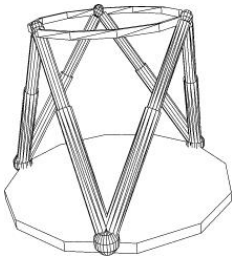


Figura 1.2: Plataforma de Stewart tipo 3-3 (Fonte:[1]).

O fato da carga na extremidade ser suportada simultaneamente por várias pernas, inerente às estruturas paralelas, apresenta outros benefícios potenciais, além da maior resistência mecânica e elevada rigidez [1] [4]. Também apresenta inércia mais baixa, pois a estrutura não precisa suportar todos os segmentos empilhados como numa arquitetura em série e, como tal, não exige uma base tão resistente, a qual poderia requerer mais espaço no chão da fábrica e dificuldades no transporte [3]. Os robôs paralelos podem, assim, ser mais rápidos e leves que um robô em série comum.

Outra grande vantagem intrínseca às estruturas paralelas é o fato de não acumular erros de posicionamento. Numa arquitetura em série, um valor impreciso em um dos eixos serve como ponto de partida para o cálculo do valor no eixo seguinte, resultando numa multiplicação de todas as imprecisões precedentes [3].

Baixa inércia de movimento também pode ser apontada como um benefício das estruturas paralelas sobre as estruturas série, pois é possível, ao contrário do que ocorre numa configuração de eixos empilhados, posicionar todos os motores na mesma base fixa [1]. Este fator também possibilita a ausência de cabos soltos no interior da estrutura, os quais são de difícil manutenção e têm alta probabilidade de causar problemas.

Podem-se enumerar, ainda, outras vantagens, como melhor repetibilidade, confiabilidade (derivado do fato da estrutura não acumular erros) e melhor comportamento dinâmico [4].

Não obstante, para alcançar uma melhor precisão e rigidez em relação às máquinas em série clássicas, a construção da estrutura torna-se altamente complexa, pois exige

[1]Esta configuração também é conhecida por "hexapod", dentro do contexto das máquinas-ferramenta, porém, esta nomenclatura foi registrada pela empresa Geodetic Technology.

tolerâncias rigorosas para a fabricação das juntas, e a mecânica também se torna muito mais complexa, exigindo mais juntas e conexões do que um robô em série equivalente e a um maior custo de manutenção e implementação. A calibração é também bastante complicada e necessária quando um resultado muito exato é exigido [1].

Outra desvantagem é que a área de trabalho (alcance espacial, "workspace") de um robô paralelo é restringida pela possibilidade de as pernas colidirem umas nas outras e, portanto, em geral são obtidas áreas de trabalho menores [1].

Estruturas Hexapod™ ou plataformas de Stewart têm sido, há muito tempo, usadas em simuladores de vôo, de modo a fornecer movimentos multiaxiais rápidos, e são a melhor opção para aplicações de alinhamento ultra precisas e de posicionamento multiaxial [5]. No entanto, também se podem encontrar aplicações no campo das máquinas-ferramenta, tecnologias de gruas, pesquisas subaquáticas, salvamentos terra-mar, simulações de vôo, posicionamento de satélites, telescópios e cirurgia ortopédica [2]. Não obstante, as desvantagens normalmente ultrapassam as vantagens em aplicações industriais comuns.

1.1.1 Descrição do robô

O robô considerado neste projeto consiste de três plataformas idênticas e simétricas, posicionadas uma sobre a outra. É um protótipo desenvolvido pela NASA para ser utilizado em missões espaciais. Uma foto do robô pode ser vista na Figura 1.3.[2]

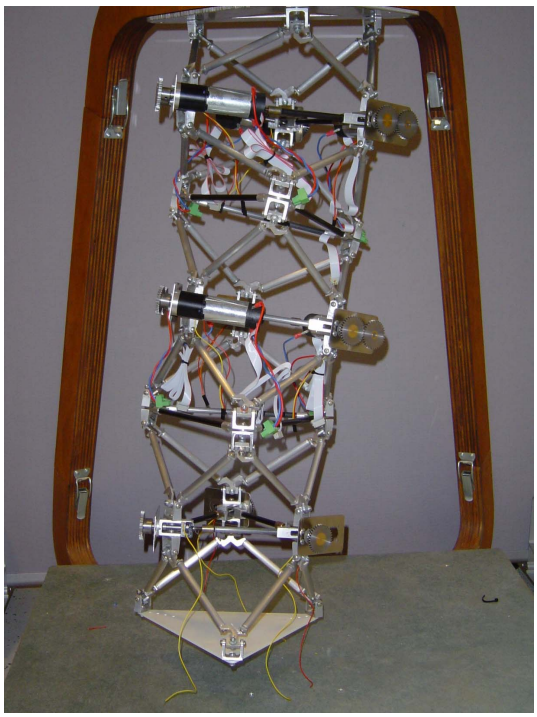
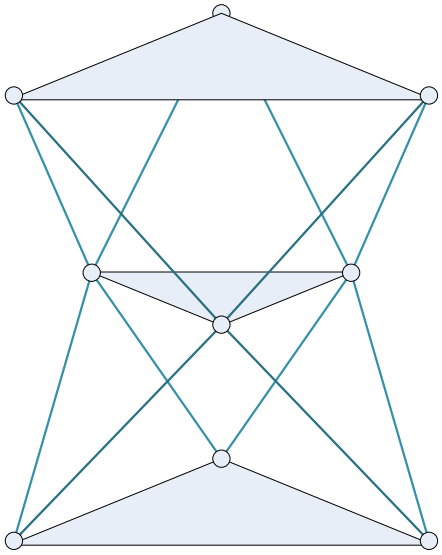
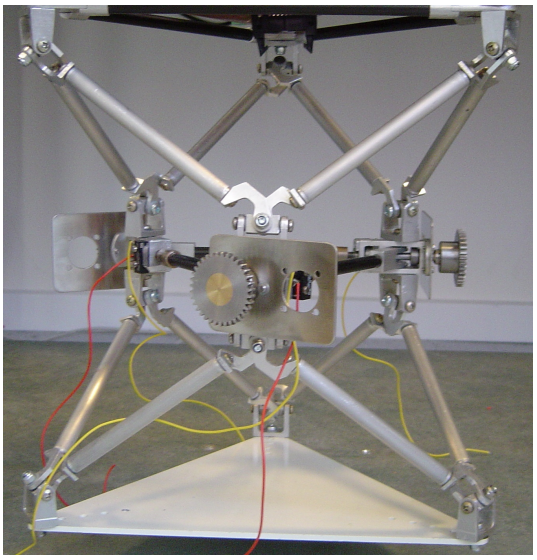


Figura 1.3: Estrutura do robô.

[2]Todas as figuras sem fonte foram criadas pelos autores.



(a) Model de uma plataforma do robô.



(b) Uma plataforma física do robô.

Figura 1.4: Plataforma do robô.

Cada plataforma do robô tem três atuadores lineares de fuso e, portanto, três motores. No ponto mais à esquerda do atuador encontra-se um interruptor, para indicar quando o motor alcançou a posição inicial.

As plataformas poder ser descritas como duas estruturas do tipo plataforma de Stewart 3-3, posicionadas em posições contrárias, uma sobre a outra, de forma que suas extremidades ("end-effector") coincidam. As bases e extremidades de cada plataforma têm formato triangular equilateral e os motores encontram-se no meio da plataforma, ou seja, na base da estrutura Stewart 3-3 superior. Cada uma das seis pernas de cada estrutura Stewart 3-3 contém duas juntas de revolução em cada conexão, permitindo o movimento em torno de dois eixos perpendiculares. A Figura 1.4(a) é o modelo de uma plataforma do robô, enquanto a Figura 1.4(b) é a plataforma real.

1.2 Formulação do problema

Robôs paralelos apresentam melhor desempenho em muitos aspectos, quando comparados com estruturas seriais, como referido na seção 1.1. No entanto, o estudo da sua cinemática é, em geral, altamente complexo, visto que, por norma, cada estrutura exige uma solução bastante específica. Efetuar o controle deste sistema também apresenta dificuldades, dada a necessidade de controlar todos os atuadores simultaneamente e num intervalo de amostragem adequado.

O problema da cinemática de posição nunca foi solucionado para o robô considerado, descrito na subseção 1.1.1, e seu sistema de controle não funciona há alguns anos, apesar dos esforços de diversos estudantes.

Os principais objetivos do projeto descrito neste relatório são: projetar e implementar um sistema de controle para o braço robótico e desenvolver e analisar ambas as cinemáticas de posição direta e inversa para esta estrutura, usando um método geral.

1.3 Estruturação do trabalho

Este relatório está dividido em quatro capítulos principais. No Capítulo 2, os princípios teóricos de todos os elementos do sistema de controle são discutidos e o método utilizado na análise de ambas as cinemáticas direta e inversa do robô é apresentado. Esta técnica foi desenvolvida pelo Professor Henrik Gordon Petersen e publicada em [6]. No Capítulo 3, todas as especificações e considerações feitas durante o desenvolvimento do projeto são apresentadas e discutidas. No Capítulo 4, são mostrados e analisados todos os resultados obtidos nos testes de bancada realizados no robô e no sistema de controle. O relatório é concluído no Capítulo 5, onde é feita uma apreciação geral do trabalho e futuras pesquisas são sugeridas. O Apêndice contém os esquemáticoe e "layouts" de todas as placas de circuito impresso e o esquemático interno da FPGA. Ademais, como parte integrante do Apêndice, um CD é anexado, contendo todos os códigos fontes desenvolvidos, as descrições técnicas ("datasheet") de todos os elementos, arquivos de esquemáticos e "layouts" e uma versão eletrônica deste relatório (em formato PDF).

Capítulo 2

Fundamentos teóricos

Neste capítulo, os fundamentos teóricos nos quais este trabalho baseia-se são apresentados em duas seções principais: *Sistema de controle* e *Cinemática*.

2.1 Sistema de controle

Esta seção descreve, brevemente, a estrutura de um sistema de controle de posição de motores, envolvendo todos os seus elementos, como motor CC e seu modelo matemático, "drivers" de motores, PWM, "encoders" incrementais, controladores de posição, FPGA e comunicações SPI.

2.1.1 Motores CC

O clássico motor de corrente contínua contém um estator, constituído por ímãs permanentes, e um rotor, o qual é uma bobina de fio de cobre com núcleo ferromagnético, alimentado através de um comutador, formando um eletroímã. O comutador funciona como um interruptor rotatório que inverte a direção da corrente elétrica, para gerar um fluxo de corrente através da armadura, de forma que a força aplicada na armadura se mantenha sempre na mesma direção por toda a volta. Durante os instantes de mudança de polaridade, a inércia mantém o motor girando na direção certa [7] (Figure 2.1).

O ímã permanente também pode ser substituído por eletroímãs e esta configuração permite um controle mais preciso da razão velocidade/torque do motor, dado que o campo também pode ser controlado, além da armadura.

Modelo matemático dos motores CC

Um motor CC pode ser eletricamente e fisicamente representados pelo modelo mostrado na Figura 2.2 [9].

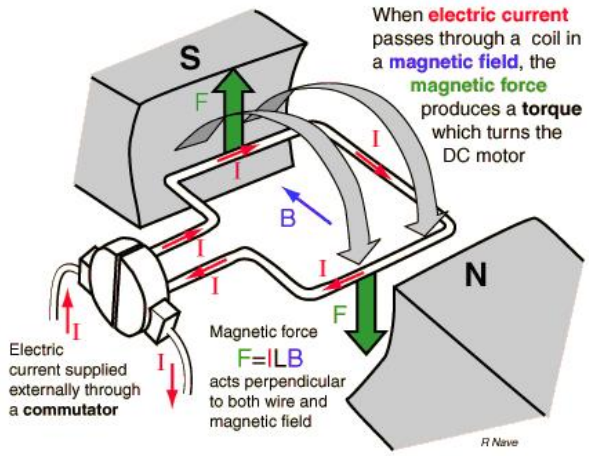


Figura 2.1: Princípio de operação de um motor CC de ímã permanente. (Fonte:[8]).

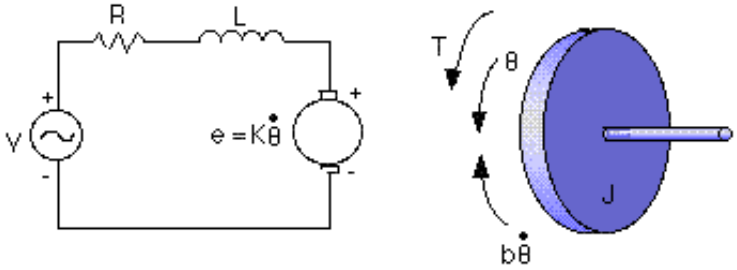


Figura 2.2: Modelo elétrico e mecânico de um motor CC (Fonte: [10]).

O torque de saída de um motor, T_m, depende apenas da corrente i_a fluindo na bobina da armadura, e sua relação pode ser vista em (2.1).

$$T_m = K_t \cdot i_a \tag{2.1}$$

onde $K_t\,[N \cdot m/A]$ é a constante de torque.

Quando o eixo do motor gira, este movimento induz uma tensão nos enrolamentos do motor. Esta tensão induzida é chamada força contra-eletromotriz, V_{emf}, e se opõe à tensão externa aplicada, V_a. A tensão induzida é proporcional à velocidade angular do eixo do motor pela relação (2.2).

$$V_{emf} = K_e \cdot \omega_m \tag{2.2}$$

onde $K_e\,[V \cdot s/rad]$ é a constante de tensão do motor.

As bobinas do motor são modeladas por uma tensão aplicada sobre uma resistência $R\,[\Omega]$ e um indutor $L\,[H]$. Assim, a equação diferencial ordinária (2.3) pode ser encontrada

para o modelo elétrico.

$$V_a - V_{emf} = L\frac{di_a}{dt} + R \cdot i_a \tag{2.3}$$

Para o sistema mecânico, aplicando-se a segunda lei de Newton, $F = m \cdot a$, na sua forma equivalente para movimentos rotacionais, obtêm-se (2.4).

$$T = J \cdot \alpha + b \cdot \omega = J\frac{d\omega}{dt} + b \cdot \omega \tag{2.4}$$

onde $T\,[N \cdot m]$ é o torque necessário para rotacionar um objeto com momento de inércia $J\left[\frac{Nms^2}{rad}\right]$ numa aceleração angular $\alpha\,[rad/s^2]$.

Desenvolvendo (2.1) a (2.4), pode-se mostrar que a função de transferência do motor CC, sem carga, é (2.5).

$$H(s) = \frac{K_t}{L \cdot J_m \cdot s^2 + R \cdot J \cdot s + K_t \cdot K_e} \tag{2.5}$$

2.1.2 Drivers de motores

Para controlar a velocidade e a direção de rotação de um motor CC, representado pela função de tranferência (2.5), faz-se necessário controlar a tensão de entrada aplicada.

Geralmente isto não pode ser feito usando-se, apenas, um controlador típico, devido à corrente elétrica fornecida, a qual não é suficiente para movimentar o motor. Além disso, o motor gera ruído elétrico que pode causar distorções nos sinais de controle quando a direção do motor ou velocidade são modificados. Este é o principal motivo de utilização de circuitos especializados ("motor drivers") desenvolvidos para fornecer potência aos motores e isolá-los dos outros circuitos integrados, minimizando problemas elétricos.

Os drivers de motores mais largamente utilizados são constituídos por um conversor CC-CC (também chamado "chopper"), numa configuração tipo ponte-H ("H-Bridge"), e dispositivos semicondutores de chaveamento. O dispositivo de chaveamento mais comum é o MOSFET de potência, mas também podem ser utilizados trasistores bipolares ou IGBT.

Pontes-H

A configuração mais geral de "chopper" é a ponte-H que pode ser vista na Figura 2.3. Este tipo de configuração pode operar nos quatro quadrantes do plano Tensão X Corrente, que permite que o motor CC seja acionado nas direções direta ou inversa em qualquer velocidade, usando uma fonte de tensão independente. Outras configurações são possíveis, mas são casos específidos desta.

Esta configuração permite que se alimente o motor com tensões desde $+V_{source}$ (Figura 2.4) a $-V_{source}$ (Figura 2.5), dependendo dos transistores acionados.

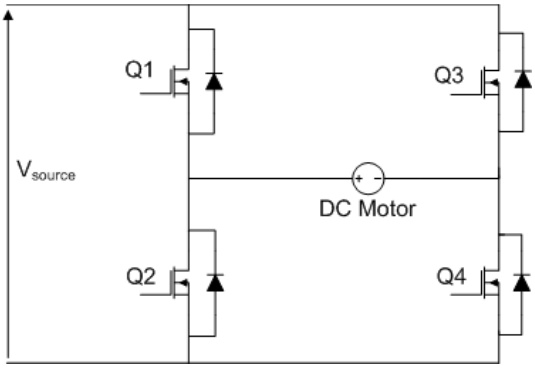


Figura 2.3: Configuração ponte-H.

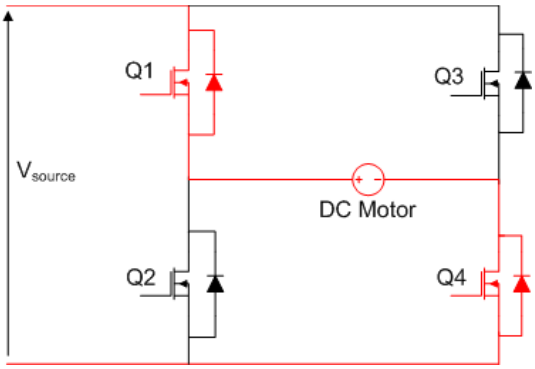


Figura 2.4: Motor CC alimentado com $+V_{source}$.

Para alimentar a carga, deve-se acionar transistores diametricamente opostos. Deve-se observar, ainda, que não se pode acionar todos os transistores ao mesmo tempo, visto que, se dois transistores do mesmo braço forem acionados juntos, ocorre um curto-circuito e uma corrente muito alta atravessa os transistores, podendo danificá-los permanentemente.

Dispositivos de chaveamento

As características mais comuns de diversos dispositivos de chaveamento podem ser observados na Tabela 2.1.

Tipo	**Tensão / Corrente**	**Freqüência máxima (Hz)**	**Tempo de chaveamento (μs)**
Diodos	5000 V / 5000 A	1 k	100
Tiristores	5000 V / 5000 A	1 k	200
Transistores de potência	400 V / 250 A	20 k	9
MOSFET de potência	500 V / 8.6 A	100 k	0.7
IGBT	1200 V / 400 A	20 k	2.3

Tabela 2.1: Características típicas de dispositivos de chaveamento (Fonte: Adaptado de [11]).

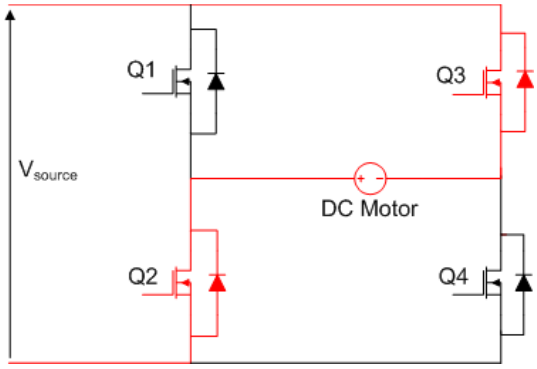


Figura 2.5: Motor CC alimentado com $-V_{source}$.

Para aplicações com motores CC, o dispositivo mais utilizado é o MOSFET de potência, já que este é o dispositivo que melhor se aplica a conversores de baixa potência e altas freqüências, devido à sua alta velocidade de chaveamento e os valores de tensão e corrente que conseguem controlar.

Drivers de ponte-H

Para acionar os transistores de uma ponte-H é necessário polarizá-los da forma correta. Há diversos drivers disponíveis, de acordo com o tipo de dispositivo de chaveamento utilizado. Visto que o dispositivo mais utilizado para este tipo de aplicação é o MOSFET de potência, como mostrado anteriormente, estes serão um pouco mais detalhados a seguir.

A condição para acionar um MOSFET de potência é aplicar um diferença de tensão positiva entre a fonte (S) e o Gate (G) (V_{GS}). O valor de V_{GS} é geralmente em torno de $10V$. Em uma ponte-H esta diferença é mais facilmente gerada para a parte inferior da ponte, já que as fontes destes transistores estão conectadas à terra.

Os transistores da parte superior têm suas fontes conectadas à carga, cuja tensão se modifica antes e depois da comutação dos transistores superiores, o que torna seu controle difícil. A maior dificuldade surge depois do acionamento dos transitores superiores, visto que a tensão nas suas fontes será igual à tensão de alimentação. Assim, para continuar conduzindo, deve-se aplicar em seus gates uma tensão superior à tensão máxima do circuito, devido à necessidade de se manter V_{GS} positiva. Isto torna indispensável o uso de circuitos "bootstrap".

2.1.3 Modulação por largura de pulso (Pulse-Width Modulation – PWM)

Como mostrado previamente, a velocidade de um motor CC é diretamente proporcional à tensão de alimentação, assim, se a tensão fornecida é reduzida de, e.g. $24V$ para $12V$, o motor vai se movimentar com metade da velocidade inicial. Contudo, deve-se

gerar uma tensão de alimentação variável a partir de uma fonte de tensão fixa, e.g. de $24V$.

O controlador de velocidade funciona variando-se a tensão média fornecida ao motor. Uma solução possível é ligar e desligar a tensão de alimentação do motor muito rapidamente. Se este chaveamento é rápido o suficiente, a tensão média no motor é contínua, dada à sua impedância indutiva, e corresponde à tensão média aplicada. A tensão média está relacionada com tempo em que a tensão está ligada comparado com o tempo em que está desligada, i.e. o ciclo de trabalho ou razão cíclica ("duty cycle", D), o qual representa a razão entre o período em que a função está acionada, τ, com o período total da função, T, como em (2.6)[12].

$$D = \frac{\tau}{T} \tag{2.6}$$

A modulação por largura de pulso modula o ciclo de trabalho de uma onda quadrada, relacionando-o a um sinal de entrada, geralmente uma tensão.

Considerando-se a função $f(t)$, o valor médio da sua curva pode ser encontrado a partir de (2.7).

$$\overline{y} = \frac{1}{T}\int_0^T f(t)dt \tag{2.7}$$

Em uma PWM, $f(t)$ é uma onda quadrada com valor mínimo y_{min}, valor máximo y_{max} e ciclo de trabalho D. Assim, seu valor é y_{max} no período $0 < t < D \cdot T$ e y_{min} em $D \cdot T < t < T$. A expressão (2.7) transforma-se, portanto, em (2.8).

$$\overline{y} = D \cdot y_{max} + (1 - D) \cdot y_{min} \tag{2.8}$$

Em muitos casos, $y_{min} = 0$, e então, (2.8) pode ser simplificado para (2.9)

$$\overline{y} = D \cdot y_{max} \tag{2.9}$$

De (2.9), é fácil verificar que o valor médio do sinal depende diretamente do seu ciclo de trabalho. Este princípio é mostrado na Figura 2.6.

Um driver para controlar motores CC, de custo relativamente baixo, utiliza um PWM e transistores de potência. Os transistores são ativados e desativados por um sinal de tensão alternada de saída do PWM.

Deve-se escolher cuidadosamente uma freqüência PWM adequada para aplicações de controle de motores CC. A baixas freqüências, pode ocorrer um fenômeno chamado "condução discontínua de corrente". Isto significa que, durante a descarga da indutância (período em que o PWM está inativo), a corrente se anula, causando o desligamento do motor. Isto pode ser muito prejudicial ao motor, reduzindo sua vida útil. A Figura 2.7

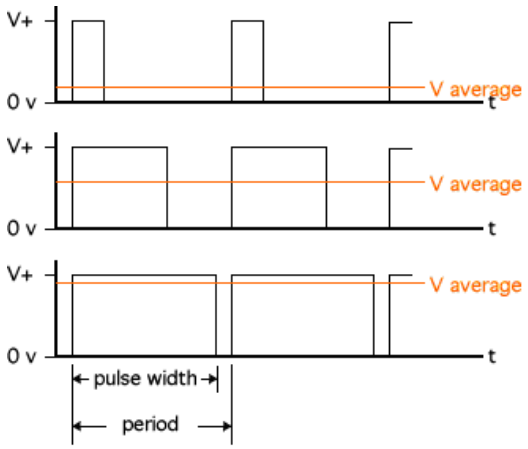


Figura 2.6: Sinal PWM (Fonte:[13]).

mostra o comportamento da corrente no motor, quando um sinal de tensão do PWM é aplicado aos seus terminais.

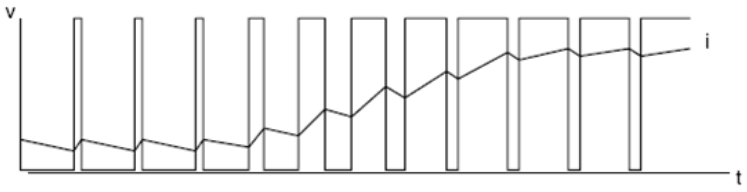


Figura 2.7: Corrente no motor quando lhe é aplicado um sinal PWM.

A altas freqüências, a corrente alterna-se rápido o suficiente de forma a nunca se anular, resultando num modo de condução contínua. Quanto mais alta a freqüência de chaveamento, mais estável é a forma de onda da corrente no motor. No entanto, a interferência RF do circuito aumenta com a freqüência e a dissipação de potência nos dispositivos de chaveamento é maior, devido ao grande número de comutações realizadas.

Há dois tipos de drivers de motores com PWM. O primeiro é chamado "Sign-magnitude", e funciona com 0% de razão cíclica fornecendo potência nula e 100% máxima potência, e um bit em separado definindo a direção. O seundo é chamado "Locked Anti-phase". O motor movimenta-se em ambas as direções, mas está sempre conectado à fonte de potência. A 50% de razão cíclica não é permitido o fluxo de corrente e o motor não se movimenta. Como o motor está sempre conectado à fonte de potência, há sempre uma baixa impedância através de seus terminais. A desvantagem é a ocorrência de ruído indutivo na freqüência de comutação.

