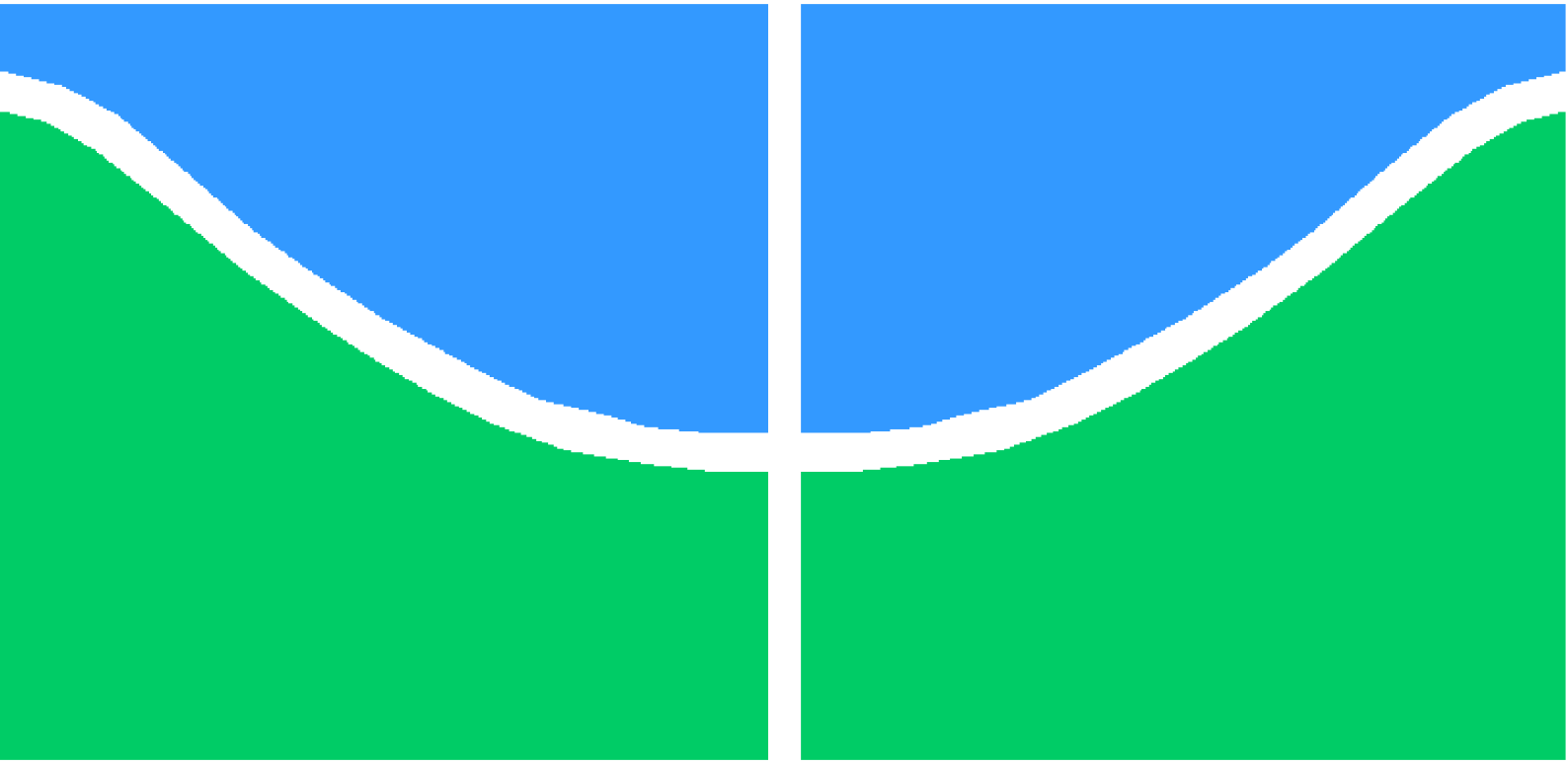


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia Eletrônica

Desenvolvimento de uma interface para acionamento de atuadores e leitura de *encoders* para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo

Autor: Ana Paula Chavier Rodrigues

Orientador: Prof. Me. Renato Coral Sampaio

Brasília, DF

2017

Ana Paula Chavier Rodrigues

Desenvolvimento de uma interface para acionamento de atuadores e leitura de *encoders* para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Me. Renato Coral Sampaio

Brasília, DF

2017

Ana Paula Chavier Rodrigues

Desenvolvimento de uma interface para acionamento de atuadores e leitura de *encoders* para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo/ Ana Paula Chavier Rodrigues. – Brasília, DF, 2017-

69 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Me. Renato Coral Sampaio

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2017.