2.1.4 Encoder incremental

Para medir a velocidade e a posição do motor, pode-se utilizar um dispositivo chamado "encoder óptico incremental". Este tipo de encoder é constituído por um LED, um foto-

diodo e um disco ranhurado, e tem por função converter uma posição mecânica em sinal elétrico.

O LED emite um raio de luz, o qual é detectado por um fotodiodo na outra ponta. Se um disco, com ranhuras, é posicionado entre o LED e o receptor, o sinal só é captado quando a ranhura se encontra entre o transmissor e o receptor. Um exemplo de disco está mostrado na Figura 2.8.

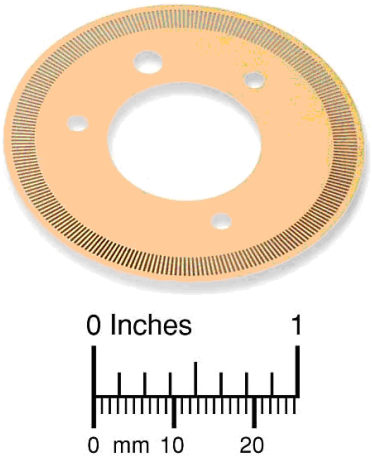


Figura 2.8: Disco do encoder (Source:[14]).

Tipicamente, os encoders têm um disco com uma trilha com várias ranhuras e outra trilha com apenas uma ranhura, que gera apenas um sinal por revolução, e pode ser usado para gerar uma posição absoluta (sinal de índice, I). Para conseguir informação sobre a direção do movimento, são lidos geralmente dois sinais, com uma certa diferença mecânica, gerando dois sinais com uma diferença de fase de 90°entre eles. Estes dois sinais são geralmente chamados A e B (Figura 2.9).

Os encoders geralmente vêm com sinais de saída complementares para os canais A, B e I, os quais podem ser usados, utilizando eletrônica adicional, em ambientes industriais ruidosos, de forma a conseguir maior tolerância a interferências.

A resolução de um encoder é normalmente expressa em número de pulsos por revolução e corresponde ao número de ranhuras do disco dos sinais A e B. Quanto mais ranhuras têm o disco, maior a resolução do encoder.

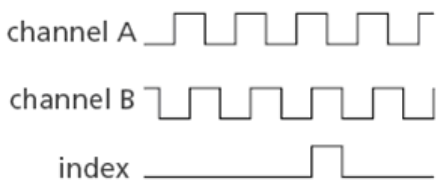


Figura 2.9: Sinais de um encoder incremental.

Para determinar a direção de rotação, é necessário comparar os sinais A e B. Escolhendo o sinal A como referência e comparando-o com o sinal B na borda ascendente, pode-se observar que, se B=1, então o motor gira na direção horária e se B=0, na direção anti-horária.

A maior desvantagem na utilização de encoders incrementais ao invés do linear é o fato de não se conseguir obter uma maneira direta de identificar a posição inicial do motor ou mesmo do sistema. Disto, surge a necessidade de utilizar dispositivos extras que permitam ao sistema alcançar uma posição conhecida.

2.1.5 Controlador de posição

No controle de motores, controladores de malha fechada são largamente utilizados, para tornar o sistema imune à presença de incertezas e pertubações externas (e.g. variações na tensão de entrada e na carga). O objetivo é variar a tensão aplicada ao motor de forma a movimentá-lo com uma velocidade específica ou para uma posição específica. No caso tratado neste projeto, somente a posição será considerada.

Faz-se necessário, então, medir a posição atual do motor e compará-la com a posição desejada. A diferença entre elas é chamada erro e é aplicado ao controlador, o qual modificará a tensão aplicada ao motor. Para obter a posição a partir do encoder, um integrador deve ser colocado na sua saída (Figura 2.10).

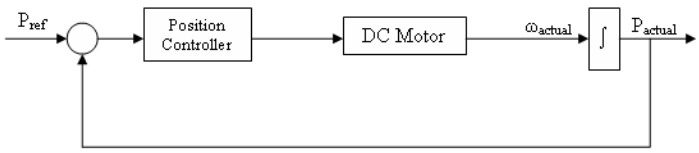


Figura 2.10: Diagrama de blocos de um controlador de posição típico.

Tipicamente, o controlador utilizado é do tipo P, PI ou PID. Um controlador PID é constituído por três efeitos diferentes: Proporcional, Derivativo e Integral (Figura 2.11). Cada uma das partes causam um efeito próprio no comportamento do sistema em malha-fechada [15].

O efeito proporcional é utilizado para lidar com o erro presente, sendo K_p chamado ganho proporcional. Quanto maior K_p, o sistema reagirá ao erro mais rapidamente, entretanto, um valor muito alto de K_p pode, eventualmente, levar o sistema à instabilidade.

O efeito integral é usado como "memória" do sistema. Os valores de erro são integrados no tempo e multiplicados por um ganho designado por K_i. Assim, é feita uma média e adicionada à quantidade a ser controlada. Este efeito integral elimina o erro em estado permanente.

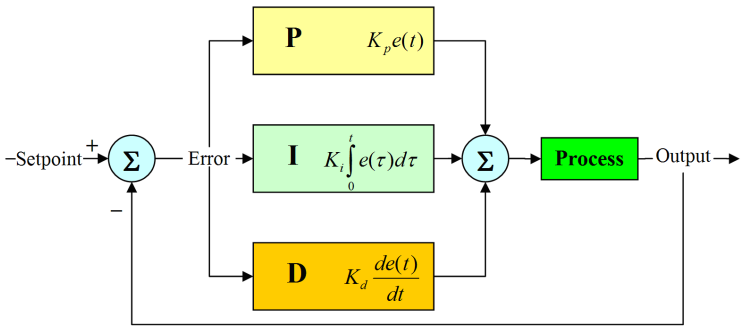


Figura 2.11: Controlador PID (Fonte:[15]).

O efeito derivativo é usado para tentar prever o estado futuro do sistema. A derivada da posição atual do sistema é multiplicada por uma constante K_d designada constante derivativa, e somada à quantidade a ser controlada. O efeito derivativo é utilizado para conseguir uma resposta mais rápida a mudanças no sistema.

A equação resultante é, portanto, (2.10) [16].

$$\begin{aligned} u(t) &= K\left[e(t) + \frac{1}{T_I}\int e(\tau)d\tau + T_D\frac{de(t)}{dt}\right] \\ &= K_p e(t) + K_i\int e(\tau)d\tau + K_d\frac{de(t)}{dt} \end{aligned} \tag{2.10}$$

Esta é a implementação analógica/contínua do controlador PID. No entanto, se um microcontroller estiver sendo utilizado, faz-se necessária a implementação discreta/digital, usando o domínio da Transformada-Z ao invés do domínio da Transformada-S. Duas possibilidades são: projetar o controlador a partir da teoria do controle digital, ou tentar discretizar o controlador analógico [17]. Seguindo esta segunda opção, a discretização pode ser feita para pequenos intervalos de tempo T_s. O termo derivativo, então, pode ser substituído por uma expressão diferencial de primeira ordem e o termo integral, por uma soma. Como o termo integral deve ser a soma de todos os erros passados, o erro precisa ser armazenado a cada intervalo T_s. Assim, o algoritmo do controlador PID discreto, derivado de (2.10), torna-se (2.11) [16].

$$u(k) = K\left[e(k) + \frac{T_s}{T_I}\sum_{i=0}^{k-1} e(i) + \frac{T_D}{T_s}(e(k) - e(k-1))\right] \tag{2.11}$$

2.1.6 "Field Programmable Gate Array" (FPGA)

Um FPGA é um dispositivo semicondutor que contém componentes lógicos programáveis, os quais podem ser configurados para realizar diversas funções, desde portas lógicas básicas como "AND", "OR" e "NOT", até funções combinacionais complexas, como decodifi-

cadores ou funções matemáticas simples. Na maior parte dos FPGA, estes blocos lógicos também incluem elementos de memória, tais como "flip-flops", ou blocos de memória mais complexos.

Este dispositivo pode ser programado, após o processo industrial, pelo projetista, de forma a realizar qualquer função lógica necessária[1].

Para determinar o comportamento da FPGA, deve-se utilizar linguagens de descrição de *hardware* ("Hardware Description Language" – HDL) ou esquemáticos. As linguagens HDL mais comuns são VHDL e Verilog. A partir de então, o *software* proprietário da empresa, que geralmente é adquirido junto com a FPGA, executará todos os procedimentos necessários para converter a linguagem num arquivo de programação conveniente, a ser executado pela FPGA.

Ademais, para verificar e validar o funcionamento dos blocos programados, há *softwares* dedicados de simulação, que permitem ao programador testar seu código fonte e corrigir enventuais problemas.

2.1.7 "Serial Peripheral Interface" (SPI)

A interface serial SPI é um protocolo simples e padrão, que permite a troca de dados entre um mestre ("master") e um ou mais escravos ("slaves") de uma forma simples e eficaz. Este protocolo especifica quatro sinais lógicos:

- *SCK* - Clock da comunicação SPI
- *MOSI* - Master Output / Slave Input: sinal de saída do mestre e entrada do escravo
- *MISO* - Master Input / Slave Output: sinal de entrada do mestre e saída do escravo
- *SS* - Slave Select: seletor de escravo

O mestre é responsável por gerar o sinal de clock, o qual permite ao escravo receber e transmitir dados, além de gerar o sinal *SS*. Se apenas um mestre e um escravo são utilizados, *SS* pode ser conectado diretamente à terra, no dispositivo escravo.

O protocolo SPI contempla quatro modos de operação, determinados pelo mestre, de acordo com o nível do sinal de clock quando o mestre está inativo (CPOL[2]) e a borda em que são feitas a amostragem ("sampling") e a transmissão ("setup") (CPHA[3]). A Figura 2.12 mostra os modos de operação com $CPHA = 0$.

Todos os quatro modos de operação são mostrados na Tabela 2.2.

[1] Não está no escopo deste trabalho detalhar a arquitetura e funcionamento dos FPGA. Informações mais detalhadas podem ser encontradas em [18].

[2] $CPOL = 0$ corresponde a um valor base de clock igual a 0.

[3] $CPHA = 0$ corresponde à amostragem dos dados na primeira ("leading") borda e transmissão de dados segunda borda ("trailing").

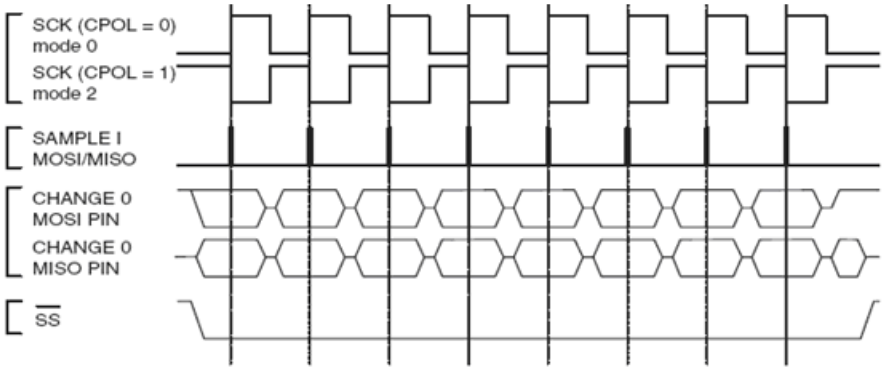


Figura 2.12: Formato de transferência SPI com CPHA=0 (Fonte: [19]).

Modo	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tabela 2.2: Modos de operação SPI.

As principais vantagens da interface SPI é o fato de ser uma comunicação "full-duplex", permitindo um alto "throughput", com modos de operação flexíveis e inexistência de endereçamento, com um reduzido "overhead" de protocolo, o que pode ser vantajoso no caso de poucos escravos.

A maior desvantagem é que os escravos só podem transmitir quando o mestre o faz, logo, não há controle de fluxo e conhecimento de escravo – o mestre pode estar enviando informação que não está sendo recebida por nenhum dispositivo, sem meios de detectar tal situação – e a inexistência de endereçamento faz necessário o uso de um pino *SS* para cada escravo, o que pode ser desvantajoso se existirem muitos escravos. Além disso, o fato de que este protocolo não é multi-mestres pode torná-lo inadequado a muitos sistemas.

2.2 Cinemática

Nesta seção, o método utilizado para o estudo da cinemática inversa e direta de robôs paralelos é introduzido.

Cinemática é o estudo do movimento sem considerar as forças que o causam. Dentro do campo da cinemática, pode-se incluir, por exemplo, a análise da posição, velocidade e aceleração [20].

Para utilizar um robô em qualquer aplicação prática, a cinemática do manipulador deve ser deduzida. Para robôs seriais, a cinemática e métodos de análise desta são amplamente

estudados e, na maioria dos casos, tem aplicação simples. No entanto, no caso de robôs paralelos,

> "os métodos existentes para cinemática inversa e direta são, ou específico à cinemática de uma determinada estrutura, ou muito difícil de se implementar" [6].

Para este projeto, somente a cinemática de posição da estrutura foi estudada. O método usado é uma técnica geral e fácil de implementar introduzida em [6], e será apresentada nas seções seguintes. Baseia-se no método jacobiano clássico para manipuladores seriais, usando a notação Denavit-Hartenberg.

2.2.1 Cinemática inversa

Manipuladores seriais

Para solucionar o problema da cinemática inversa para manipuladores seriais, deve-se, primeiro, definir sistemas de eixos a cada uma das juntas do manipulador e descrever as relações entre eles. Dois sistemas de eixos sempre podem ser relacionados através de uma matriz 4×4 como em (2.12).

$$T = \begin{bmatrix} R & | & P \\ - & - & - \\ 0 & | & 1 \end{bmatrix} = \begin{bmatrix} X & Y & Z & | & P \\ - & - & - & - & - \\ 0 & 0 & 0 & | & 1 \end{bmatrix} \tag{2.12}$$

onde R é uma matriz 3×3 que corresponde à rotação em torno de cada eixo e P, uma matriz 3×1 que corresponde à diferença de posição.

Uma forma simples de determinar estas matrizes é utilizar o método Denavit-Hartenberg. Este método consiste em determinar quatro parâmetros para cada sistema de eixos[4]:

- Ângulo da junta θ_i;
- Distância entre juntas d_i;
- Comprimento do *link* a_i;
- Ângulo de revolução do *link* α_i.

A partir destes parâmetros, a matriz de transformação relacionando um *link* com o próximo pode ser calculada como em (2.13).

[4] Uma explicação mais detalhada pode ser encontrada em, por exemplo, [20].

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cdot \cos\alpha_{i-1} & \cos\theta_i \cdot \sin\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1} \cdot d_i \\ \sin\theta_i \cdot \sin\alpha_{i-1} & \cos\theta_i \cdot \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} \cdot d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

Aplicando-a a um manipulador, pode-se obter a matriz 4×4 de transformação $T_\alpha^\beta(q(t))$ relacionando o sistema de eixos β ao sistema de eixos α, através dos ângulos das juntas $q(t) \equiv (q_1(t), \cdots, q_n(t))$.

Após a determinação destas matrizes de transformação, um Jacobiano Manipulador tradicional pode ser calculado a partir de (2.14).

$$J(q(t)) \equiv \begin{bmatrix} A(q(t)) \\ - - - \\ B(q(t)) \end{bmatrix} \tag{2.14}$$

onde,

$$\begin{aligned} A(q(t))_{lj} &= \frac{\partial \left[P_{Base}^{Tool}(q(t))\right]_l}{\partial [q(t)]_j} \\ l = 1,2,3 \quad & j = 1, \cdots, M \\ B(q(t))_{lj} &= \eta_j \left[Z_{Base}^{j-1}(q(t))\right]_l \\ l = 1,2,3 \quad & j = 1, \cdots, M \end{aligned}$$

sendo η_j zero se a j'ésima junta é prismática e um, se é rotacional.

O Jacobiano Manipulador relaciona uma mudança infinitesimal num ângulo de uma junta, dq, a diferenças infinitesimais na rotação e na posição da extremidade, $d\phi$ e dp, respectivamente, como em (2.15).

$$J(q(t))dq = \begin{bmatrix} dp \\ d\phi \end{bmatrix} \tag{2.15}$$

O método numérico iterativo Newton-Raphson pode ser usado para computar a variação necessária nos ângulos das juntas para mover a extremidade pela diferença desejada na posição Δp e na rotação $\Delta\phi$.

Se o robô é redundante, o jacobiano obtido não é uma matriz quadrada, sendo, portanto, impossível invertê-la. Deve-se utilizar, então, a inversa generalizada de Moore-Penrose.

Manipuladores paralelos

O método anterior, usado para calcular a cinemática inversa de manipuladores seriais, pode ser extendida ao estudo de manipuladores paralelos.

Considera-se uma estrutura paralela geral, com L pernas conectando sua base à sua extremidade. Todas as pernas são supostamente iguais. Também é considerado que todas as juntas são prismáticas ou de rotação, e que o critério de Gruebler[5] é satisfeito.

Primeiramente, dois sistemas de coordenadas são definidos, um ligado à base da plataforma e outro à extremidade. O próximo passo consiste em aplicar o método de Denavit-Hatenberg (D-H) a uma das pernas, para determinar todas as variáveis das juntas. Com isto, é possível calcular as matrizes de transformação relacionando cada junta da perna e, multiplicando-as, a matriz de transformação relacionando a base à extremidade.

Matrizes de transformação constantes relacionando a base geral do sistema à base de cada perna e a extremidade de cada perna à extremidade geral do sistema também devem ser igualmente determinadas.

Após o cálculo de todas as matrizes de transformação, um jacobiano manipulador para cada perna pode ser computado a partir de (2.14).

Dada a estrutura física do robô, a relação (2.16) verifica-se.

$$J_1(q^{(1)})dq^{(1)} = J_2(q^{(2)})dq^{(2)} = \ldots = J_L(q^{(L)})dq^{(L)} \tag{2.16}$$

Um jacobiano manipulador geral para estruturas paralelas pode então ser construído através de (2.17).

$$J(q) = \begin{bmatrix} J_1 & -J_2 & 0 & \ldots & \ldots & \ldots & 0 \\ 0 & J_2 & -J_3 & 0 & \ldots & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & \ldots & \ldots & 0 & J_{L-1} & -J_L \\ 0 & \ldots & \ldots & \ldots & 0 & 0 & J_L \end{bmatrix} \tag{2.17}$$

O método Newton-Raphson pode então ser aplicado a (2.18):

$$J(q)\Delta q = \Delta x_\lambda \tag{2.18}$$

onde

[5] $F = \lambda\,(n - J - 1) + J$, onde n é o número total number de *links*, F é o número de graus de liberdade da extremidade, J é o número total de juntas e λ é a dimensão do espaço em que a extremidade se move em relação à base.

$$\Delta x_\lambda = \begin{bmatrix} 0_{\lambda(L-1)} \\ dp \\ d\phi \end{bmatrix}$$

e o erro é determinado por $\epsilon = \|\Delta x_\lambda\|$.

Este método é iterativo e, portanto, é importante atualizar a cada iteração não apenas q mas também a variação desejada e o jacobiano, utilizando os novos valores dos ângulos das juntas.

Deve-se observar, ainda, que este método é válido apenas para pequenas variações, devido à linearização imposta pelo jacobiano. Quando se tratar de grandes variações, o caminho deve ser repartido e o método deve repetir-se para cada um dos pequenos passos da trajetória obtida.

2.2.2 Cinemática direta

O problema da cinemática direta para estruturas paralelas também pode ser solucionado usando o jacobiano manipulador geral definido previamente. Assume-se que existem F juntas atuadas (correspondendo aos graus de liberdade) de valor conhecido e, consequentemente, $J - F$ juntas não-atuadas, onde J é o número total de juntas.

O vetor contendo os ângulos das juntas q pode, então, ser dividido em duas partes: q^A, com os valores das juntas atuadas, e q^{UA}, com os valores das juntas não-atuadas. Portanto, o jacobiano manipulador também pode ser divido em dois jacobianos menores: J^A, constituído pelas F colunas correspondendo às juntas atuadas e as $\lambda\,(L-1)$ primeiras linhas, e J^{UA}, constituído pelas $J - F$ colunas correspondendo às juntas atuadas e as $\lambda\,(L-1)$ primeiras linhas.

Para solucionar o problema da cinemática direta para qualquer variação Δq^A, o primeiro passo é calcular (2.19),

$$J^{UA}\Delta q^{UA} = -J^A\Delta q^A \tag{2.19}$$

com relação a Δq^{UA}.

Então, deve-se calcular $q^{UA}{=}q^{UA}{+}\Delta q^{UA}$ e aplicar o procedimento de Newton-Raphson a (2.20),

$$J^{UA}(q)\Delta q^{UA} = \Delta y_\lambda \tag{2.20}$$

para obter Δq^{UA}. O erro deve ser determinado a partir de (2.21),

$$\epsilon = \|\Delta y_\lambda\| \tag{2.21}$$

onde Δy_λ é definido como (2.22).

$$\Delta y_\lambda = \begin{bmatrix} \Delta p_{1,2} \\ \Delta \phi_{1,2} \\ \vdots \\ \Delta p_{L-1,L} \\ \Delta \phi_{L-1,L} \end{bmatrix} \tag{2.22}$$

sendo $\Delta p_{k,k+1}$ a diferença na posição entre as pernas k e $k+1$ e $\Delta \phi_{k,k+1}$ a diferença na rotação entre as mesmas pernas.

2.2.3 Cinemática para várias plataformas

Assumindo, agora, um robô formado por N estruturas idênticas como aquelas tratadas nas seções anteriores, posicionadas umas sobre as outras, i.e., com a extremidade de cada plataforma correspondendo à base da próxima plataforma. Nesta configuração, o método usado até agora pode ser generalizado para ser aplicado a este tipo de manipuladores.

O problema da cinemática direta pode ser tratado aplicando-se, apenas, o algoritmo descrito anteriormente a cada plataforma individualmente.

Para solucionar o problema da cinemática inversa, porém, deve-se, primeiro, redefinir $q = (q^{(1)}, \ldots, q^{(N)})^T$, onde $q^{(j)}$ representa os ângulos das juntas da j'ésima plataforma.

Então, um pré-multiplicador para cada plataforma, o qual representa a variação entre a origem atual da i'ésima extremidade e a extremidade geral, deve ser determinado através de (2.23).

$$H^{(i)}(q) = \left[\begin{array}{c|c} I_{\lambda(L^{(i)}-1)} & 0 \\ \hline 0 & \begin{bmatrix} R_0^{i-1} & [\times p_i^N] \\ 0 & R_0^{i-1} \end{bmatrix} \end{array}\right] \tag{2.23}$$

onde R_0^{i-1} é a matriz de rotação relacionando o sistema de eixos da base da $i-1'$ésima plataforma à base geral e $[\times p]$ é definido como em (2.24)

$$\begin{bmatrix} 0 & p_3 & -p_2 \\ -p_3 & 0 & p_1 \\ p_2 & -p_1 & 0 \end{bmatrix} \tag{2.24}$$

onde p_k é o k'ésimo elemento do vetor 3×1 p.

O jacobiano de cada plataforma deve ser então recalculado usando o pré-multiplicador,

como em (2.25),

$$j^{(i)}(q) \equiv H^{(i)}(q) J(q^{(i)}) \equiv \begin{bmatrix} j_1^{(i)}(q) \\ j_2^{(i)}(q) \end{bmatrix} \tag{2.25}$$

e um novo jacobiano manipulador deve ser construído, como a matriz (2.26).

$$j(q) = \begin{bmatrix} j_1^{(1)} & 0 & 0 & \dots & \dots & \dots & 0 \\ 0 & j_1^{(2)} & 0 & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 & 0 & j_1^{(N)} \\ j_2^{(1)} & j_2^{(2)} & \dots & \dots & \dots & \dots & j_2^{(N)} \end{bmatrix} \tag{2.26}$$

O método da cinemática inversa deve então ser aplicado como antes, usando este novo jacobiano manipulador.

Capítulo 3

Projeto e implementação

Este capítulo apresenta todas as considerações, restrições e decisões de projeto. Divide-se em duas seções principais: *Sistema de controle* e *Cinemática*.

3.1 Sistema de controle

O sistema de controle projetado consiste de dois dispositivos principais de controle, um microcontrolador e um FPGA, módulos de PWM e contador de posição, e um controlador PID de posição, além de um sistema de comunicação e uma unidade central de gerenciamento. Uma representação simplificada do sistema é mostrado no diagrama de blocos da Figura 3.1.

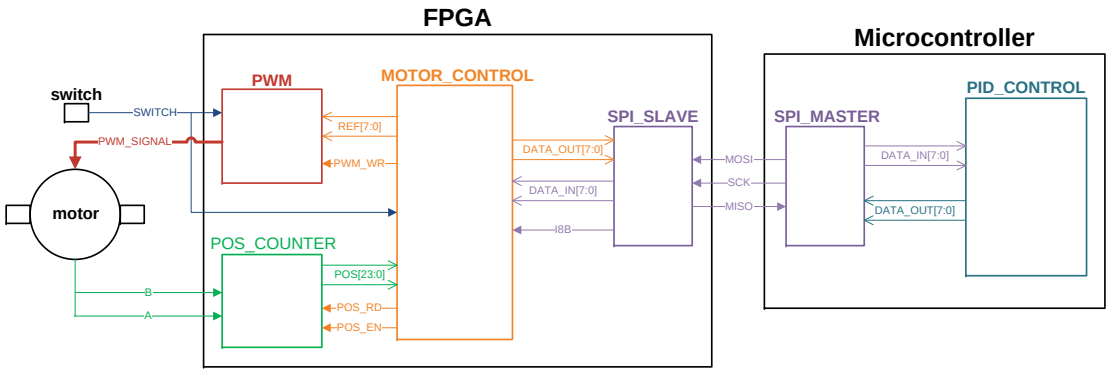


Figura 3.1: Diagrama de blocos do sistema de controle.

Como pode ser observado no diagrama de blocos, o FPGA é o dispositivo responsável por receber os sinais dos encoders do motor e, portanto, calcular a posição atual. Também gera o PWM que controlará a tensão de entrada do motor. A informação sobre a posição é, então, transmitida ao microcontrolador ATMega através das comunicações SPI.

O microcontrolador é a unidade mestre do módulo SPI e é responsável pelos cálculos do PID. O resultado do controlador PID é transmitido de volta ao FPGA, o qual ajusta a saída do PWM para o motor correspondente.

O sistema que controla os nove motores recebe e processa todos os sinais dos encoders e gera todas as saídas de PWM em paralelo. O controle PID é um ciclo contínuo que processa todos os motores sequencialmente, num intervalo de tempo apropriado.

3.1.1 Motores CC

Para projetar um controlador adequado, um modelo conveniente do motor deve ser deduzido. Os motores usados neste projeto são Maxon A-Max, modelo 236 655, $32mm$ de diâmetro, $24V$, $15W$, com um redutor 18/1 e um encoder Maxon integrados (Figura 3.2).

Figura 3.2: Conjunto motor, encoder e redutor.

Já que estão sendo utilizados nove motores e não havia instrumentos disponíveis para medir com precisão os valores de resistência e indutância do motor, os valores encontrados nas especificações técnicas do fabricante foram considerados aceitáveis para a construção do modelo.

Estes valores são $7,19\Omega$ e $1,04mH$ para resistência e indutância, respectivamente.

Ademais, também são necessárias informações sobre inércia, torque e constante elétrica. Novamente, consultando as especificações técnicas, pode-se encontrar $K_t = K_e = 38,2 \cdot 10^{-3} Nm/A$ e a inércia do conjunto – correspondendo à inércia do motor mais a inércia do redutor – $J_{total} = J_{motor} + J_{gearhead} = 2,815 \cdot 10^{-5} Nms^2/rad$.

A partir destes parâmetros, foi construído um modelo para o motor CC através de (2.5). A função de transferência resultante é, então, (3.1).

$$H(s) = \frac{2,12 \cdot 10^{-3}}{2,9276 \cdot 10^{-8} s^2 + 2,024 \cdot 10^{-4} s + 1,459 \cdot 10^{-3}} \tag{3.1}$$

3.1.2 Drivers de motores

Dadas as características mostradas na subseção 2.1.2, a configuração ponte-H é a opção mais adequada para o driver dos motores, e transistores MOSFET de potência são os dispositivos de chaveamento mais convenientes.

O componente L6203 contém um driver tipo ponte-H com transistores DMOS integrados e circuito de controle. Esta opção provê simplificação para o projeto dos circuitos elétricos, além de desempenho ótimo garantido para um conjunto de componentes equivalentes. O componente inclui, ainda, medidas de segurança, como proteção contra condução cruzada e aquecimento, as quais podem preservar os motores no caso de mal-funcionamento do sistema.

3.1.3 PWM

No projeto do PWM, há dois fatores a serem considerados: limites máximos e mínimos para a freqüência e número de níveis (resolução).

Para determinar o limite mínimo da freqüência, deve-se considerar a constante de tempo elétrica do motor, para garantir condução contínua de corrente.

A partir do modelo elétrico do motor, sua constante de tempo elétrica pode ser calculada por (3.2).

$$\tau_e = \frac{L}{R} = \frac{1,04m}{7,19} \approx 1,45 \times 10^{-4}s \tag{3.2}$$

Supondo que, para garantir a condução contínua, o período do PWM deve ser, pelo menos, cinco vezes inferior à constante de tempo elétrica do circuito, é requerida uma freqüência mínima de, aproximadamente, $35kHz$.

O número de níveis da PWM deve ser escolhido de acordo com a resolução de tensão desejada. Um valor de $0,2V$ propicia precisão suficiente a um controle suave dos motores. Assim, o número de níveis pode ser calculado através de (3.3).

$$\frac{V_{max} - V_{min}}{\text{Resolução}} = \frac{24 - (-24)}{0,2} = 240 \tag{3.3}$$

Escolheu-se, então, 256 níveis (8 bits).

O limite máximo da freqüência é determinado, essencialmente, pelas capacidades comutadoras do driver do motor e pela freqüência máxima gerada pela FPGA. Com 256 níveis, a freqüência máxima permitida pela FPGA pode ser calculada como em (3.4).

$$f_{PWM_{MAX}} = \frac{f_{FPGA}}{\text{n° de níveis}} = \frac{50 \times 10^6}{256} \approx 195kHz \tag{3.4}$$

Consultando as especificações técnicas do driver do componente L6203, a freqüência de comutação máxima indicada é de $100kHz$.

Escolheu-se, então, um valor em torno de $50kHz$ para a freqüência do PWM deste projeto, já que se encontra entre os limites máximo e mínimo estabelecidos.

O sinal de PWM é gerado como mostrado na Figura 3.3, comparando uma onda dente-de-serra com uma referência. O sinal dente-de-serra tem 256 níveis, correspondendo aos valores de referência do PWM.

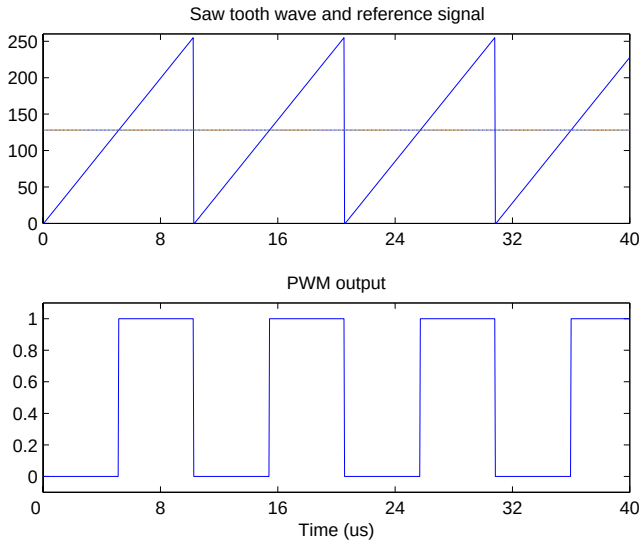


Figura 3.3: Geração do PWM a partir de uma onda dente-de-serra.

Para obter a freqüência do PWM próxima de $50kHz$ foi necessário criar um sinal de clock interno, com período quatro vezes superior ao clock do sistema. A freqüência obtida foi (3.5).

$$f_{PWM} = \frac{f_{FPGA}}{4} \cdot 256 = 48,828kHz \tag{3.5}$$

O algoritmo implementado (Figura 3.4) utiliza uma referência interna, visto que o sinal de referência é um barramento conectado a todos os módulos de todos os nove motores. A referência interna é atualizada com o valor da referência somente quando o sinal WRITE é ativado. Por razões de segurança, a referência interna muda seu valor para 128, correspondendo a $0V$, sempre que o interrruptor é pressionado. Além disso, quando o interruptor está ativo, somente valores positivos de tensão (referência superior a 128) são permitidos.

3.1.4 Encoder incremental

Os motores Maxon A-Max utilizados no projeto vêm com um encoder incremental integrado. Estes encoders têm resolução de 256 pulsos por volta, três canais com sinais complementares de saída, "line driver" e operam numa frequência máxima de $80kHz$.

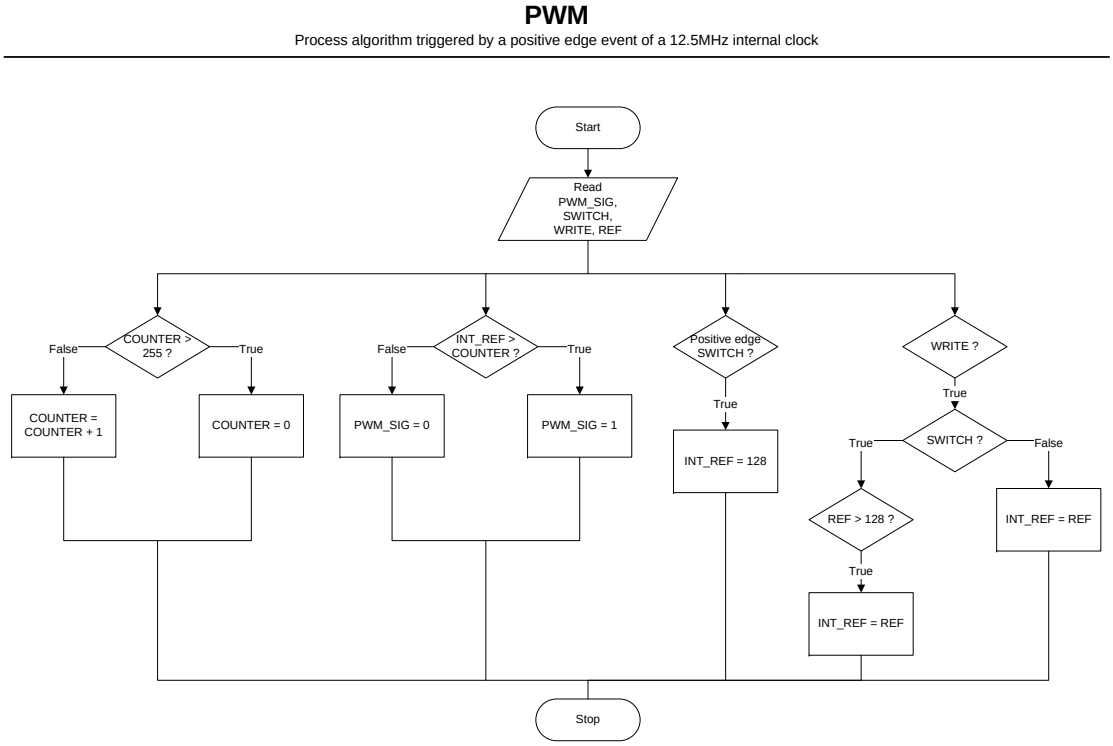


Figura 3.4: Fluxograma do algoritmo do módulo PWM.

Na tentativa de minizar problemas relacionados a ruído, um "line receiver" está sendo utilizado para os sinais A e B. O componente escolhido foi o SN75115, já que suas características atendem a todos os requisitos: operação com linhas diferenciais e compatibilidade com sinal TTL. Além disso, cada componente tem dois canais, o que facilita a integração ao sistema, pois apenas um componente é necessário para cada motor.

O algoritmo implementado determina a posição do motor com referência a um ponto inicial, o qual, neste caso, é dado por um interruptor posicionado no limite mínimo do atuador de fuso de cada motor (Figure 3.5).

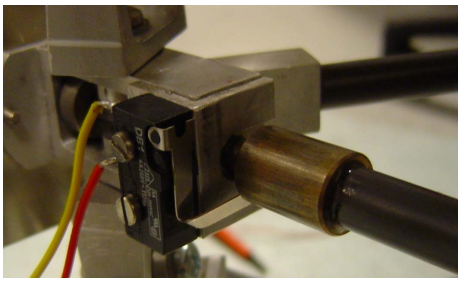


Figura 3.5: Interruptor utilizado para indicar a posição extrema do motor.

O procedimento simplificado é mostrado na Figura 3.6.

O sinal de clock do sistema utilizado no algoritmo de contador de posição tem freqüência de $50MHz$. A velocidade do motor, entretanto, não excede $5930rpm$, o que implica numa

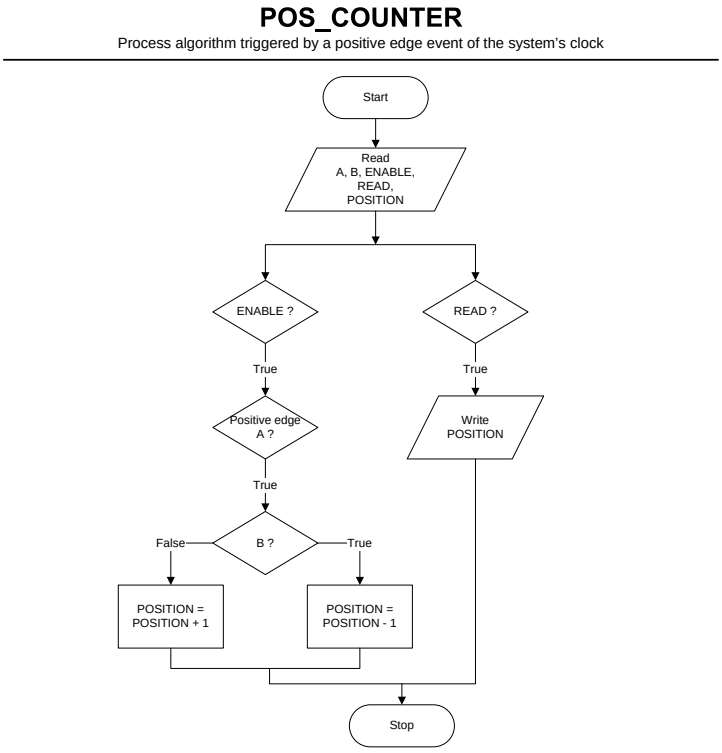


Figura 3.6: Algoritmo POS_COUNTER.

freqüência máxima de pulsação do encoder igual a $25,3kHz$. Assim, é fácil verificar que o algoritmo implementado tem uma freqüência de amostragem alta o suficiente para não perder nenhum pulso.

3.1.5 Controlador de posição

Visando a controlar a posição do motor em malha-fechada, um controlador deve ser convenientemente dimensionado. Decidiu-se utilizar para este propósito um controlador PID.

O objetivo era dimensionar um compensador que permitisse alcançar as seguintes características:

- Malha-fechada estável;
- Sem erro em estado permanente, com relação ao degrau;
- "Overshoot" inferior a 5%, com relação ao degrau;
- Boa resposta a pertubações externas, como variações no sinal de alimentação e na carga.

O "software" Matlab Simulink© foi utilizado para simular o sistema de controle, baseado no modelo matemático deduzido, e para ajustar os termos de ganho PID. A simulação mostrou que apenas um ganho proporcional era suficiente para obter a resposta desejada do sistema (Figura 3.7). No entanto, pequenos ajustes experimentais foram feitos para melhorar a resposta do sistema físico. O valor final para o ganho do controlador proporcional, ajustado empiricamente, é $0,04$.

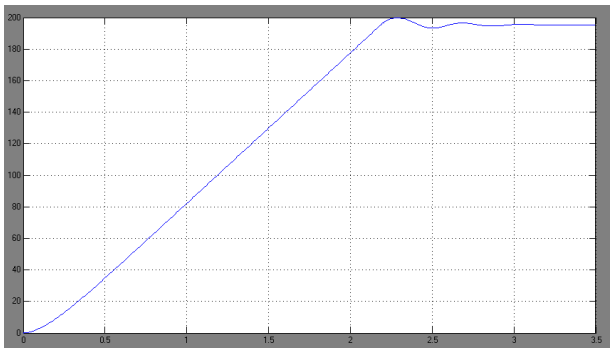


Figura 3.7: Simulação do controlador de posição.

3.1.6 FPGA e microcontrolador

Seria bastante complicado usar microcontroladores para implementar sistemas de alta complexidade que envolvem diversos motores, implicando em:

- gerar todos os sinais de controle PWM;
- receber todas as entradas dos encoders;
- fazer todos os cáculos de posição;
- e, ainda, implementar as malhas de controle.

Nestes casos, o uso de um semicondutor programável como o FPGA é muito conveniente, visto que é possível executar a maior parte destas tarefas de forma muito mais rápida e confiável, sendo o FPGA um conjunto de blocos lógicos altamente integrados.

O FPGA Xilinx Spartan 3 foi escolhido para este projeto, dadas as suas milhares de portas lógicas disponíveis, além do fato de conter um *kit* de desenvolvimento, o que torna a conexão com outros dispositivos mais fácil, a partir das suas portas de entrada e saída já instaladas.

O microcontrolador escolhido foi o Atmel ATmega128. Este microcontrolador tem un sinal de clock de 16MHz e módulos de comunicação SPI e serial, características que permitem a implementação do controlador de posição dos motores com tempo de amostragem conveniente e comunicação fácil com o FPGA.

3.1.7 Comunicações SPI

Para conseguir controlar o sistema, o microcontrolador precisa receber informação sobre as posições dos motores, coletadas pelo FPGA, e enviar de volta os valores ajustados de PWM para cada motor. Portanto, como a comunicação deve ser rápida e há apenas um mestre e um escravo, o protocolo SPI foi escolhido.

O estabelecimento de uma comunicação eficiente depende do ajuste de diversos parâmetros do protocolo SPI. A escolha destes parâmetros afeta principalmente a implementação SPI no FPGA e não segue nenhuma regra particular. Assim, decidiu-se usar o modo de operação 3 (mostrado na subseção 2.1.7), transmitindo o bit mais significativo primeiro e operando a $125kHz$ de freqüência.

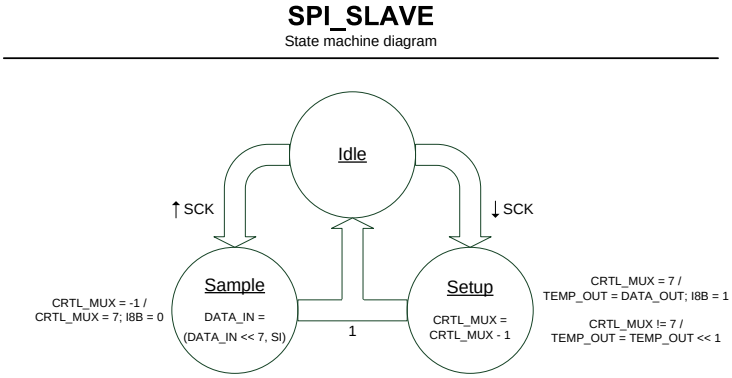


Figura 3.8: Diagrama de estados SPI_SLAVE.

O diagrama de estados implementado (Figura 3.8) segue os parâmetros definidos para a comunicação SPI.

Um sinal interno I8B é ativado para sinalizar o fim da transmissão de um byte, permitindo o funcionamento adequado do módulo de controle do FPGA, MOTOR_CONTROL.

3.1.8 Algoritmos do sistema de controle

O sistema de controle é constituído por dois algoritmos principais, um implementado no FPGA (MOTOR_CONTROL) e o outro implementado no microcontrolador ATMega (PID_CONTROL). Os diagramas de estado de cada um desses módulos podem ser vistos na Figura 3.9 e na Figura 3.10.

O FPGA é iniciado no estado STOP, com todos os motores parados e os contadores desativados. Quando o microcontrolador envia o sinal 0xFA, o sistema parte para a inicialização. Durante este processo, o ATMega transmite a mensagem 0xAA, para permitir que o módulo escravo da comunicação SPI no FPGA possa enviar a mensagem 0xF0, sinalizando o fim do processo de inicialização.

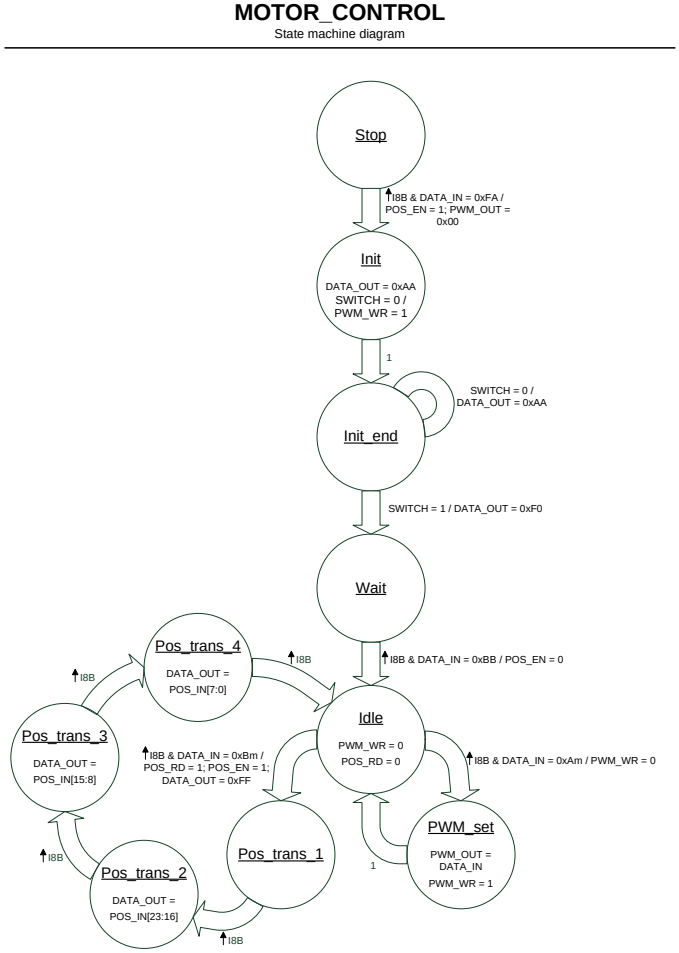


Figura 3.9: Diagrama de estados MOTOR_CONTROL.

Em seguida, o FPGA entra num estado WAIT, o qual tem duração definida é pelo microprocessador. Este estado é finalizado quando o FPGA recebe 0xBB.

Após este processo, o ciclo sequencial de controle dos motores é iniciado. O microcontrolador envia uma mensagem formada pela letra B seguida de um número correspondente ao motor desejado (representado por m nos diagramas), o qual indica ao FPGA a posição de qual motor ele deve transmitir. Este processo é seguido por quatro mensagens 0xAA do ATMega, permitindo a emissão dos três bytes que compõem o valor da posição atual.

O módulo PID_CONTROL, então, baseado nesta informação, calcula a tensão de saída apropriada dada pelo compensador PID e envia à função SET_PWM, a qual converte a tensão num nível de referência PWM apropriado e transmite-o ao FPGA. A transmissão é feita, primeiro, enviando-se uma mensagem para o FPGA formada pela letra A seguida do número correspondente ao motor, e, logo após, outra mensagem, com o valor do PWM.

Esta informação, após ser recebida pelo FPGA, é processada no módulo PWM para

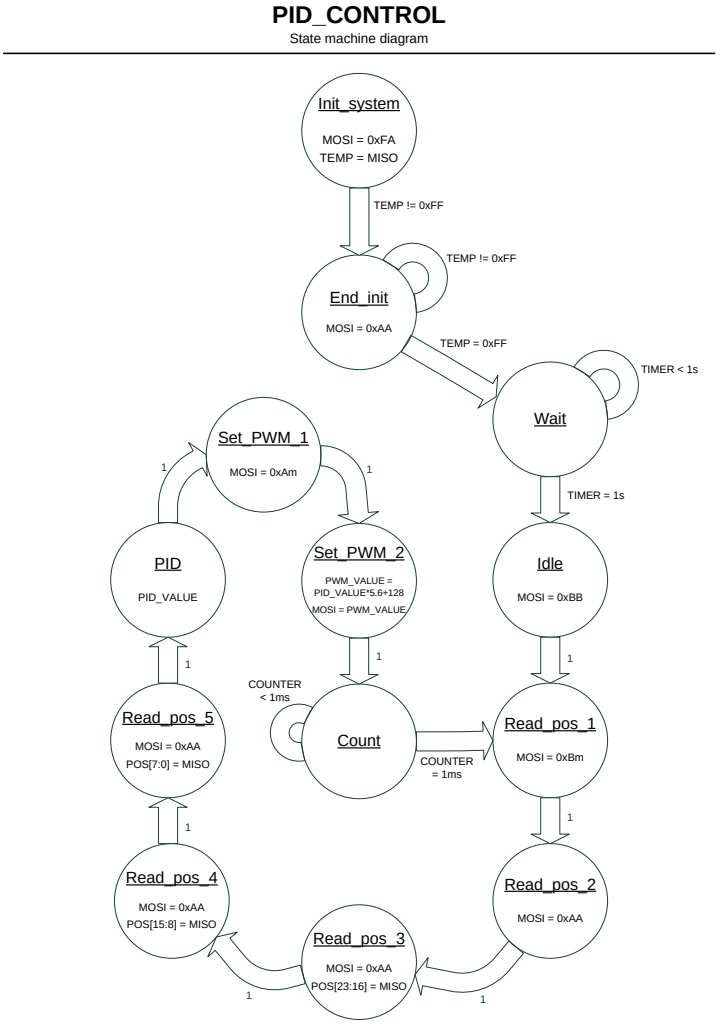


Figura 3.10: Diagrama de estados PID_CONTROL.

gerar um sinal adequado de PWM para o motor.

O controlador de posição foi dimensionado para controlar o número de voltas do motor. No entanto, a saída do sistema é uma variação linear de um atuador de fuso. Para transformar o valor rotacional na quantidade linear correspondente, foram realizados alguns experimentos e definiu-se empiricamente o valor $0,08cm/volta$.

3.1.9 Implementação do sistema de controle

Devida à estrutura física única do robô, decidiu-se projetar e construir quatro placas de circuito impresso ("Printed Circuit Board" – PCB). Três destas seriam usadas para receber os sinais dos encoders e dos interruptores, além de alimentar cada um dos três motores presentes em cada plataforma. A quarta PCB recolheria os sinais das placas dos motores e transmitiria-os à estapa de controle do sistema, assim como, geraria todas as

tensões de alimentação através de drivers dos motores, utilizando os sinais recebidos pelo FPGA.

No entanto, como a tecnologia de construção de PCB disponível na instituição é muito limitada, foi necessário dividir a quarta placa em duas placas, uma contendo os drivers dos motores e a outra, todos os dispositivos de controle. A divisão foi feita desta maneira para separar os componentes digitais dos dispositivos de chaveamento, minimizando, detsa maneira, a interferência eletromagnética entre elas.

A tecnologia de construção de PCB mencionada não permite mais do que duas camadas, tem limitação do tamanho correspondente a uma folha de papel tipo A4 e, como não há preenchimento de cobre nos orifícios, todas as trilhas conectadas aos componentes precisam estar na camada inferior. Todas essas restrições tornam o projeto de layout de PCB muito complicado e o processo de montagem demanda muito tempo, visto que as placas precisam ser manualmente perfuradas e os orifícios preenchidos com fio de cobre, para conectar as camadas superior e inferior.

Os sinais precisam ser conduzidos entre as placas separadas através de "flat cables". Devido ao posicionamento das placas, alguns desses cabos são muito longos, o que pode introduzir problemas, causados principalmente pela natureza ruidosa do sistema.