1. Exoesqueleto. 2. *Fild Programmable Gate Arrays*. I. Prof. Me. Renato Coral Sampaio. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Desenvolvimento de uma interface para acionamento de atuadores e leitura de *encoders* para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo

CDU 02:141:005.6

Ana Paula Chavier Rodrigues

Desenvolvimento de uma interface para acionamento de atuadores e leitura de *encoders* para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Brasília, DF

Prof. Me. Renato Coral Sampaio
Orientador

Prof. Dr. Daniel Maurício Muñoz Arboleda
Convidado 1

Prof. Dr. Gilmar Silva Beserra
Convidado 2

Brasília, DF
2017

Dedico este trabalho à minha mãe, à minha irmã Julia e aos meus avós por todo o amor e apoio que me fortalecem a cada dia da minha vida.

Agradecimentos

Agradeço primeiramente a Deus por me permitir chegar até aqui e por estar comigo em todos os momentos.

Agradeço à minha mãe pelo apoio incondicional em todos os momentos.

Agradeço à minha irmã Julia por todos os sorrisos e pelo carinho que fazem os meus dias serem muito melhores.

Agradeço aos meus avós, tias, tios, primos e primas que sempre me apoiaram e estiveram comigo em cada etapa da minha vida.

Agradeço aos meus professores, em especial ao professor Renato Coral Sampaio pela orientação e por estar sempre disponível para auxiliar na elaboração deste trabalho.

"The most beautiful thing we can experience is the mysterious.
It is the source of all true art and science."
(Albert Einstein)

Resumo

Atualmente, um projeto que visa a construção de um exoesqueleto destinado a membros inferiores está sendo coordenado por professores da Universidade de Brasília. Este equipamento será destinado a hemiplégicos, com o objetivo de auxiliar na locomoção destes indivíduos, proporcionando assim maior independência em atividades cotidianas exercidas pelos mesmos. O exoesqueleto projetado possuirá dois servo motores alocados na altura do joelho e da cintura do usuário, os quais serão controlados por uma plataforma *System on Chip* que contém um *Field Programmable Gate Array* - FPGA integrado com um processador baseado em arquitetura ARM. Estes motores serão acoplados a sensores de posição, os quais serão lidos pela mesma plataforma. Neste trabalho, será implementado um sistema de interface que possibilite o acionamento de motores e a leitura de sensores de posição, *encoders*, por meio de uma plataforma *System on Chip* do modelo Zybo, produzida pela empresa Xilinx. Este sistema será posteriormente utilizado como uma interface para a validação de um controlador de posição que está sendo desenvolvido por um dos membros do projeto citado.

Palavras-chaves: *encoder*, exoesqueleto, *field programmable gate array*, FPGA, hemiplegia, servo motor, Zybo.

Abstract

Currently, a project aimed at building an exoskeleton for lower limbs is being coordinated by professors from the University of Brasilia. This equipment will be destined to hemiplegic, with the objective of assisting in the locomotion of these individuals, thus providing greater independence in daily activities exercised by them. The designed exoskeleton will have two servo motors allocated at the knee and waist of the user, which will be controlled by a System on Chip platform containing a Field Programmable Gate Array (FPGA) integrated with a processor based in ARM architecture. These engines will be coupled to position sensors, which will be read by the same platform. In this work, will be implemented a system that allows the activation of motors and the reading of position sensors, encoders, through of a System on Chip (SoC) platform of the Zybo model, produced by Xilinx. This system will later be used as an interface for the validation of a position controller being developed by other member of the cited project.