A posição ideal dos "line receivers" dos sinais dos encoders seria na placa de controle, o mais próximo possível do FPGA. Tal posição, porém, implicaria em cabos ainda maiores, o que não é possível devido a limitações físicas. Assim, os "line receivers" foram colocados nas placas dos motores e os sianis foram transportados através de "flat cables" após a reconstrução. Para minimizar a interferência de ruído quando estes sinais são recebidos pelo FPGA, as portas de entrada do FPGA foram configuradas internamente com terminação "pull-down".

Após a construção das placas, em muitas ocasiões foram encontradas trilhas quebradas e curto-circuitos entre as trilhas ou vias e o plano de massas. A detecção e correção destes problemas foi a etapa do processo de implementação em que se gastou mais tempo.

Além disso, observou-se, após a implementação, que a fonte de $24V$ estava sendo afetada por ruído eletromagnético significativo, induzido pelas comutações em alta freqüência dos drivers dos motores. Para minizar este problema, colocou-se um capacitor de $100\mu F$ entre os terminais da fonte.

Os esquemáticos e layouts finais das cinco placas de circuito impresso podem ser encontraddos no Apêndice A. As placas montadas podem ser vistas na Figura 3.11 à Figura 3.13.

3.2 Cinemática

Algumas considerações devem ser feitas para se conseguir um modelo do robô conve-

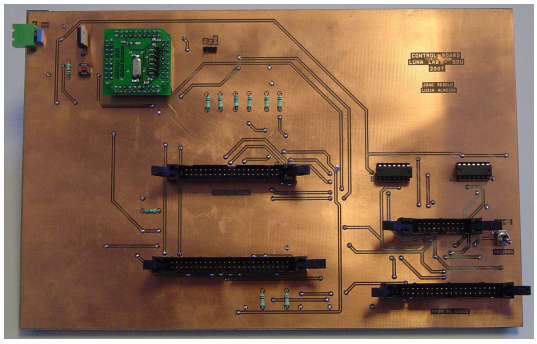


Figura 3.11: Placa de controle.

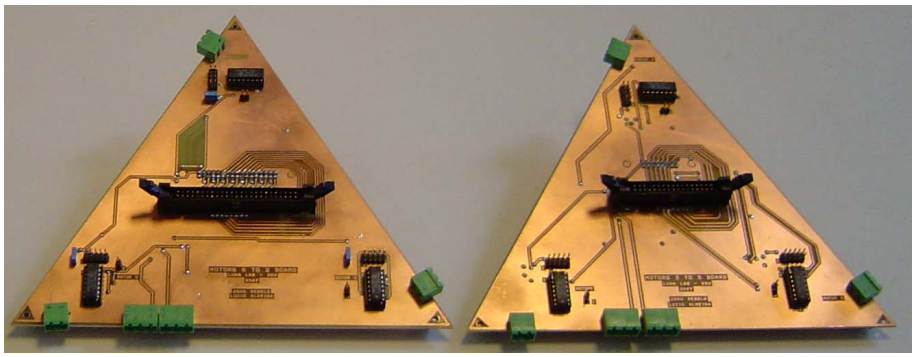


Figura 3.12: Placas de motores.

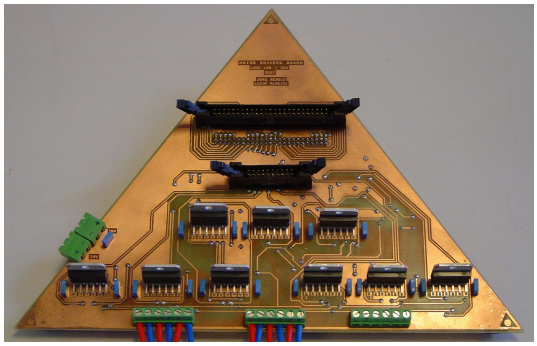


Figura 3.13: Placa dos drivers de motores.

niente para o estudo da cinemática.

As três plataformas de que o robô é constituído são consideradas idênticas e numa versão simplificada. As juntas físicas são muito complexas para serem estudadas analiticamente, logo, um modelo matemático imaginário correspondente foi deduzido para a cinemática do robô.

O modelo matemático de cada plataforma consiste de uma base e uma extremidade, ambos no formato triangular equilátero, conectados por três pernas. Cada perna é constituída por três juntas, duas de revolução e uma esférica. Estas pernas imaginárias são conectadas na bissetriz de cada lado da base física, e neste mesmo ponto é colocado cada vértice da base matemática.

A junta esférica foi, mais tarde, substituída por três juntas, duas cilíndricas e uma de revolução, conectadas por links de de comprimento nulo. Desta forma, todas as juntas têm um grau-de-liberdade cada, tornando o modelo matemático mais simples de ser deduzido.

A extremidade de cada plataforma é considerada a própria base da plataforma seguinte. Tal suposição não corresponde à realidade, uma vez que há uma distância de aproximadamente $4cm$ entre plataformas subsequentes, relacionada com as conexões físicas.

As difereças e relações entre os modelos físico e matemático podem ser vistas na Figura 3.14, onde o modelo físico é representado na cor azul e o matemático, em laranja.

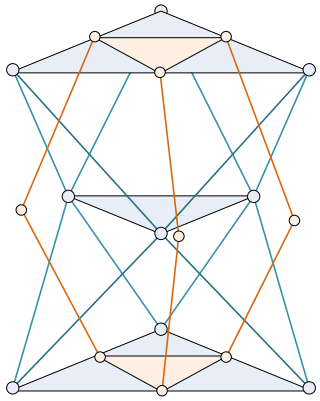


Figura 3.14: Relação entre os modelos físico (em azul) e matemático (em laranja).

A formulação Denavit-Hartenberg foi utilizada para descrever os mecanismos dos links, e sistemas de coordenadas foram definidos para relacionar a posição de um link com seus vizinhos, como ilustrado na Figura 3.15[1]. Apenas os eixos X e Z são mostrados, porém o eixo Y pode ser facilmente encontrado através da regra da mão direita.

Seguindo o procedimento da convenção D-H, os parâmetros mostrados na Tabela 3.1 foram determinados, de acordo com as coordenadas definidas previamente.

Sistemas de coordenadas	a_{i-1}	α_{i-1}	d_i	θ_i
$0 \rightarrow 1$	0	0^o	0	θ_1
$1 \rightarrow 2$	0	-90^o	$-L$	θ_2
$2 \rightarrow 3$	$-a_2$	90^o	0	θ_3
$3 \rightarrow 4$	$-a_3$	-90^o	L	θ_4
$4 \rightarrow 5$	0	-90^o	0	θ_5

Tabela 3.1: Parâmetros Denavit-Hatenberg.

Os ângulos das juntas resultantes e distâncias calculadas podem ser observadas na Figura 3.16.

[1] Juntas 1,3 e 5 são de revolução e juntas 2 e 4, cilíndricas

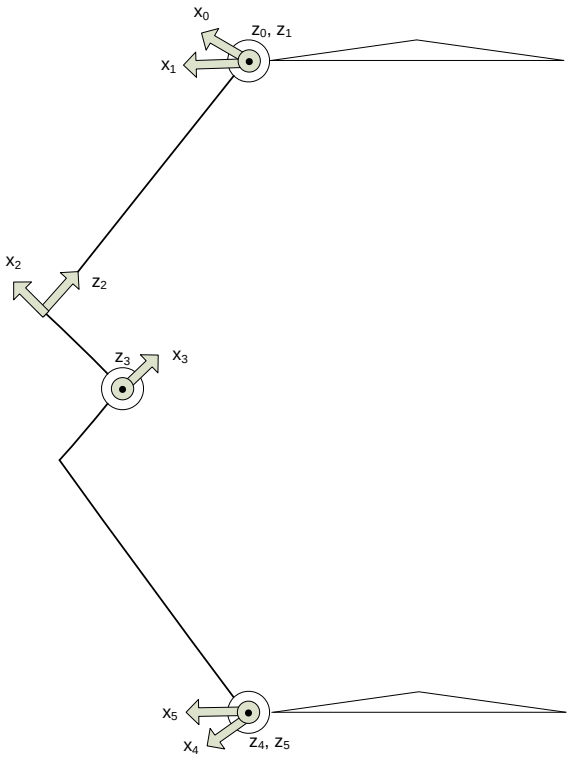


Figura 3.15: Sistemas de coordenadas das juntas.

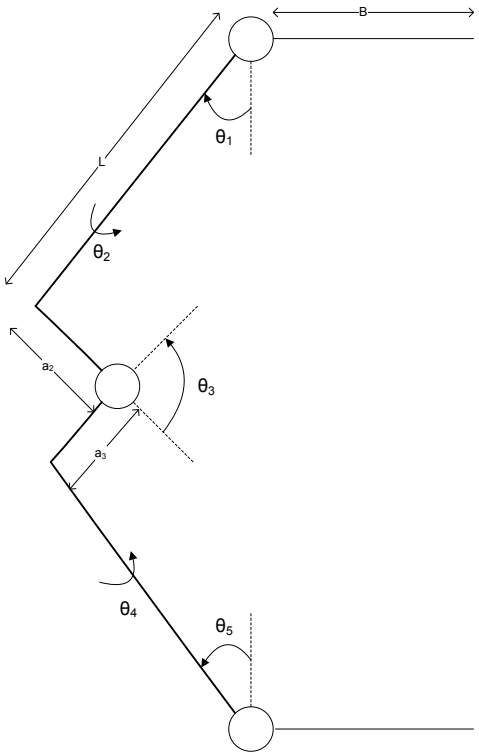


Figura 3.16: Ângulos das juntas.

Como pode ser observado na Figura 3.16, na Figura 3.16 e na Figura 3.17, a_2 e a_3 correspondem a "offsets" mecânicos entre o fim da perna e a junta esférica imaginária. Eles são considerados no modelo matemático, contudo, para simplificar, estes "offsets" são ajustados a um valor nulo durante a computação dos algoritmos da cinemática. No entanto, no futuro, tais valores podem ser experimentalmente ajustados para melhor calibrar a cinemática e obter-se resultados mais precisos.

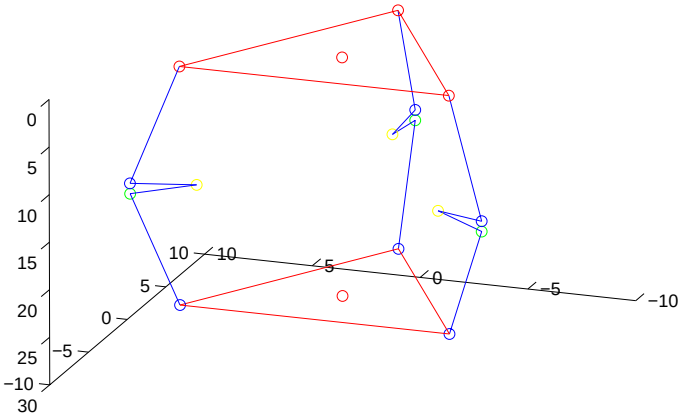


Figura 3.17: Simulação do modelo matemático de uma plataforma com $a_2 = a_3 = 2cm$.

A partir dos parâmetros D-H determinados, foram obtidas as matrizes 4×4 de transformação, relacionando cada sistema de coordenadas, como mostrado em (3.6) a (3.10).

$$T_0^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$T_1^2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & -L \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$T_2^3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ 0 & 0 & -1 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$T_3^4 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ 0 & 0 & 1 & L \\ -\sin\theta_4 & -\cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$T_4^5 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_5 & -\cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Como foi referido anteriormente, cada plataforma contém uma base, uma extremidade e três pernas idênticas. Assim, para relacionar a base a cada uma das pernas, deve-se colocar o sistema de coordenadas da base no seu centro e calcular, diretamente, as transformações para cada uma das pernas (Figura 3.18). As matrizes de transformação resultantes são (3.11).

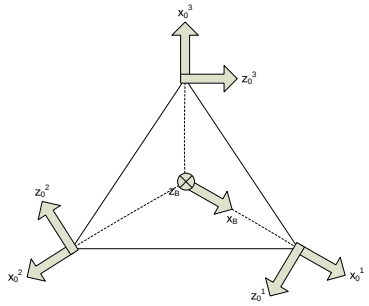


Figura 3.18: Relação entre as pernas e a base.

$$\begin{aligned} T_{0^{(1)}}^B &= \begin{bmatrix} 1 & 0 & 0 & B/\sqrt{3} \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ T_{0^{(2)}}^B &= \begin{bmatrix} -1/2 & 0 & -\sqrt{3}/2 & -B/2\sqrt{3} \\ \sqrt{3}/2 & 0 & -1/2 & B/2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ T_{0^{(3)}}^B &= \begin{bmatrix} -1/2 & 0 & \sqrt{3}/2 & -B/2\sqrt{3} \\ -\sqrt{3}/2 & 0 & -1/2 & -B/2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.11}$$

Visto que a extremidade é idêntica e simétrica à base, a forma mais fácil de relacionar o último sistema de coordenadas de cada perna à extremidade é, simplesmente, colocar o sistema de coordenadas da extremidade na mesma posição que o sistema da base e calcular a transformação inversa da base. As matrizes de tranformação resultantes são (3.12), e uma ilustração do sistema da extremidade pode ser vista na Figura 3.19.

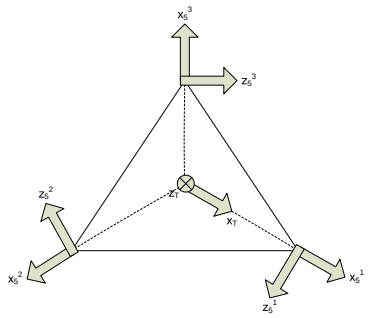


Figura 3.19: Relação entre as pernas e a extremidade.

$$
\begin{aligned}
T_T^{0(1)} &= \begin{bmatrix} 1 & 0 & 0 & -B/\sqrt{3} \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
T_T^{0(2)} &= \begin{bmatrix} -1/2 & \sqrt{3}/2 & 0 & -B/\sqrt{3} \\ 0 & 0 & -1 & 0 \\ -\sqrt{3}/2 & -1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
T_T^{0(3)} &= \begin{bmatrix} -1/2 & -\sqrt{3}/2 & 0 & -B/\sqrt{3} \\ 0 & 0 & -1 & 0 \\ \sqrt{3}/2 & -1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{3.12}
$$

De acordo com 2.17, e após o cálculo do jacobiano padrão para cada uma das três pernas, o jacobiano manipulador de uma plataforma do robô considerado consiste de 15 variáveis (5 variáveis para cada perna) e, portanto, é uma matriz de blocos 18×15, como em (3.13).

$$
J(q) = \begin{bmatrix} J_1 & -J_2 & 0 \\ 0 & J_2 & -J_3 \\ 0 & 0 & J_3 \end{bmatrix} \tag{3.13}
$$

3.2.1 Cinemática inversa

Uma vez determinadas as transformações dos sitemas de coordenadas e o jacobiano manipulador, a cinemática inversa para pequenas variações foi computada através do método Newton-Raphson, como mostrado no Algoritmo 1.

Algorithm 1 Cinemática inversa.

```
function inverse_kinematics(q, T, T_desired)
Δp = position_displacement(T, T_desired);
Δφ = angle_displacement(T, T_desired);
Δx = (Δp, Δφ);
error = |Δx|;
while error > ϵ do
    J = J(q);
    Δq = inverse(J) * Δx;
    q = q + Δq;
    T = T(q);
    Δp = position_displacement(T, T_desired);
    Δφ = angle_displacement(T, T_desired);
    Δx = (Δp, Δφ);
end while
return q;
```

A função "position_displacement" referida no algoritmo apenas calcula a diferença entre as partes relativas à posição das transfomações atual e desejada.

A função "angle_displacement" é análoga, porém, como não é possível subtrair ângulos diretamente, foi usada a representação eixo-ângulo ("angle-axis representation"). Como apenas pequenas variações são consideradas, a aproximação para pequenos ângulos foi feita, resultando na equação (3.14) de equivalência eixo-ângulo.

$$u_{EAA}(R) = \frac{1}{2}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{3.14}$$

A função inversa usada neste caso foi a inversa generalizada (ou inversa Moore-Penrose), já que o jacobiano é uma matriz 18×15, como mostrado previamente.

O erro máximo escolhido foi $\epsilon = 10^{-10}$, visto que o método Newton-Raphson converge quadraticamente.

Para calcular a transformação desejada necessária no Algoritmo 1, pode-se usar, por exemplo, a convenção de ângulos fixos X-Y-Z para determinar a porção da rotação a partir dos ângulos de rotação desejados α, β e γ correspondendo a cada um dos eixos X, Y e Z, respectivamente. Isto resulta em (3.15) [20].

$$R_{XYZ} = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{3.15}$$

As transformações desejadas podem ser obtidas a partir de (2.12), para uma posição desejada (x, y, z) e uma rotação desejada correspondente aos ângulos (α, β, γ), resultando em (3.16).

$$T = \left[\begin{array}{cc|c} & & x \\ R_{XYZ}(\alpha,\beta,\gamma) & & y \\ & & z \\ \hline 0 & & 1 \end{array}\right] \tag{3.16}$$

Para uma plataforma, considerou-se como variação arbitrária diferenças na posição superiores a $0,25cm$ ou variações angulares maiores que $2°$.

Quando uma destas situações ocorre, um caminho constituído de pequenos passos deve ser calculado e, então, o algoritmo inverso (Algoritmo 1) é chamado para cada um dos passos definidos. O algoritmo completo para variações arbitrárias pode ser visto no Algoritmo 2.

As funções "position_extraction" e "rotation_extraction" referidas no algoritmo, simplesmente extraem as porções relativas à posição e à rotação, respectivamente, das matrizes de transformação.

As funções "quartenion" são utilizadas para permitir a interpolação de rotações. São vetores 4×1 formados pelos parâmetros equivalentes de Euler para matrizes de rotação [20].

Pode-se observar do Algoritmo 1 e do Algoritmo 2, que o método Newton-Raphson utiliza variações e, portanto, requer a definição de uma posição inicial para funcionar. Para o robô considerado, decidiu-se que a posição inicial do sistema seria aquela em que todos os atuadores de fuso teriam comprimento mínimo, i.e., com os interruptores pressionados.

Nesta configuração, mediu-se a base física do robô, o comprimento físico do atuador e a altura do atuador até a base. A partir destas medidas, os valores usados no modelo matemático foram determinados.

Da Figura 3.20, é possível verificar que o valor da base matemática, B, é metade do valor da base física, B_T, já que ambas são triângulos equiláteros e estão posicionados como mostrado na Figura 3.20. Mediu-se $B_T = 24.2cm$, resultando, assim, em $B = 12.2cm$.

Mediu-se, ainda, o comprimento do atuador $L_{act} = 16.2cm$ e a altura entre o atuador

Algorithm 2 Interpolação de pontos para variações arbitrárias.

```
function inverse_kinematics_arbitrary(q, T, T_desired)
Pinitial = position_extraction(T);
Pfinal = position_extraction(T_desired);
Rinitial = rotation_extraction(T);
Rfinal = rotation_extraction(T_desired);
R = transpose(Rinitial) * Rfinal;
θ = acos((R11 + R22 + R33 − 1)/2);
Qinitial = quartenion(T);
Qfinal = quartenion(Tdesired);
sp = 0,25/ |Pfinal − Pinitial|;
Mp = ceil(1/sp);
sr = 2/θ;
Mr = ceil(1/sr);
M = max(Mp, Mr);
for i=1 to M do do
    s = i/M;
    Pi = Pi+s*(Pdes − Pi);
    Ri = Qinitial+s*(Qfinal − Qinitial)/ |Qinitial + s * (Qfinal − Qinitial)|;
    Ti = (Pi, Ri);
    q = inverse_kinematics(q, T, Ti);
    T = Ti;
end for
return q;
```

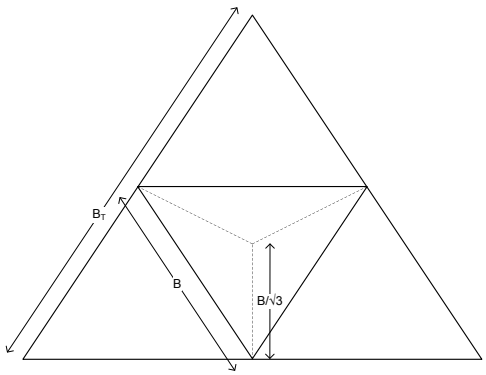


Figura 3.20: Bases real e imaginária.

e a base, $H = 12.1cm$ (Figura 3.21).

O comprimento da perna, L, pode ser calculado através de (3.17).

$$L = \frac{H}{\cos\theta_1} \tag{3.17}$$

O valor inicial do ângulo θ_1 pode ser calculado baseando-se nestas medidas e na vista superior da plataforma (Figura 3.22), resultando em (3.18).

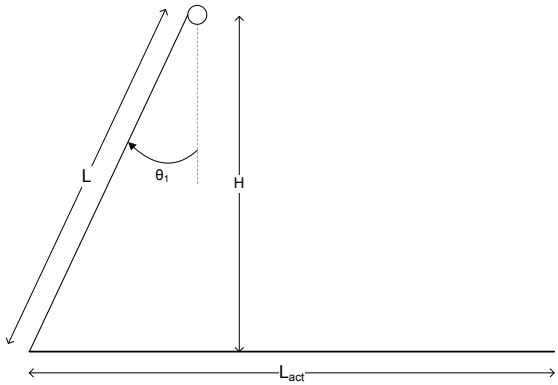


Figura 3.21: Relação entre o comprimento da perna e o ângulo de junta θ_1.

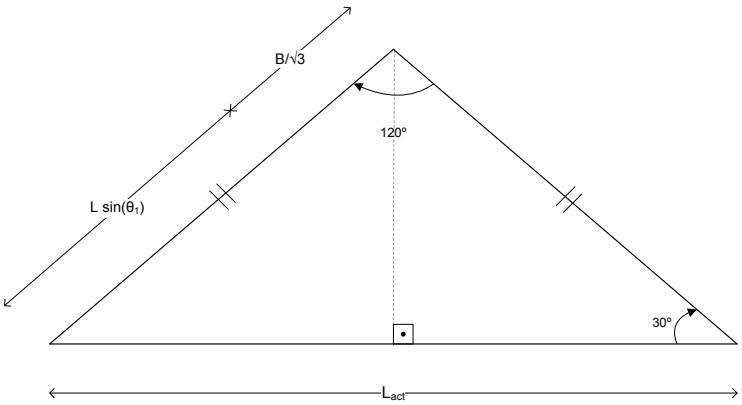


Figura 3.22: Cálculo da posição inicial.

$$\theta_1 = \arcsin\left(\frac{1}{L} \cdot \left(\frac{L_{act}}{2 \cdot \cos\left(30\right)} - \frac{B}{\sqrt{3}}\right)\right) \tag{3.18}$$

Os outros ângulos de junta θ_3 e θ_5 podem ser encontrados através das relações (3.19).

$$\begin{aligned} \theta_5 &= \theta_1 \\ \theta_3 &= \pi - (\theta_1 + \theta_5) \end{aligned} \tag{3.19}$$

e θ_2 é considerado igual a 0 °, enquanto $\theta_4 = \pi rad$, nesta configuração inicial.

Portanto, a posição inicial é dada pelos ângulos de junta:

$$
\begin{aligned}
\theta_1 &= 0.1886\ rad \\
\theta_2 &= 0\ rad \\
\theta_3 &= 2.7644\ rad \\
\theta_4 &= \pi\ rad \\
\theta_5 &= 0.1886\ rad
\end{aligned}
$$

Esta configuração iniciaç foi simulada no *software* Matlab©, resultando na ilustração mostrada na Figura 3.23.

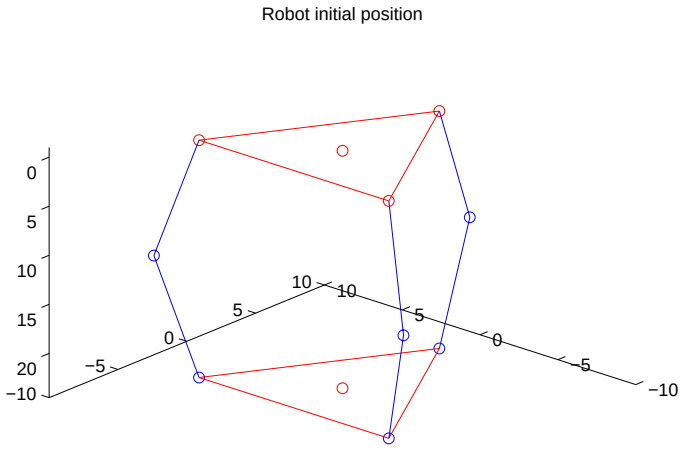


Figura 3.23: Simulação da posição inicial de uma plataforma.

3.2.2 Cinemática direta

No problema da cinemática direta, decidiu-se atuar sempre os ângulos θ_1 de cada perna. Desta forma, q^A é um vetor de dimensão 3 e q^{UA}, um vetor de dimensão 12. Assim, o jacobiano atuado consiste de uma matriz 12×3, enquanto o jabobiano não-atuado é uma matriz 12×12, já que as últimas seis linhas sorrespondentes à variação desejada são desconsideradas na cinemática direta.

A partir da definição destas matrizes, o Algoritmo 3 foi, então, implementado.

De forma semelhante ao algoritmo da cinemática inversa (Algoritmo 1), este utiliza o método numérico Newton-Raphson e, assim, necessita de uma posição inicial para calcular a primeira variação. A partir do cálculo do primeiro vetor q^{UA}, considera-se que o vetor q^A já alcançou a posição desejada final e é preciso realizar, apenas, pequenos ajustes para alcançar as posições desejadas do ângulos de juntas não-atuadas, dentro de uma certa

Algorithm 3 Cinemática direta.

```
function forward_kinematics(qAinitial, qUAinitial, T, qAdesired)
qUA = qUAinitial;
qA=qAinitial;
JUA = JUA(qUA, qA);
JA = JA(qUA, qA);
ΔqA = qAdesired - qAinitial;
ΔqUA = inverse(JUA) * (−JA * ΔqA);
qUA = qUA + ΔqUA;
qA = qAdesired;
JUA = JUA(qUA, qA);
Δp = position_displacement(T1, T2, T3);
Δφ = angle_displacement(T1, T2, T3);
Δy = (Δp, Δφ);
ΔqUA = inverse(JUA) * Δy;
error = |Δy|;
while error > ϵ do
    qUA = qUA + ΔqUA;
    qA = qAdesired;
    JUA = JUA(qUA, qA);
    Δp = position_displacement(T1, T2, T3);
    Δφ = angle_displacement(T1, T2, T3);
    Δy = (Δp, Δφ);
    ΔqUA = inverse(JUA) * Δy;
    error = |Δy|;
end while
return q;
```

margem de erro ϵ. O valor escolhido para ϵ foi também de 10^{-10}, como no algoritmo da cinemática inversa.

As funções "position_displacement" e "angle_displacement" são as mesmas utilizadas no algoritmo da cinemática inversa. No entanto, neste caso, as variações são calculadas entre as pernas, para se certificar de que as equações de malha-fechada 2.16 se mantêm.

É possível utilizar o algoritmo da cinemática direta a partir da medida do comprimento dos atuadores. Os ângulos das juntas θ_1 podem ser determinados através da vista superior da plataforma (Figura 3.24), usando a Lei dos Cossenos, e resultando no sistema de três equações e três variáveis (3.20), onde $\theta_{1,Li}, i = 1, 2, 3$ é o ângulo θ_1 de cada perna.

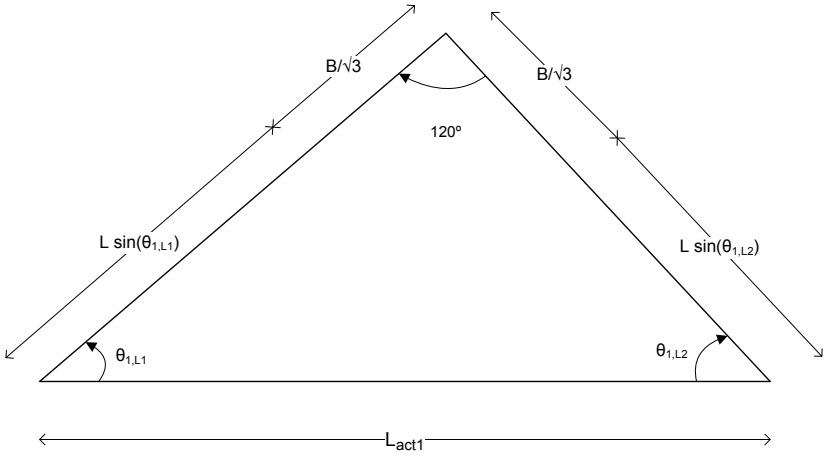


Figura 3.24: Cáculo dos ângulos a partir do comprimento dos atuadores.

$$
\begin{aligned}
L_{act1}^2 &= \left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L1}\right)^2 + \left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L2}\right)^2 \\
&\quad - 2\cdot\left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L1}\right)\cdot\left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L2}\right)\cdot\cos 120 \\
L_{act2}^2 &= \left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L2}\right)^2 + \left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L3}\right)^2 \\
&\quad - 2\cdot\left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L2}\right)\cdot\left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L3}\right)\cdot\cos 120 \\
L_{act3}^2 &= \left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L1}\right)^2 + \left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L3}\right)^2 \\
&\quad - 2\cdot\left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L1}\right)\cdot\left(\frac{B}{\sqrt{3}} + L\cdot\sin\theta_{1,L3}\right)\cdot\cos 120
\end{aligned}
\tag{3.20}
$$

3.2.3 Cinemática para várias plataformas

Como referido na subseção 2.2.3, para solucionar o problema da cinemática inversa para um robô constiuído por várias plataformas, matrizes pré-multiplicadora devem ser calculadas para cada jacobiano de cada plataforma e um novo jacobiano manipulador deve ser, então, determinado, envolvendo todas as plataformas.

O robô utilizado neste projeto é formado por três plataformas; entretanto, duas plataformas apenas podem ser controladas neste momento, pois há somente seis motores disponíveis. Assim, duas situações distintas serão consideradas:

- robô com duas plataformas, consistente com a situação atual, considerando a extremidade geral do robô como sendo a extremidade da segunda plataforma;

- robot com três plataformas e nove atuadores, correspondendo ao robô completo.

É mencionada, apenas, a cinemática inversa, visto que a cinemática direta para várias plataformas consiste, apenas, em aplicar o Algoritmo 3 para cada plataforma individualmente e computar a matriz de transformação final.

Duas plataformas

As matrizes pré-multiplicadoras obtidas, considerando o robô formado por duas plataformas, são (3.21) e (3.22).

$$H^{(1)}(q) = \left[\begin{array}{c|c} I_{12} & 0 \\ \hline 0 & \begin{bmatrix} I_3 & [\times p_1^2] \\ 0 & I_3 \end{bmatrix} \end{array}\right] \tag{3.21}$$

$$H^{(2)}(q) = \left[\begin{array}{c|c} I_{12} & 0 \\ \hline 0 & \begin{bmatrix} R_0^2 & 0 \\ 0 & R_0^2 \end{bmatrix} \end{array}\right] \tag{3.22}$$

onde p_1^2 é a porção de posição da matriz de transformação que relaciona a base da segunda plataforma à sua extremidade, visto que a base da segunda plataforma é a extremidade da primeira plataforma e a extremidade da segunda plataforma é a extremidade geral do robô; e R_0^2 é a porção de rotação da matriz de transformação que relaciona a base da primeira plataforma à sua extremidade, ou, em outras plavras, a base geral à extremidade da primeira plataforma.

Tendo determinado os pré-multiplicadores, o novo jacobiano manipulador foi computado a partir de 2.26, resultando em (3.23).

$$j(q) = \begin{bmatrix} j_1^{(1)} & 0 \\ 0 & j_1^{(2)} \\ j_2^{(1)} & j_2^{(2)} \end{bmatrix} \tag{3.23}$$

O problema da cinemática inversa pode então ser resolvido utilizando-se o Algoritmo 1, onde o jacobiano manipulador é, agora, a matriz 30×30 calculada previamente. Tendo em vista que o robô dobrou de tamanho, devido ao empilhamento das plataformas, o passo mínimo para variações arbitrárias é de $0,5cm$.

No entanto, quando o algoritmo foi implementado, ele não convergiu como esperado. Isto se deve, provavelmente, a redundância de graus-de-liberdade na estrutura. Ou seja, apesar de haver seis atuadores, devido à construção mecânica da estrutura, apenas cinco ou menos graus-de-liberdade estão disponíveis. Foi possível verificar tal fato, observando

que o jacobiano manipulador sempre tinha um autovalor muito próximo a zero, em diversas configurações diferentes.

Para resolver este problema, uma nova junta foi criada para adicionar um grau-de-liberdade. Esta junta permite somente rotação em torno do eixo Z, movimento que não é muito útil na maior parte das aplicações. Assim, uma junta de revolução teórica foi posicionada na extremidade geral e incluída no jacobiano manipulador na forma de uma nova coluna. O método iterativo, então, ajusta o valor dessa junta extra para alcançar a posição desejada, mas, como esta junta não existe fisicamente, seus movimentos não interferem no movimento do robô e seu valor é simplesmente descartado no módulo de controle dos atuadores.

Após adicionar esta junta, o manipulador jacobiano obtido é (3.24).

$$j(q) = \begin{bmatrix} j_1^{(1)} & 0 & 0 \\ 0 & j_1^{(2)} & 0 \\ j_2^{(1)} & j_2^{(2)} & j_j \end{bmatrix} \tag{3.24}$$

onde:

$$j_j(q) = \begin{bmatrix} 0_3 \\ [Z_{Base}^{Tool}(q(t))]_1 \\ [Z_{Base}^{Tool}(q(t))]_2 \\ [Z_{Base}^{Tool}(q(t))]_3 \end{bmatrix} \tag{3.25}$$

e $[Z_{Base}^{Tool}(q(t))]_i, i = 1, 2, 3$, é a coluna Z da porção de rotação da matriz de transformação da base geral à extremidade geral.

Três plataformas

Nesta situação, há nove graus-de-liberdade, logo, mesmo havendo redundância, espera-se que o robô tenha, no mínimo, seis graus-de-liberdade, permitindo qualquer movimento no espaço tridimensional.

Seguindo procedimento análogo ao usado no caso das duas plataformas, são calculadas as matrizes pré-multiplicadoras (3.26), (3.27) e (3.28).

$$H^{(1)}(q) = \left[\begin{array}{c|c} I_{12} & 0 \\ \hline 0 & \begin{bmatrix} 0 & [\times p_1^3] \\ 0 & 0 \end{bmatrix} \end{array}\right] \tag{3.26}$$

onde p_1^3 é a porção de posição da matriz de transformação formada pela multiplicação das matrizes de transformação da base à extremidade da segunda plataforma e da base à extremidade da terceira plataforma.

$$H^{(2)}(q) = \left[\begin{array}{c|c} I_{12} & 0 \\ \hline 0 & \begin{bmatrix} R_0^1 & [\times p_2^3] \\ 0 & R_0^1 \end{bmatrix} \end{array}\right] \tag{3.27}$$

onde p_2^3 é a porção de posição da matriz de transformação da base à extremidade da terceira plataforma e R_0^1 é a porção de rotação da matriz de transformação que relaciona a base e a extremidade da primeira plataforma.

$$H^{(3)}(q) = \left[\begin{array}{c|c} I_{12} & 0 \\ \hline 0 & \begin{bmatrix} R_0^2 & 0 \\ 0 & R_0^2 \end{bmatrix} \end{array}\right] \tag{3.28}$$

onde R_0^2 é a porção de rotação da matriz de transformação formada pela multiplicação das matrizes de transformação da base à extremidade da segunda e terceira plataformas.

Novamente, neste caso, o passo mínimo é aumentado para $1cm$, já que o robô triplicou de tamanho.

Após a implementação e análise do algoritmo, observou-se que o robô continha uma singularidade na posição inicial escolhida, o que impedia que o método funcionasse corretamente. No entanto, apesar de terem sido testadas diversas configurações iniciais aleatórias, não foi possível computar com sucesso a cinemática para várias plataformas. Também não foi identificadas as razões que causavam este problema.

Capítulo 4

Resultados dos testes e análise

Neste capítulo, os resultados dos testes aplicados ao sistema são apresentados.

Na tentativa de verificar a precisão do controlador de posição, foi implementado um teste bastante simples. Consistia em fazer mover um atuador uma certa distância específica e, a seguir, medí-la, comparando os valores desejados e obtidos. Os resultados são mostrados na Tabela 4.1.

Distância desejada (cm)	**Distância obtida** (cm)	**Erro**
1	1.0	0.0
3	2.9	0.1
5	5.1	0.1
7	7.0	0.0
9	9.0	0.0

Tabela 4.1: Resultados do teste aplicado sobre o controlador de posição.

Como pode ser observado, o controlador de posição tem um erro de $\pm 0,1cm$. O erro da ferramenta de medição utilizada neste é de $\pm 0,05cm$, logo, não se pode definir com precisão o erro do controlador de posição, já que se encontra muito próximo dos limites da ferramenta. O erro total deve ser considerado como o acúmulo dos dois erros, ou seja, igual a $\pm 0,15cm$, e faz-se necessário o uso de uma ferramenta mais precisa para melhor determinação do erro do controlador de posição do sistema, a qual não estava disponível no momento da medição.

Para estimar a força máxima de compressão do robô, foi colocada uma balança embaixo da estrutura do robô e mediu-se o peso exercido pelo robô, quando este se movimentava verticalmente para baixo, na direção da balança (Figura 4.1).

O valor obtido foi $12kgf$. No entanto, esta estimativa não representa a força real máxima do robô, pois, para medí-la seria necessário executar uma teste destrutivo, o qual danificaria a estrutura permanentemente.

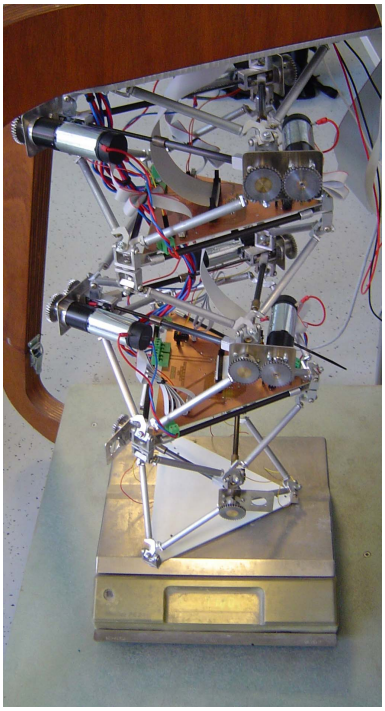


Figura 4.1: Teste para estimar a força de compressão do robô.

Um teste mais relevante é verificar o comportamento do robô quando manipula cargas. Para isto, foi medido o tempo em que o robô levanta uma certa carga (Figura 4.2), por uma distância de $77mm$, e a velocidade foi calculada em seguida (Tabela 4.2). Os valores de velocidade têm apenas dois algarismos significativos, devido ao erro namedição do tempo ser muito elevado. A Figura 4.3 mostra a relação entre a velocidade do robô e o peso carregado.

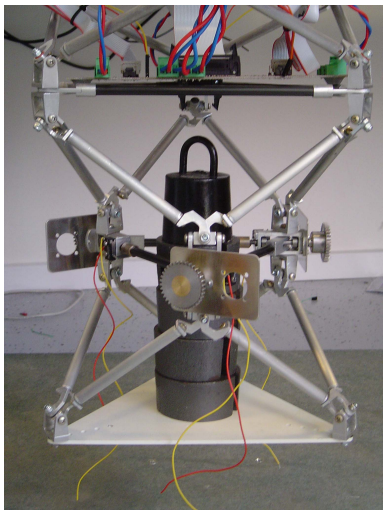
y 

Figura 4.2: Teste para determinar a habilidade do robô para manipular cargas.

Observou-se a corrente que fluía nos motores, para detectar qualquer aumento no torque dos motores. No entanto, nenhuma mudança relevante foi registrada.

Peso (g)	**Tempo médio** (s)	**Velocidade** (mm/s)
0	20.9	3.7
200	20.9	3.7
500	20.6	3.7
700	20.9	3.7
1000	20.9	3.7
1500	20.9	3.7
2000	20.9	3.7
2500	21.0	3.7
3000	20.6	3.7
3500	21.0	3.7
4000	21.0	3.7
4500	21.0	3.7
5000	20.9	3.7

Tabela 4.2: Resultados do teste da influência do peso da carga sobre a velocidade do robô.

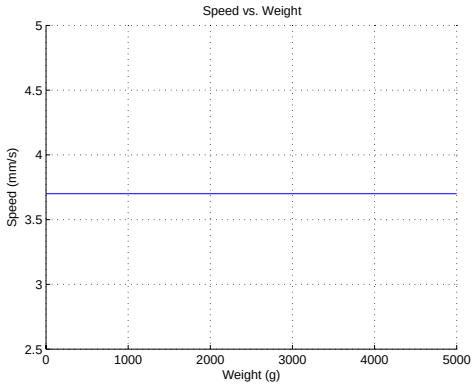


Figura 4.3: Relação entre velocidade e peso.

Verificou-se, ainda, a habilidade do robô em suspender cargas em diferentes direções (Figura 4.4). Novamente, nenhuma mudança na velocidade ou aumento na corrente foram observados.

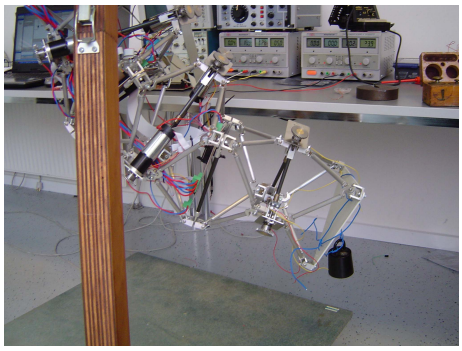
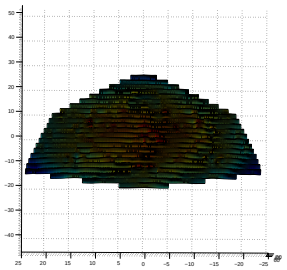
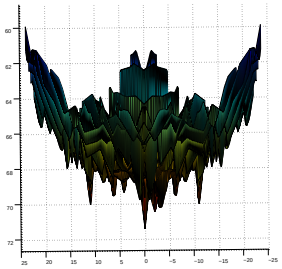


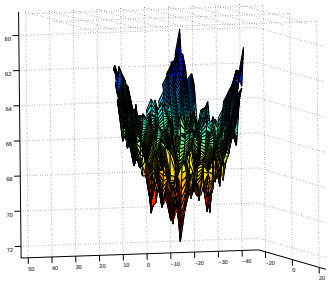
Figura 4.4: Teste para verificar a influência de pesos sobre o movimento do robô, em diferentes direções.



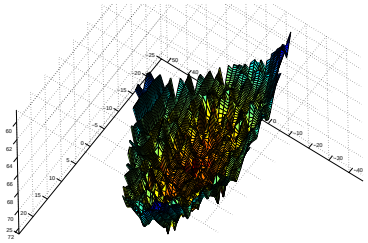
(a) Vista superior.



(b) Vista lateral.



(c) Vista 3D.



(d) Vista 3D superior.

Figura 4.5: Estimativa do "workspace" do robô.

Destes testes, é possível afirmar que a habilidade do robô em manipular cargas é restrita pela resistência da sua estrutura, e não pelas capacidades elétricas dos motores.

Uma importante característica em um robô é o seu "workspace", ou seja, seu alcance espacial. Para determinar o alcance do robô estudado, o algoritmo para a cinemática direta de 3 plataformas foi aplicado. A posição da extremidade geral foi computada para várias configurações distintas, dentro dos limites dos atuadores. A Figura 4.5(a) à Figura 4.5(d) ilustra os resultados obtidos.

Da Figura 4.5(a) e da Figura 4.5(b), pode-se extrair os limites do "workspace". A altura mínima alcançada pela extremidade está em torno de $60cm$. A posição do eixo Y axis varia em $\pm 23cm$ e a do eixo X, de -20 a $26cm$.

Isto é apenas uma aproximação, pois não foi possível computar mais que 700 configurações, devido à baixa capacidade de processamento. Porém, já é possível observar que o suporte atual da estrutura do robô restringe seus movimentos.

Capítulo 5

Conclusões

O controlador de posição implementado é capaz de controlar nove motores numa freqüência de amostragem de aproximadamente $100Hz$, o que permite o domínio total da estrutura paralela. Eventualmente, pode ocorrer alguma instabilidade no sistema, provavelmente relacionada com interferência eletromagnética, entretanto, na maior parte das vezes o sistema se comporta normalmente. Imagina-se que o ruído afeta a comunicação entre o microcontroldador e o FPGA, ou os sinais do encoder são corrompidos, resultando numa posição mal-calculada.

O problema da cinemática direta para três plataformas também foi resolvido com sucesso e o algoritmo foi desenvolvido completamente no Matlab© para implementar a solução obtida.

Já o problema da cinemática inversa também foi estudado e descrito para uma, duas e três plataformas. O método converge, como esperado, para a solução, no caso de uma plataforma. No entanto, quando se tentou extendê-lo para duas plataformas, um problema de redundância foi encontrado e, mesmo superando esta situação, o algoritmo não foi implementado com sucesso. Ademais, a extesão para três plataformas se comportou de maneira semelhante ao de duas, sem êxito.

Observou-se que posição inicial escolhida corresponde a uma configuração singular do robô, contudo, tentativas de configurações aleatórias não resolveram a situação. Faz-se necessária uma análise mais profunda da questão, para compreender a causa do problema.

A partir dos testes aplicados, verificou-se que o robô possui um alcance espacial bastante largo e boa habilidade para manipular pesos, sem perdas notáveis. Porém, estes resultados são preliminares e mais experimentos na estrutura do robô são requeridos, para uma melhor avaliação.

A melhoria mais necessária no sistema é um novo projeto e desenho das PCB, aplicando técnicas de redução de ruído, e a construção das mesmas utilizando uma tecnologia mais confiável, o que deve resolver a maior parte dos problemas de instabilidade observados.

Além da calibração e melhorias no modelo da cinemática, e aperfeiçoamento dos lay-

outs das PCB, outras sugestões para pesquisas futuras no robô incluem:

- executar mais experimentos na estrutura para determinar completamente suas capacidades;
- implementar uma interface de usuário, por exemplo, usando um controle remoto;
- construir um novo suporte para o robô, o qual permita todos os moviemntos possíveis; e
- definir de forma mais precisa a "workspace" do robô.

Referências Bibliográficas

[1] BRUYNINCKX, H. et al., The robotics *WEBook*, http://www.roble.info/,Open content, under WEBook License, 2005, accessed on May 23, 2007.

[2] WIKIPEDIA, Stewart platform, http://en.wikipedia.org/wiki/Stewart_platform, accessed on May 24, 2007.

[3] Ingersoll Milling Machine Company, *Octahedral Hexapod Design Promises Enhanced Machine Performance*, 2000, Octahedral Hexapod Machine Development Program.

[4] Hexapods, http://www.hexapods.net/, accessed on May 24, 2007.

[5] VORNDRAN, S., ParallelMic (2002), accessed on May 24, 2007.

[6] PETERSEN, H. G., Easy and general kinematics for parallel manipulators, in *Proceedings of the IASTED Internacional Conference*, pages 317–032, 2000.

[7] STEMME, O. et al., *Principles and Properties of Highly Dynamic DC Miniature Models*, Maxon Motor, 1994.

[8] UNIVERSITY, G. S., Dc motor operation, http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/motdc.html, 1997, accessed on May 10, 2007.

[9] MIU, D. K., *Mechatronics: Electromechanics and Contromechanics*, chapter 5, Springer-Verlag.

[10] LIBRARIES, C. M. U., Dc motor speed modeling, http://www.library.cmu.edu/ctms/ctms/examples/motor/motor.htm, accessed on May 4, 2007.

[11] RASHID, M. H., *Power Electronics - Circuits, Devices and Applications*, Prentice-Hall, USA, 2nd edition, 1993.

[12] WIKIPEDIA, Pulse-width modulation, http://en.wikipedia.org/wiki/Pulse-width_modulation, accessed on May 2, 2007.

[13] MICROMOUSEINFO.COM, Hints, ideas, inspiration for mice builders, http://www.micromouseinfo.com/introduction/dcmotors.html, accessed on May 2, 2007.

[14] TECH-ETCH, Encoder disc, http://www.tech-etch.com/photoetch/photogallery/EncoderDisc.html, accessed on May 2, 2007.

[15] WIKIPEDIA, Pid controller, http://en.wikipedia.org/wiki/PID_controller, accessed on May 2, 2007.

[16] ISERMANN, R., *Digital Control Systems*, Springer-Verlag, Berlin, 2nd edition, 1989.

[17] WESCOTT, T., Pid without a phd, http://www.embedded.com/2000/0010/0010feat3.htm, accessed on May 13, 2007.

[18] WAKERLY, J. F., *Digital Design - Principles and Practices*, Pearson Prentice Hall, New Jersey, 4th edition, 2005.

[19] Atmel, *ATmega 128 datasheet*, 2006.

[20] CRAIG, J. J., *Introduction to robotics - mechanics and control*, Pearson Prentice Hall, New Jersey, 3rd edition, 2005.

APÊNDICES

Apêndice A

Esquemáticos e layouts das PCB

A.1 Control board

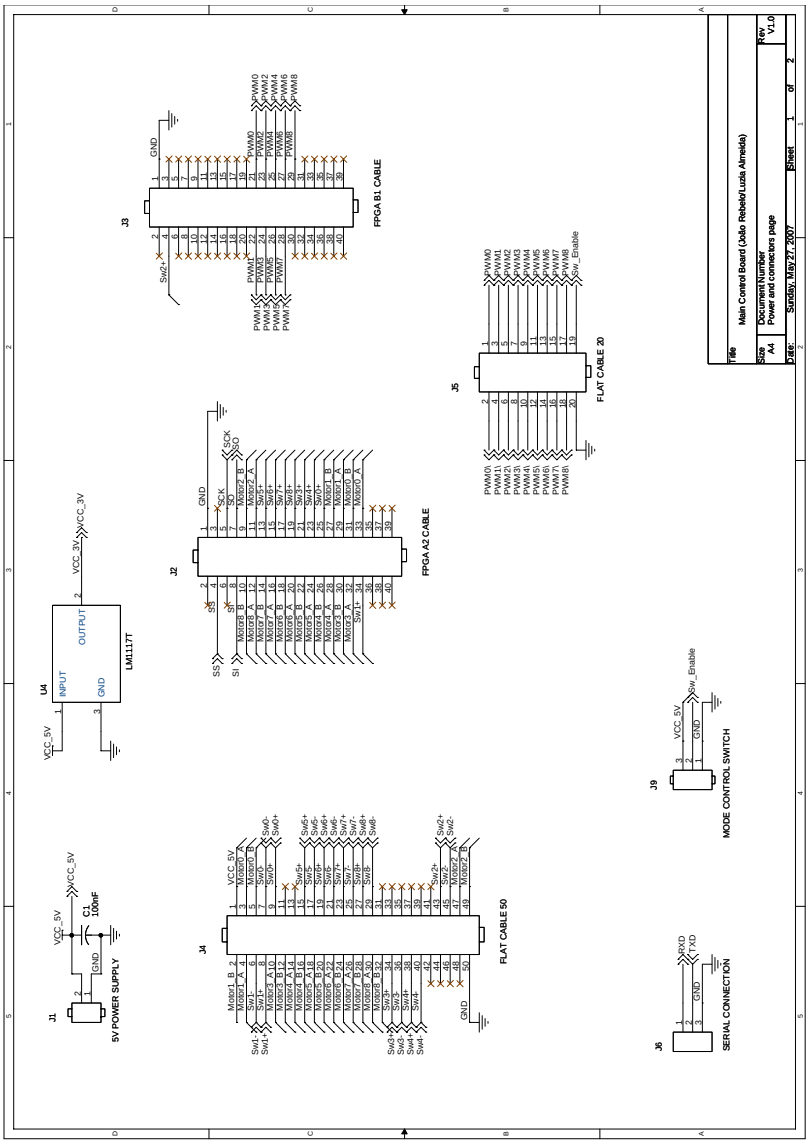


Figura A.1: Página 1 do esquemático da placa de controle.

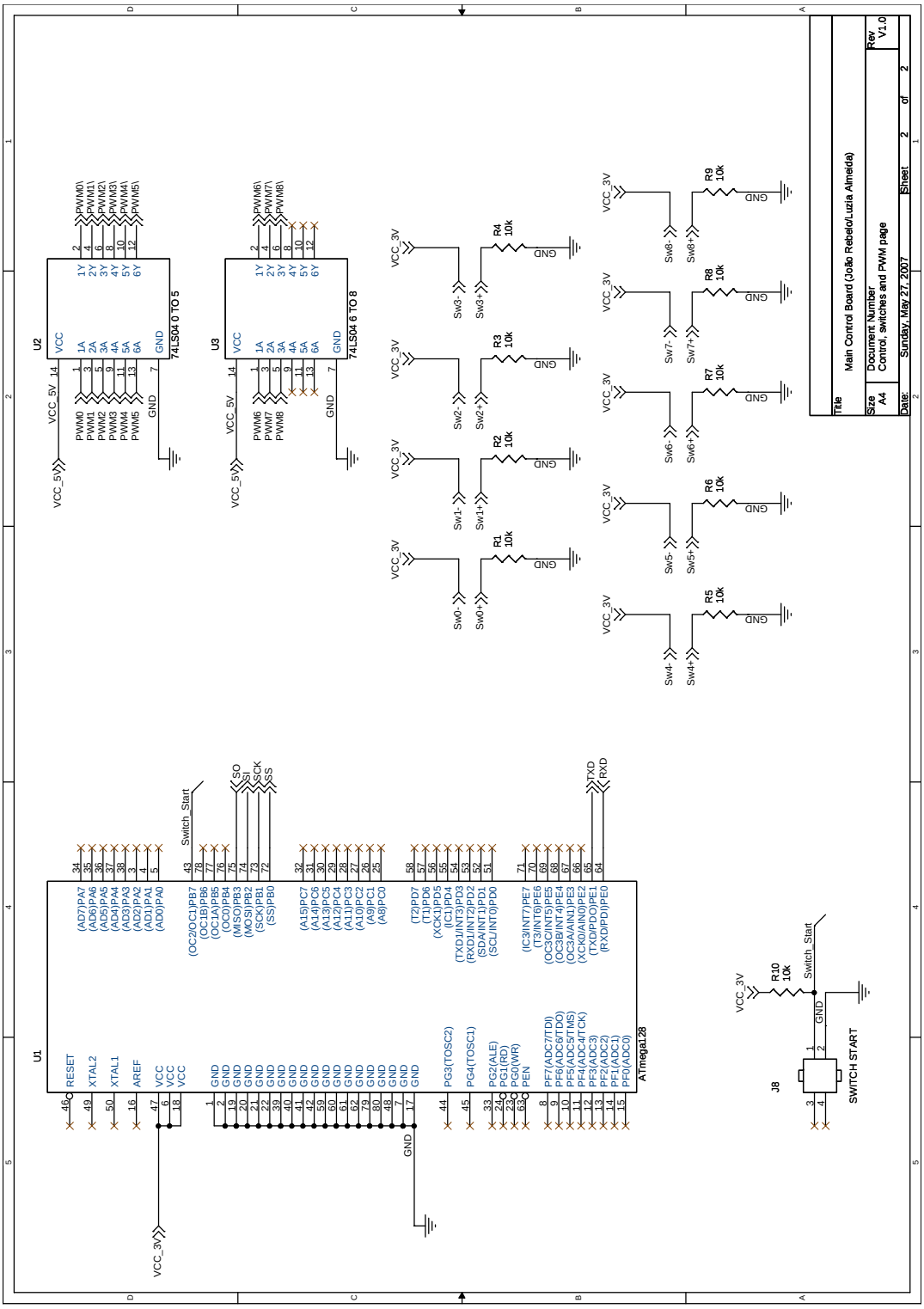


Figura A.2: Página 2 do esquemático da placa de controle.

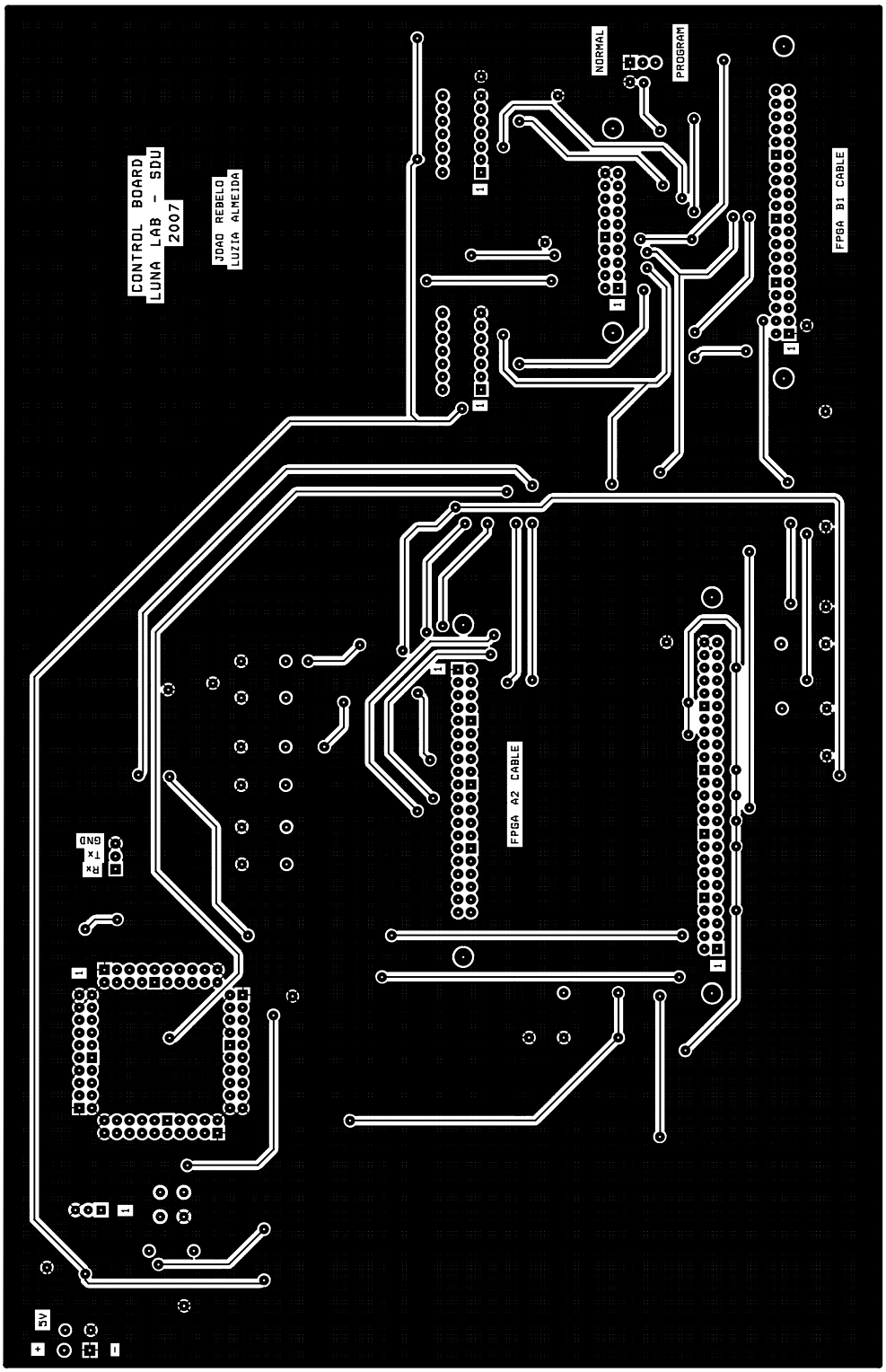


Figura A.3: Camada superior do esquemático da placa de controle (fora de escala).

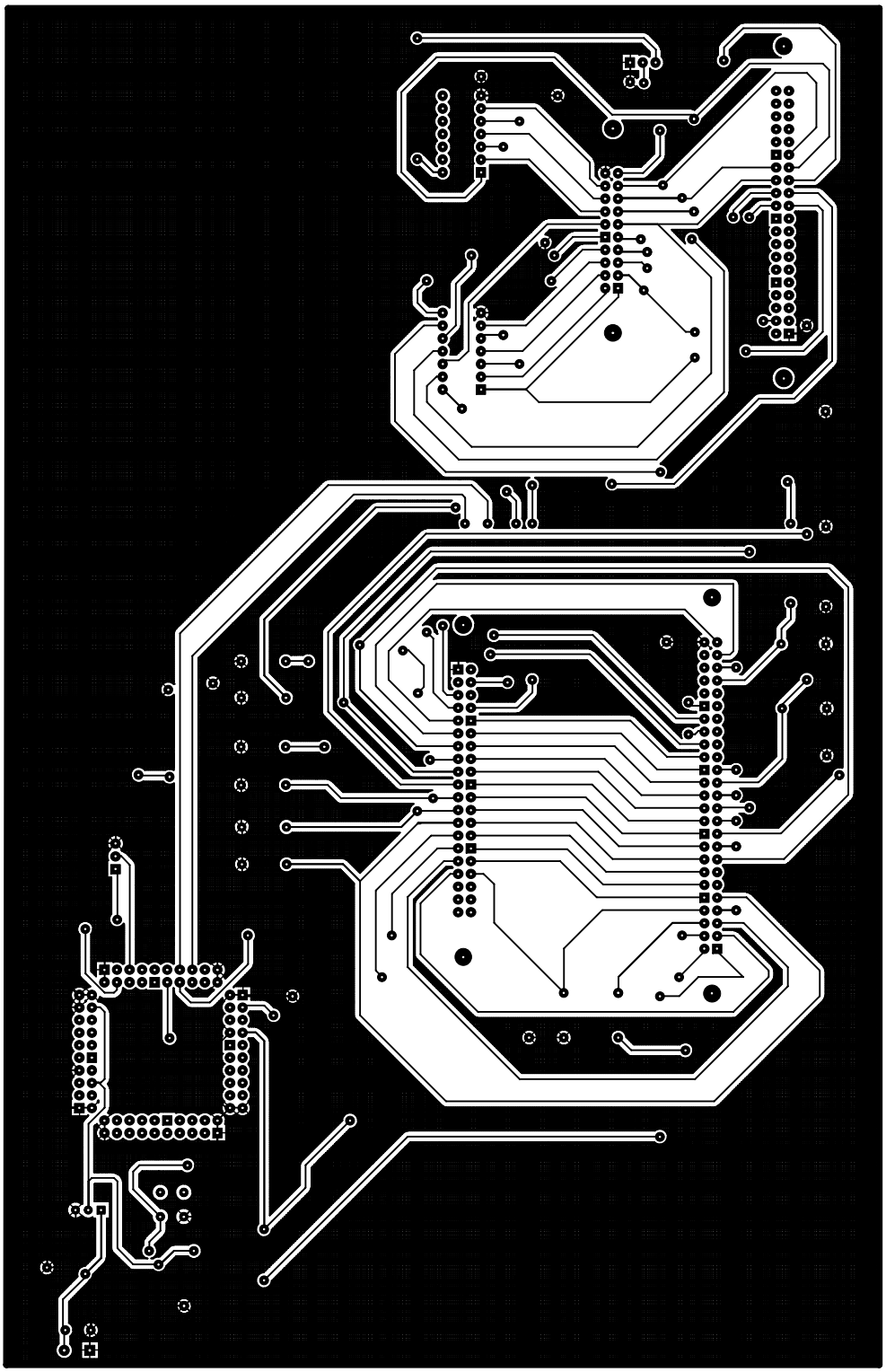


Figura A.4: Camada inferior do esquemático da placa de controle (fora de escala).

A.2 Motor drivers board

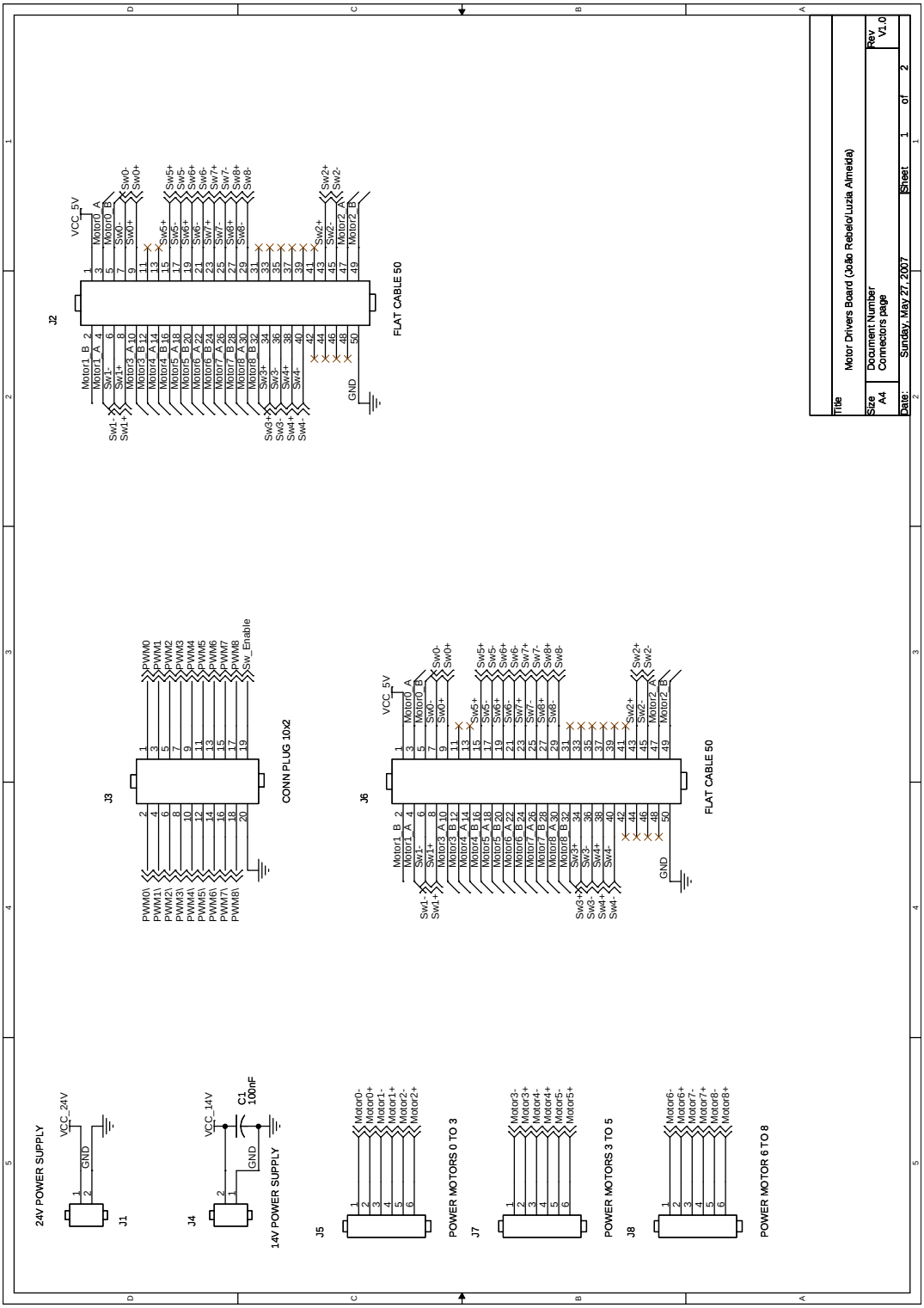


Figura A.5: Página 1 do esquemático da placa dos drivers dos motores.

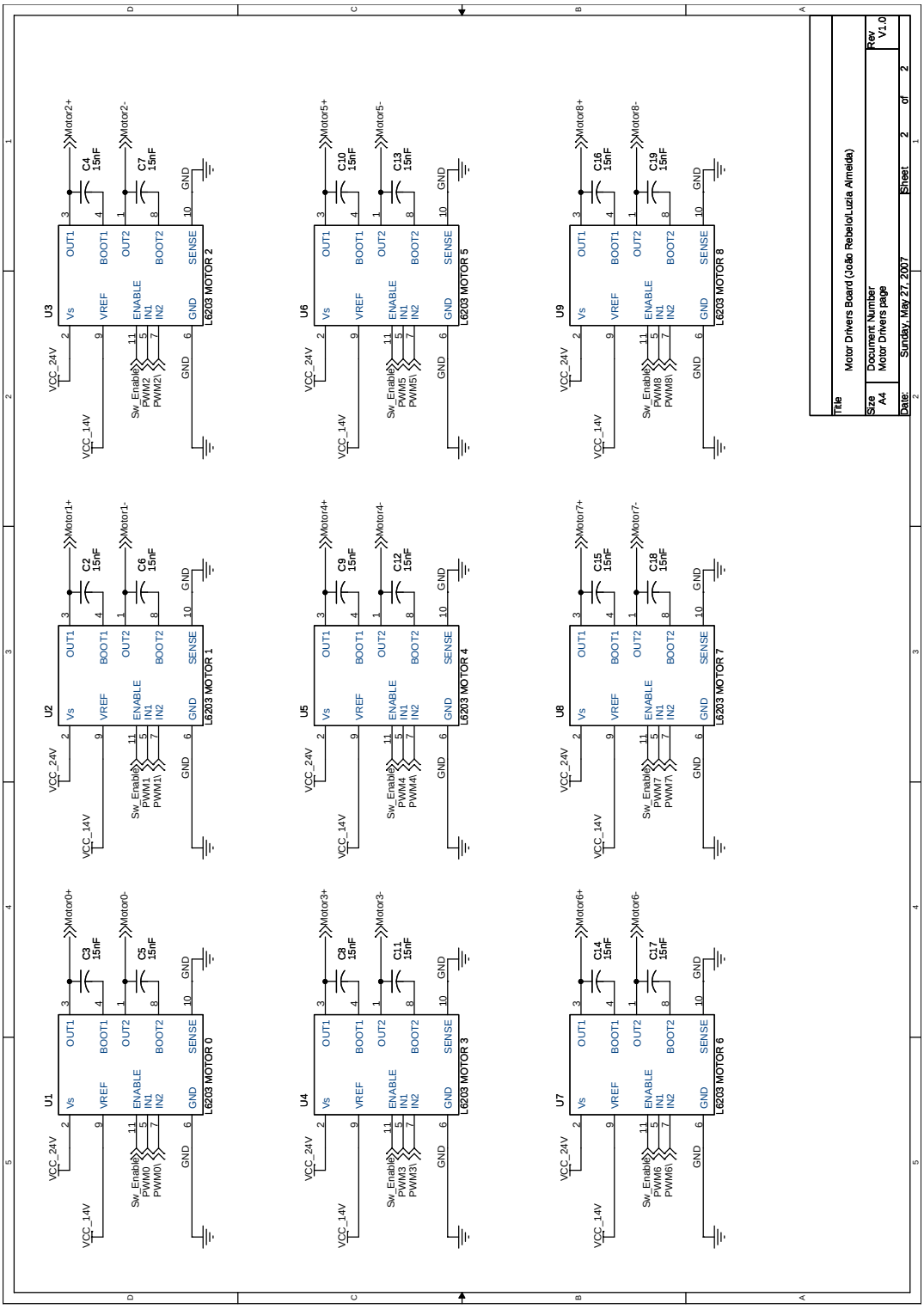


Figura A.6: Página 2 do esquemático da placa dos drivers dos motores.

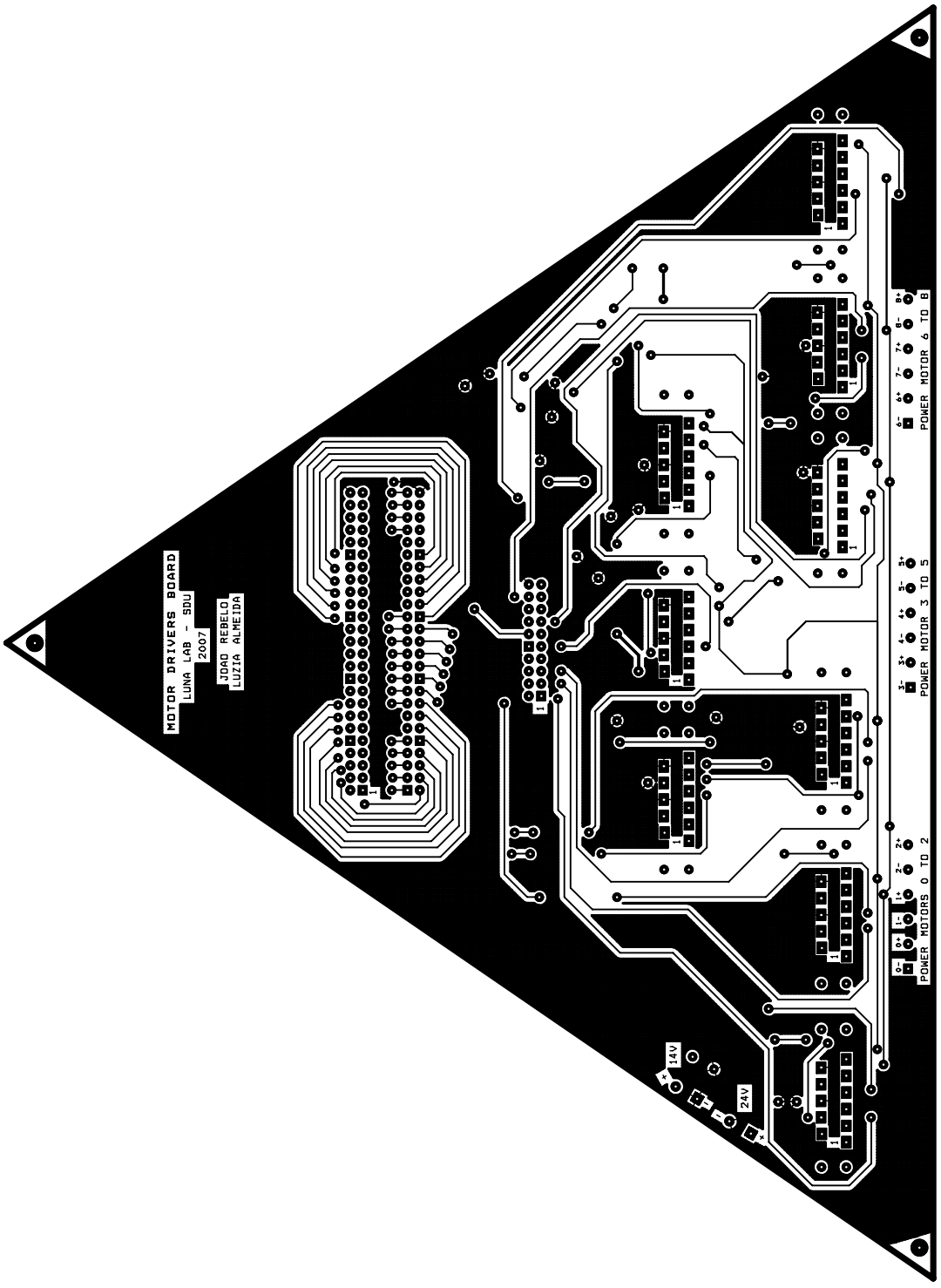


Figura A.7: Camada superior do esquemático da placa dos drivers dos motores (fora de escala).

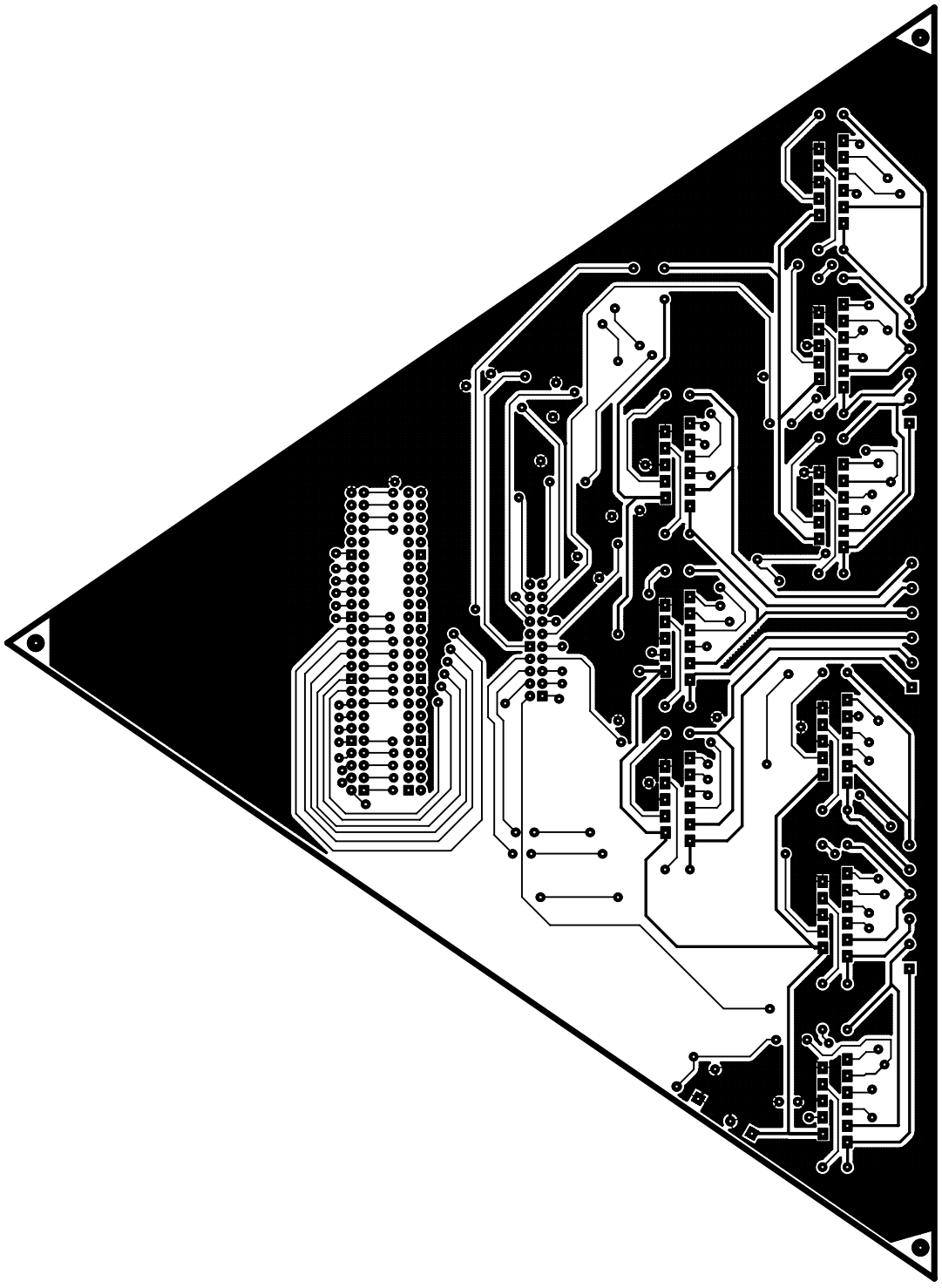


Figura A.8: Camada inferior do esquemático da placa dos drivers dos motores (fora de escala).

A.3 Motors 0 to 2 board

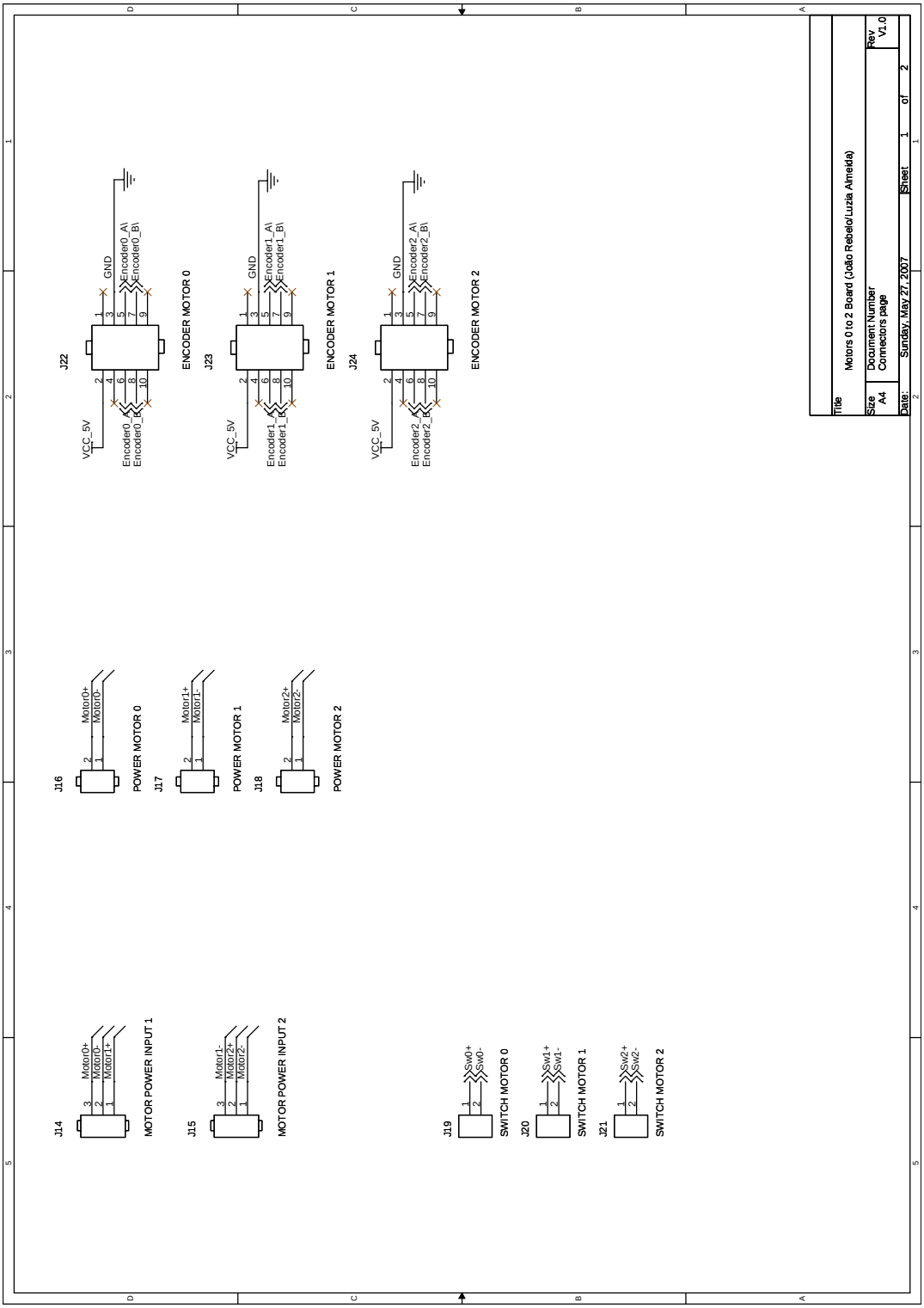


Figura A.9: Página 1 do esquemático da placa dos motores 0 a 2.

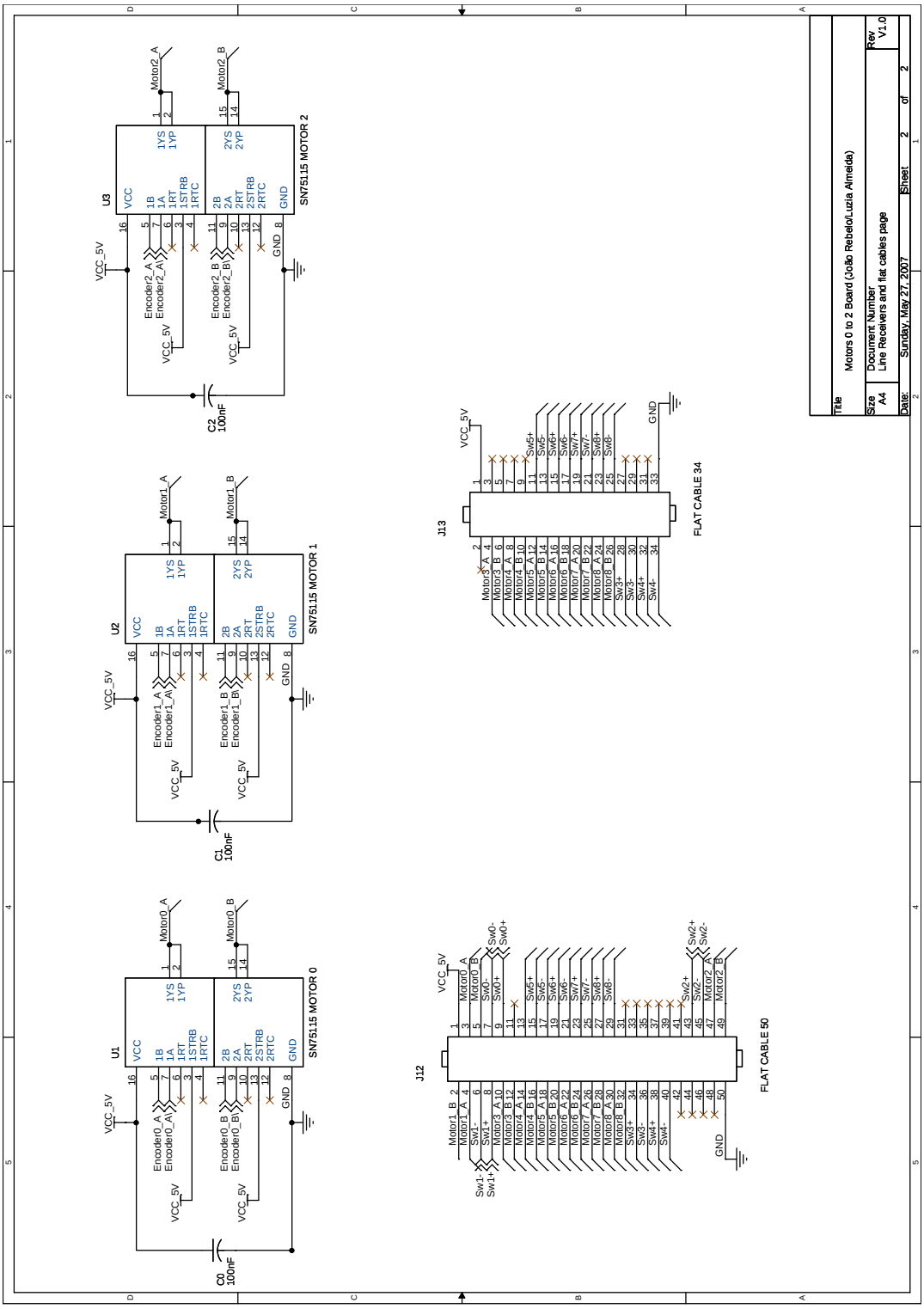


Figura A.10: Página 2 do esquemático da placa dos motores 0 a 2.

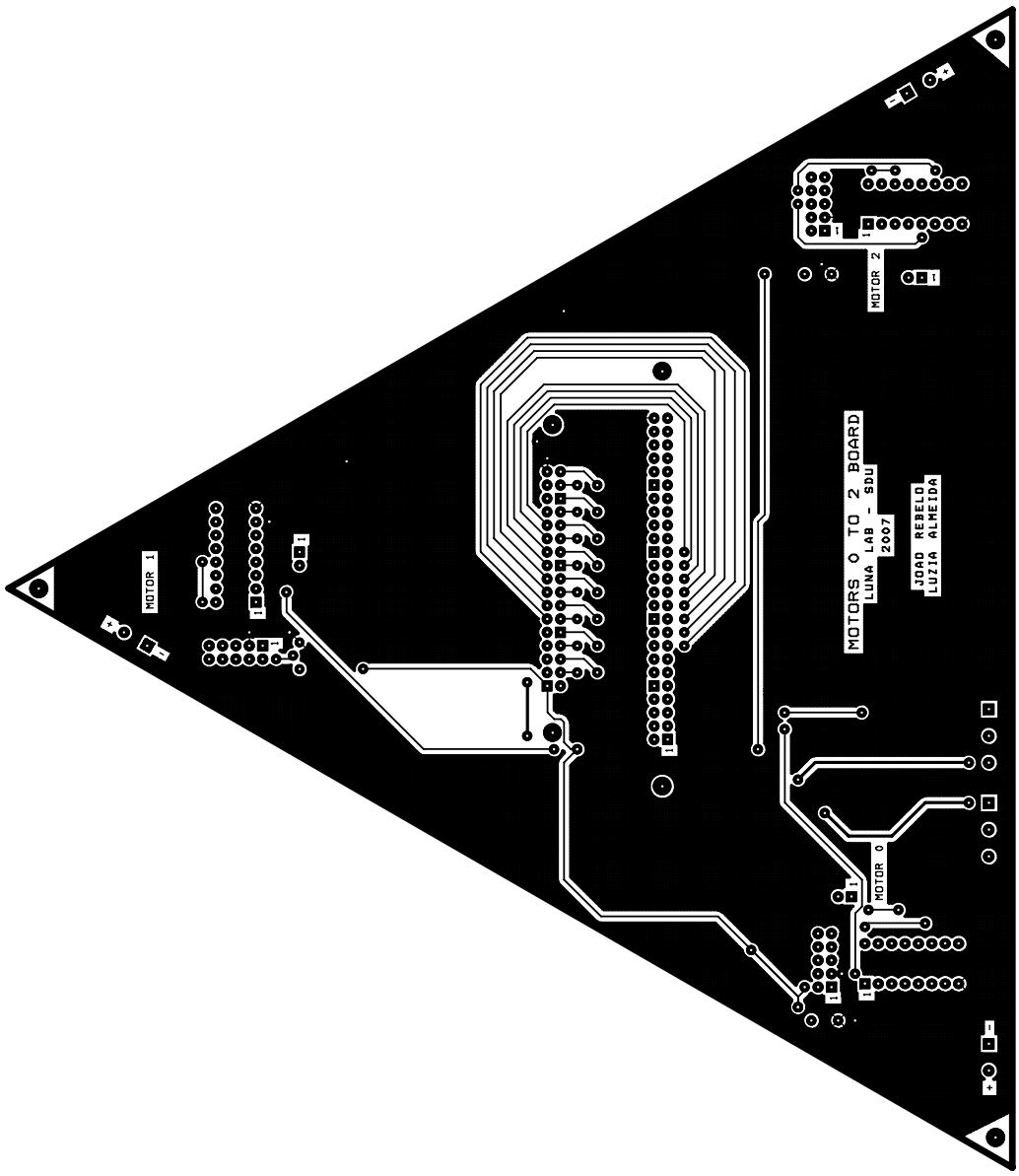


Figura A.11: Camada superior do esquemático da placa dos motores 0 a 2 (fora de escala).

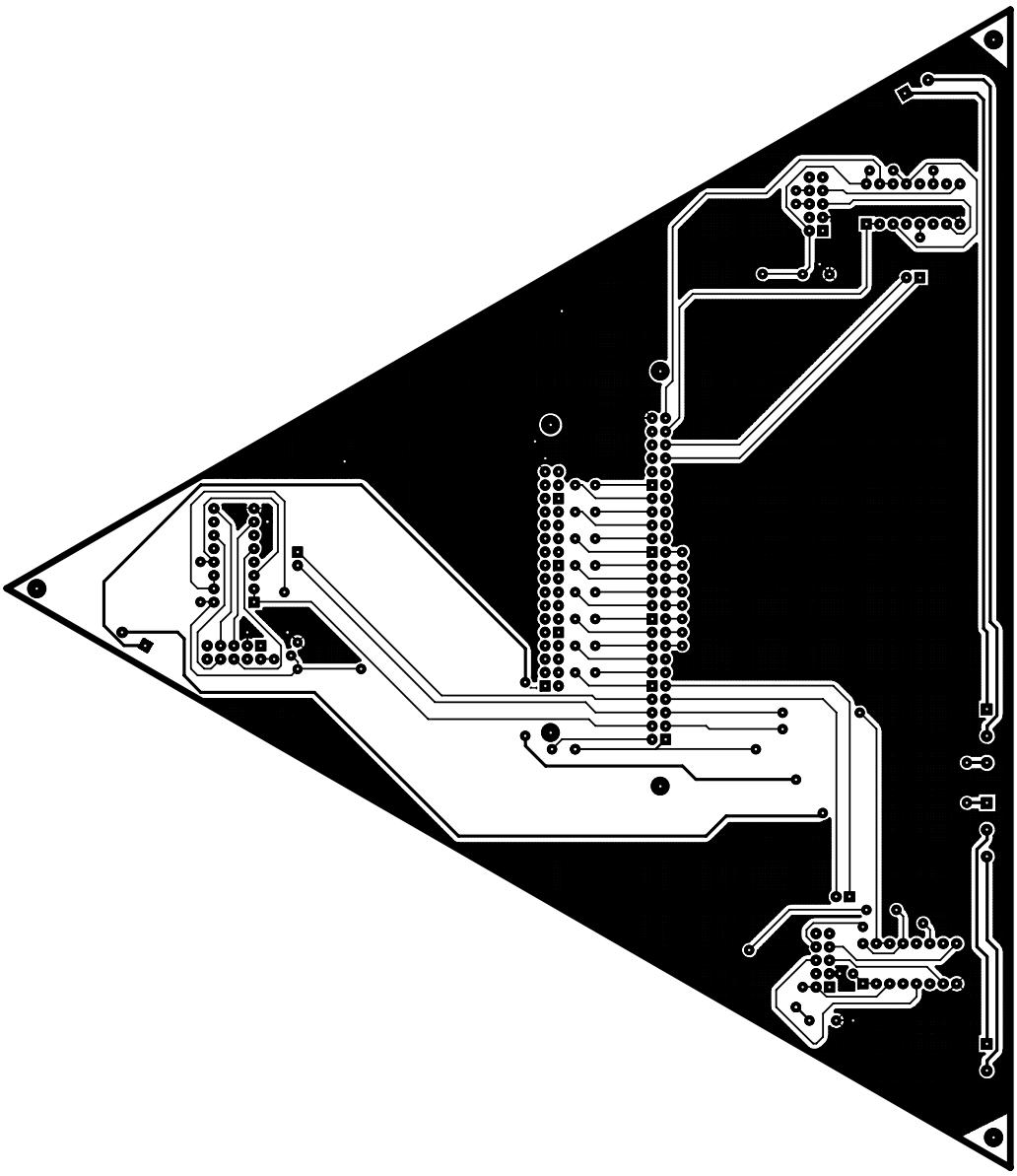


Figura A.12: Camada inferior do esquemático da placa dos motores 0 a 2 (fora de escala).

A.4 Motors 3 to 5 board

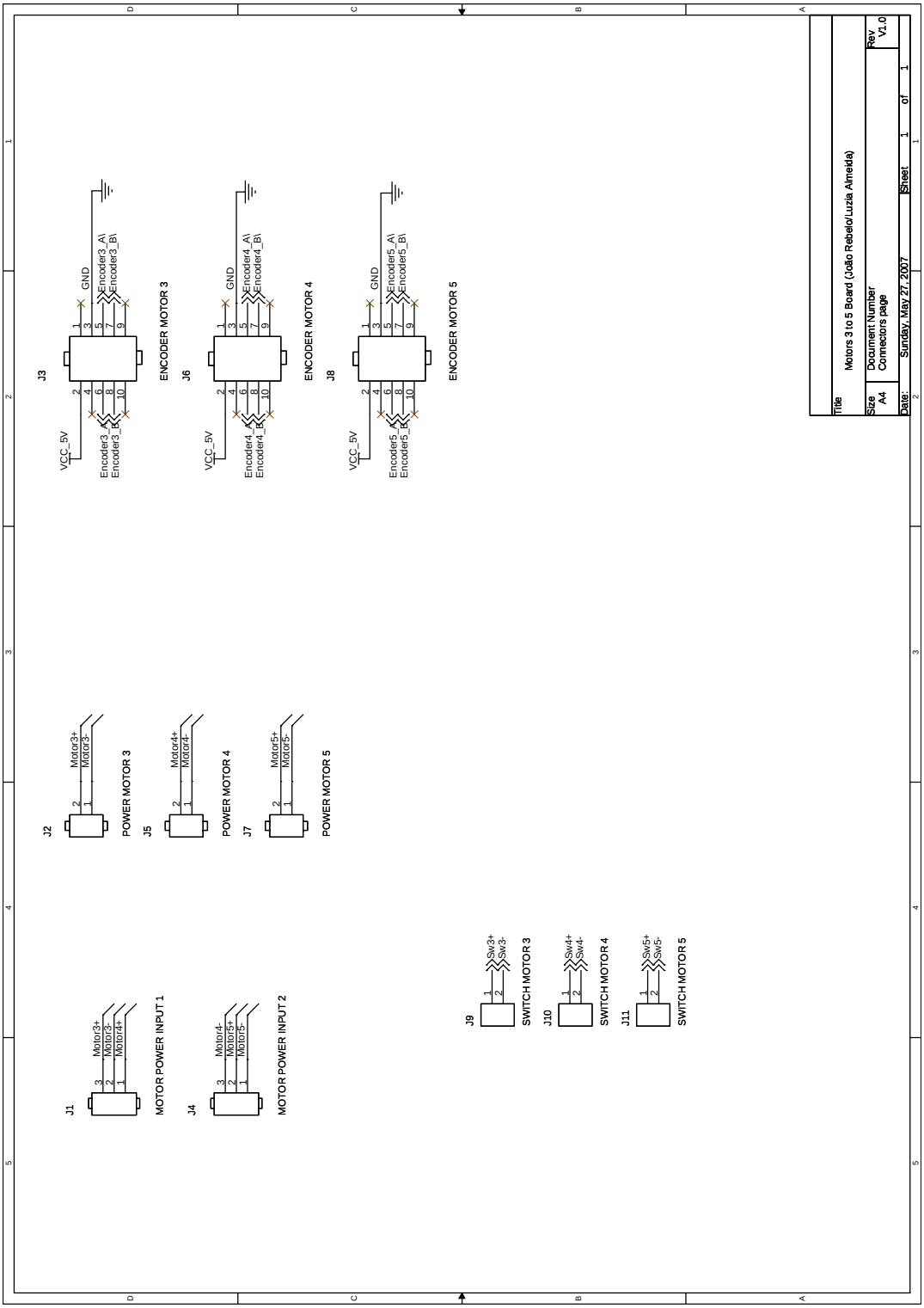


Figura A.13: Página 1 do esquemático da placa dos motores 3 a 5.

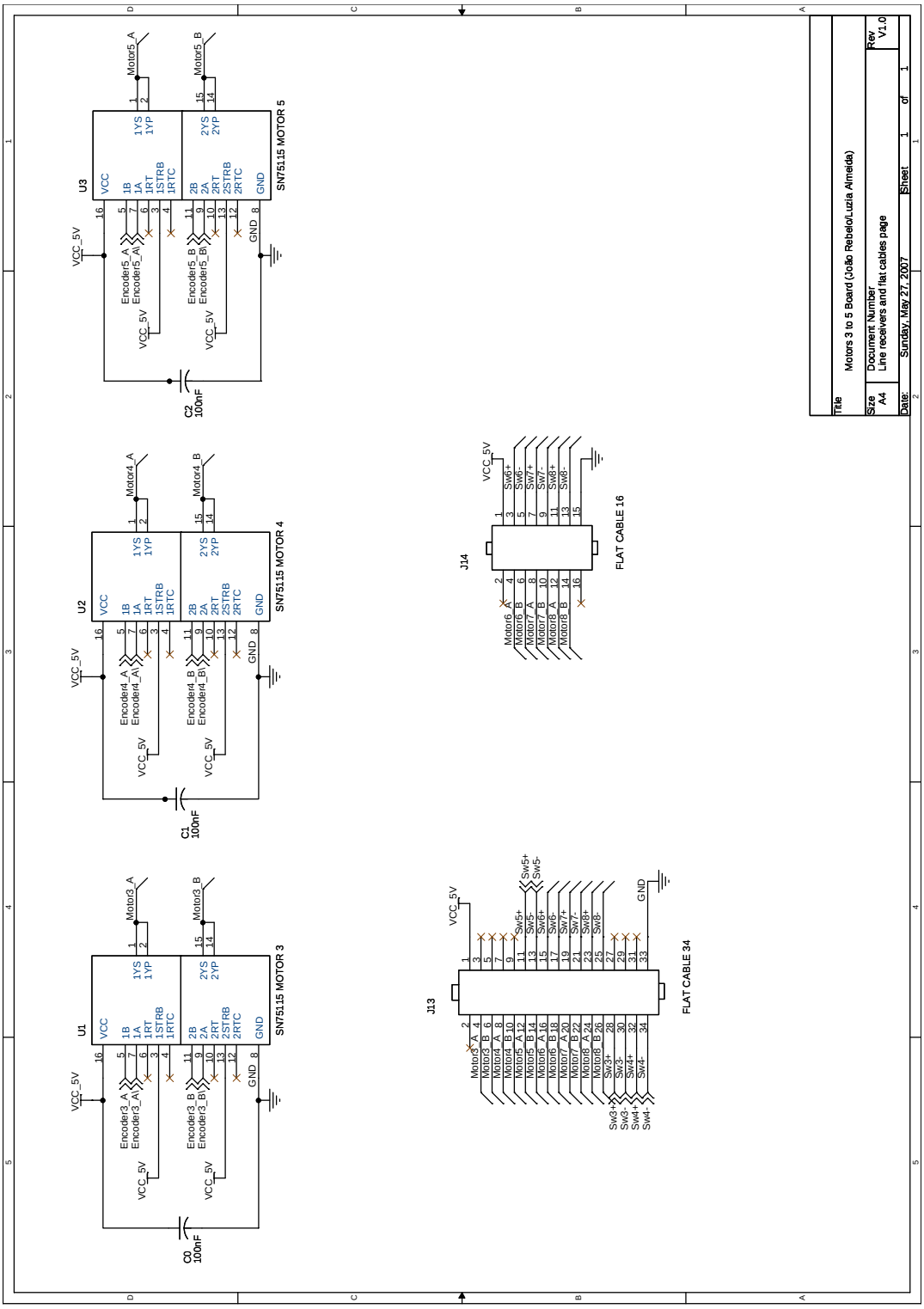


Figura A.14: Página 2 do esquemático da placa dos motores 3 a 5.

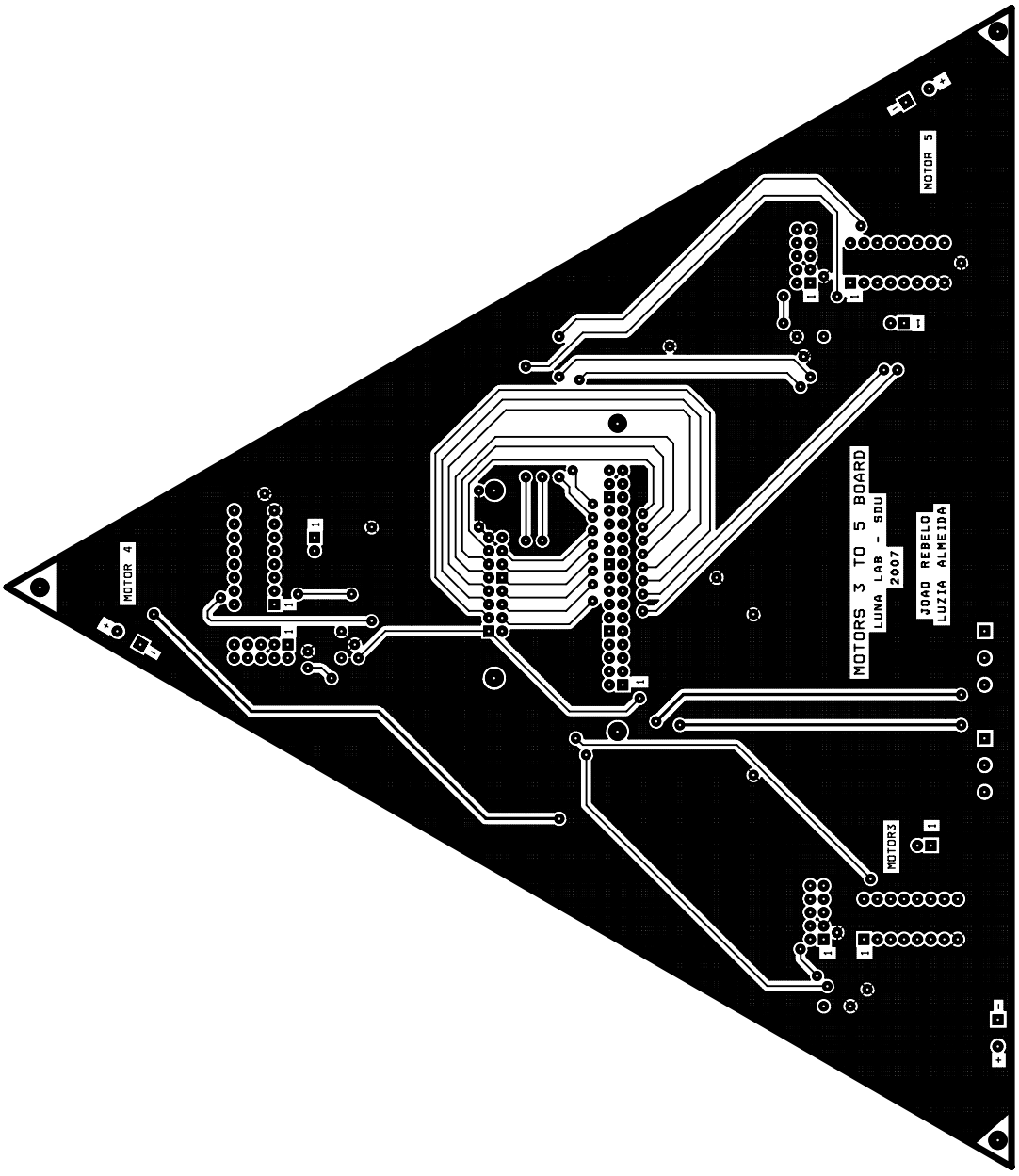


Figura A.15: Camada superior do esquemático da placa dos motores 3 a 5 (fora de escala).

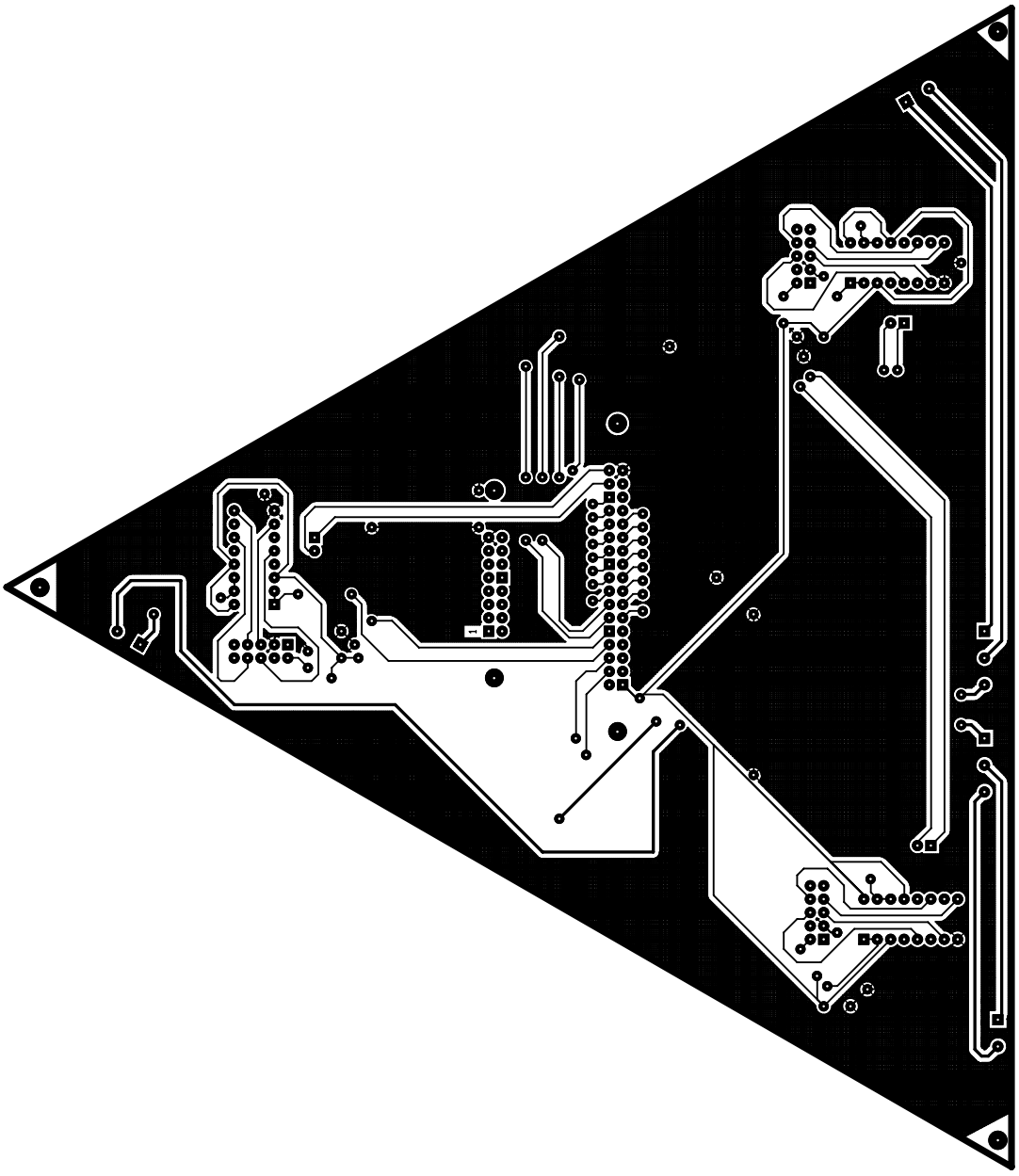


Figura A.16: Camada inferior do esquemático da placa dos motores 3 a 5 (fora de escala).

A.5 Motors 6 to 8 board

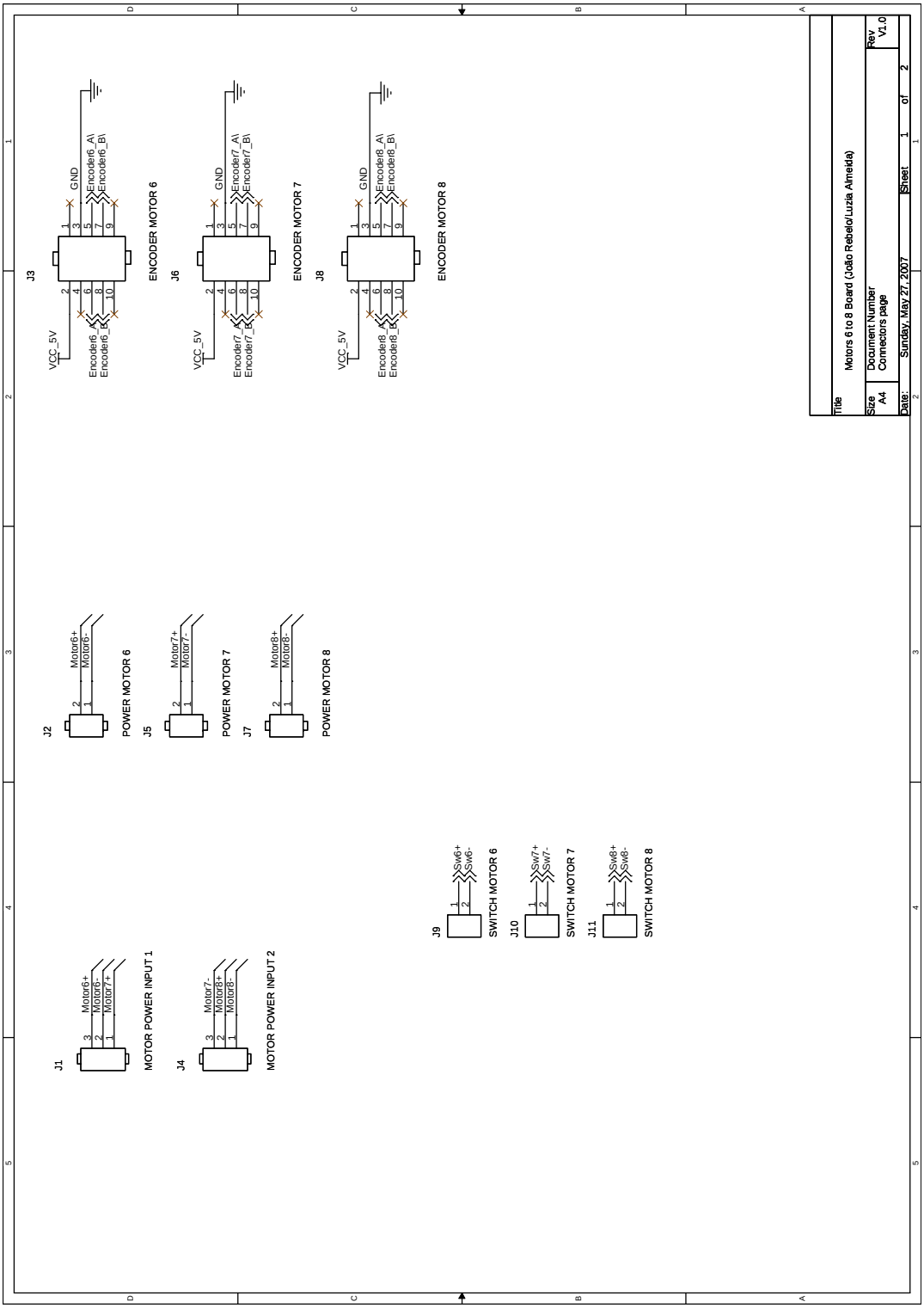


Figura A.17: Página 1 do esquemático da placa dos motores 6 a 8.

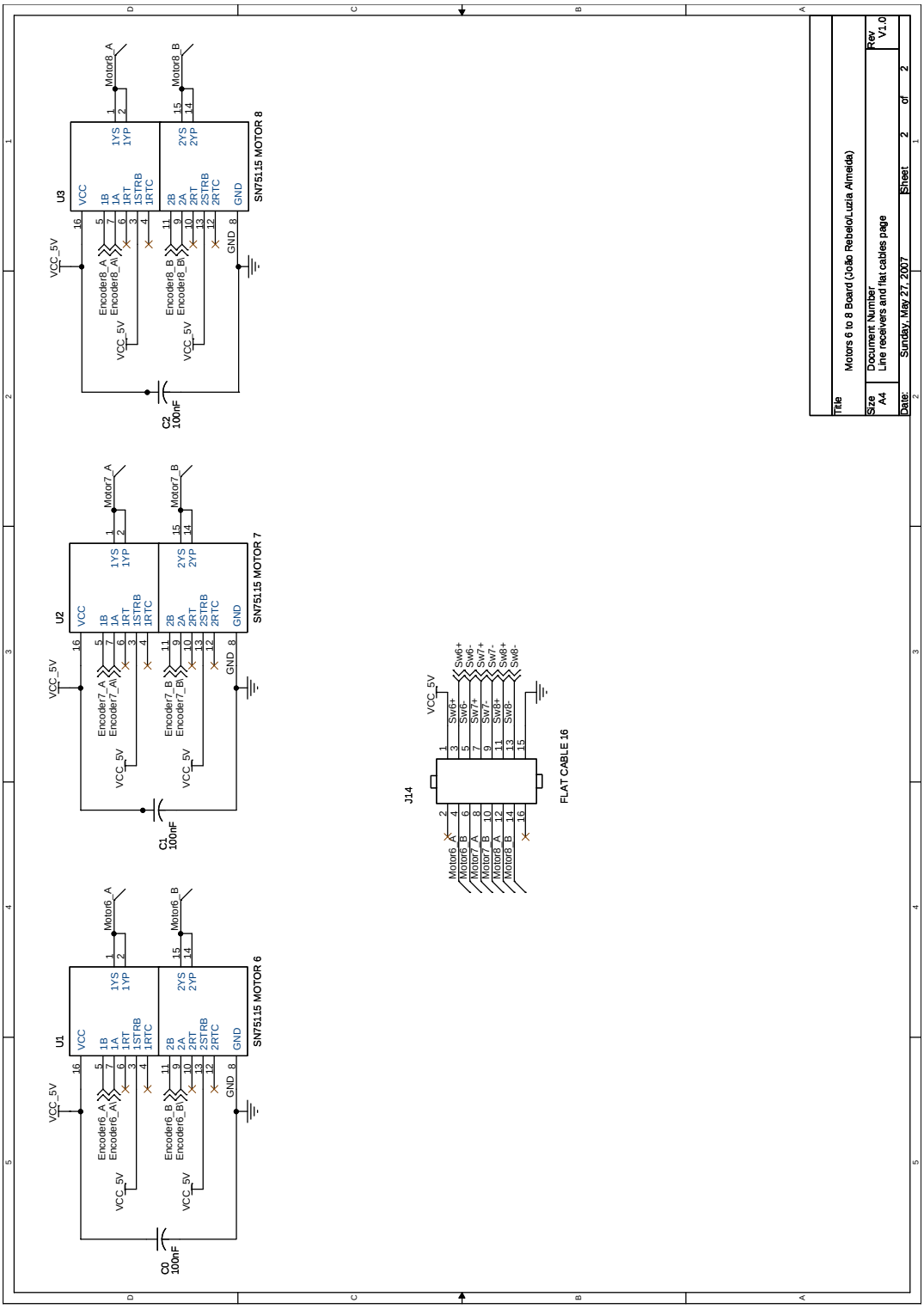


Figura A.18: Página 2 do esquemático da placa dos motores 6 a 8.

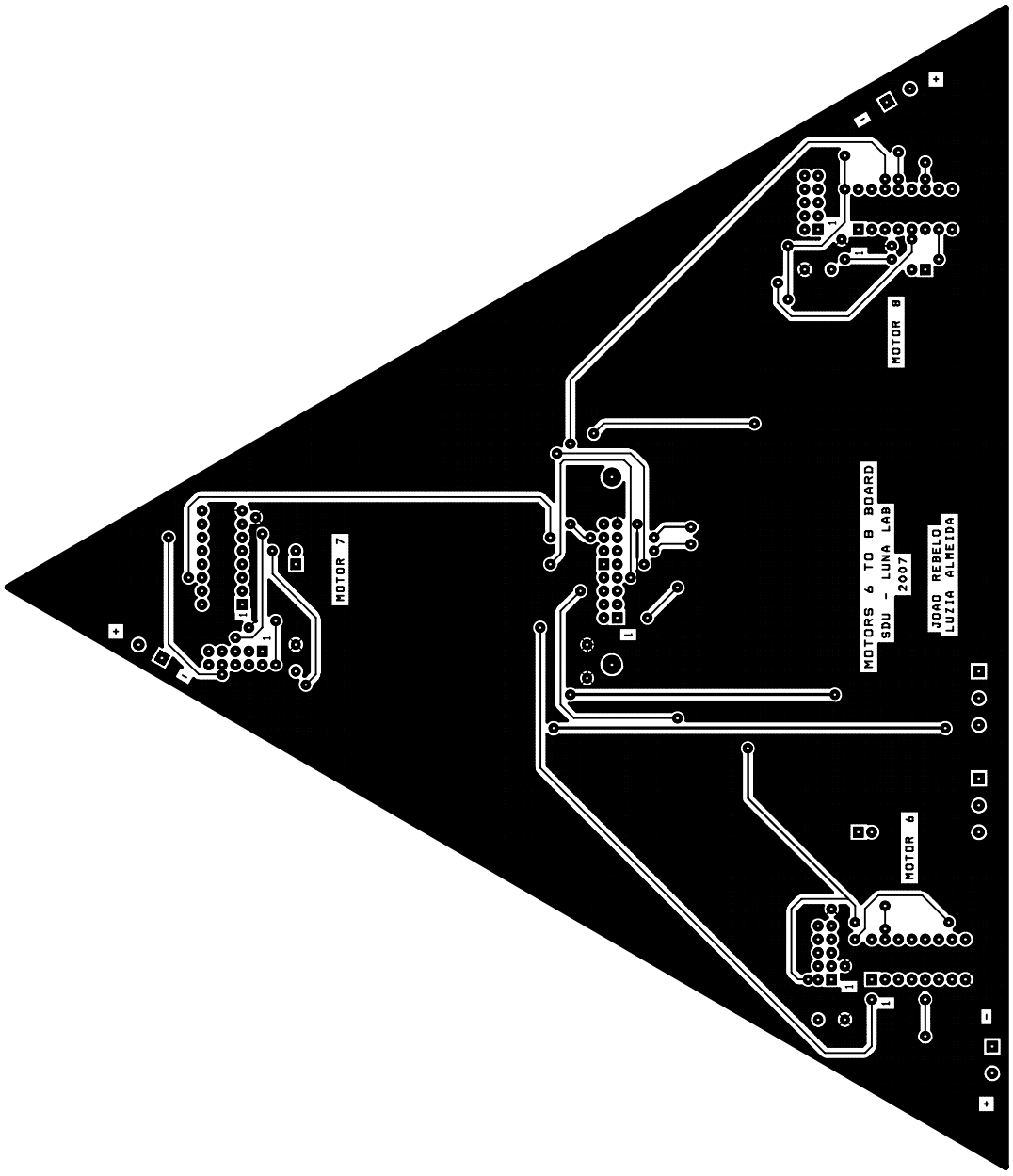


Figura A.19: Camada superior do esquemático da placa dos motores 6 a 8 (fora de escala).

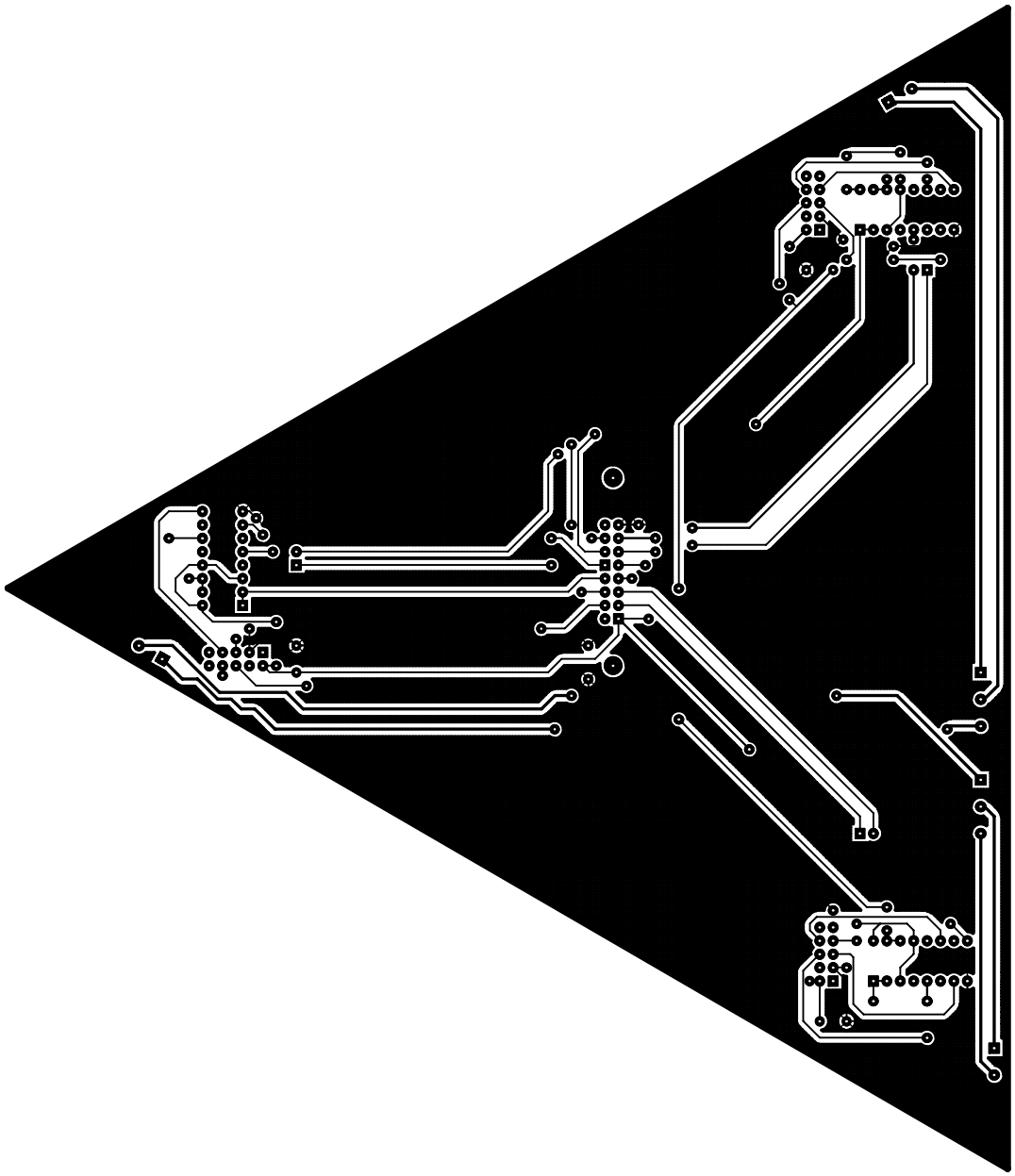


Figura A.20: Camada inferior do esquemático da placa dos motores 6 a 8 (fora de escala).

Apêndice B

FPGA schematic

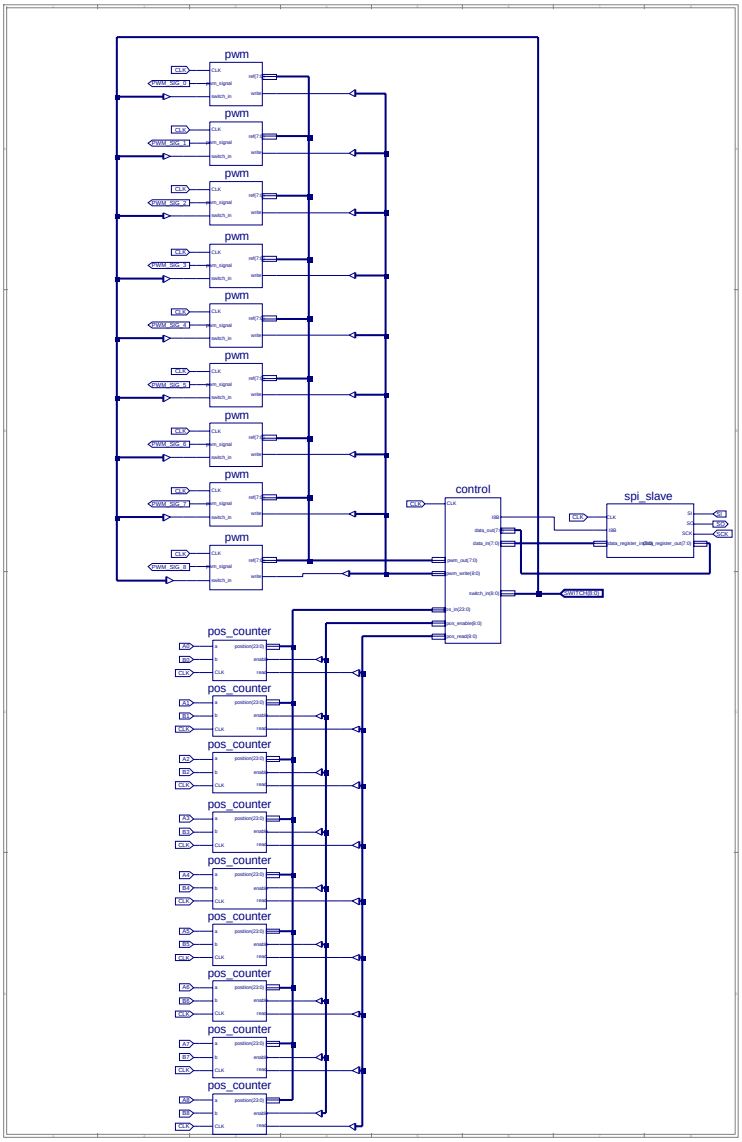


Figura B.1: Esquemático do algoritmo principal do FPGA.

Apêndice C

Conteúdo do CD anexo

O CD em anexo divide-se em 6 pastas principais: *Código fonte ATMega*, *Código fonte FPGA*, *Código fonte Matlab*, *Datasheets*, *Printed Circuit Boards* e *Apresentação*.

As pastas *Código fonte ATMega* e *Código fonte FPGA* contêm os códigos fonte completos implementado no microcontrolador e no dispositivo semicondutor. Já a pasta *Código fonte Matlab* subdivide-se em *Kinematics_1plat*, *Kinematics_2plat* e *Kinematics_3plat*, contendo os códigos fonte implementados no *software* Matlab, correspondendo à cinemática para uma, duas e três plataformas, respectivamente.

A pasta *Datasheets* contém as especificações técnicas dos principais dispositivos utilizados, a saber:

- Inversor 74LS04
- Regulador de tensão 7805
- Microcontrolador ATMega128
- ATMega guide
- Enconder
- Redutor
- Ponte-H L6203
- Regulador de tensão LM1117
- MAX232
- Motor MAXON
- "Line receiver" SN75115n
- Spartan 3 datasheet (FPGA)

A pasta *Printed Circuit Boards* é subdividida em 5 pastas:

- *controlboard*: esquemáticos e layouts da placa de controle
- *mboard1*: esquemáticos e layouts da placa dos motores 0 a 2

- *mboard2*: esquemáticos e layouts da placa dos motores 3 a 5
- *mboard3*: esquemáticos e layouts da placa dos motores 6 a 8
- *powerboard*: esquemáticos e layouts da placa dos drivers dos motores

Por fim, a pasta *Apresentação* contém as apresentações, em formato de slides, realizadas.