Key-words: encoder, exoskeleton, field programmable gate array, FPGA, hemiplegia, servo motor, Zybo

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADC	*Analog to Digital Converter*
AMBA	*Advanced Microcontroller Bus Architecture*
ASIC	*Application-Specific Integrated Circuit*
AVC	Acidente Vascular Cerebral
AVE	Acidente Vascular Encefálico
AXI	*Advanced eXtensible Interface*
BRAM	*Block Random Access Memory*
CLB	*Configurable Logic Block*
DC	*Direct Current*
DLL	*Delay-Locked Loop*
DMA	*Direct Memory Access*
DSP	*Digital Signal Processing*
FSL	*Fast Simplex Link*
FF	*Flip Flop*
FPGA	*Field Programmable Gate Array*
IBGE	Instituto Brasileiro de Geografia e Estatística
IMU	*Inertial Measurement Unit*
IOB	*In/Out Block*
IP	*Intellectual Property*
LED	*Light-Emitting Diode*
PLL	*Phase-Locked Loop*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*

RTOS	*Real-Time Operating System*
SO	Sistema Operacional
SoC	*System on Chip*
UART	*Universal Asynchronous Receiver/Transmitter*
VHDL	*Very high-speed integrated circuit Hardware Description Language*

Lista de símbolos

θ	Letra grega theta
ω	Letra grega omega
Δ	Letra grega Delta

Sumário

1 Introdução

No ano 2000, o censo realizado pelo IBGE apontou que cerca de 1.000.000 de brasileiros eram deficientes físicos. O censo realizado em 2010 pelo mesmo órgão mostrou que este número aumentou para aproximadamente 13,36 milhões. Dentre os argumentos utilizados para justificar este aumento encontra-se o fato de que a população envelheceu: segundo o censo demográfico de 2010 do IBGE, a expectativa de vida dos brasileiros aumentou 5,57 anos com relação ao ano de 1991, passando a ser igual a 72,57 anos.

Dentre as várias classificações de deficiência física existentes encontra-se a hemiplegia, utilizada para definir a paralisia que acomete um hemisfério do corpo humano. A hemiplegia em adultos geralmente ocorre como uma sequela decorrente de um Acidente Vascular Cerebral (AVC) ou de Acidente Vascular Encefálico (AVE) (MASSOCO; LUCINIO; SANTOS, 2013). A recuperação da capacidade de caminhar é um dos principais objetivos durante a reabilitação de um indivíduo que sofreu AVC, pois este é um fator importante para torná-lo independente na realização de tarefas cotidianas (NUNEN et al., 2015).

Dados mostram que pelo menos 70% dos hemiplégicos recuperam a capacidade de andar (CONDIE; CAMPBELL; MARTINA, 2003). Entretanto, grande parte desses indivíduos necessita de algum mecanismo de auxílio à marcha, que geralmente são bengalas ou andadores combinados ou não à uma órtese alocada na perna afetada com o objetivo de auxiliar no suporte do peso corporal e evitar a hiperextensão do joelho (HSU; MICHAEL; FISK, 2008). Estudos de cinemática mostram que apesar de bengalas e andadores aumentarem o equilíbrio e a mobilidade durante a marcha, a utilização contínua destes equipamentos pode causar sobrecarga aos membros superiores do indivíduo (KAKURAI; AKAI, 1996). Além disso, indivíduos que fazem uso constante de bengalas e andadores podem apresentar uma marcha mais assimétrica e lenta quando comparada à de indivíduos saudáveis.

Atualmente, o avanço de pesquisas no âmbito de tecnologias assistivas tem impulsionado o desenvolvimento de equipamentos modernos para auxílio à marcha que apresentam algumas vantagens se comparados à bengalas e andadores. O exoesqueleto é um exemplo de equipamento que pode ser utilizado no contexto de tecnologias assistivas tanto para reabilitação, quanto para assistência na locomoção de deficientes físicos (KAWAMOTO et al., 2009). Estes equipamentos, quando utilizados em membros inferiores, promovem um aumento na mobilidade de indivíduos hemiplégicos por meio de extensão de força articular específica inserida através das articulações dos membros paralisados por meio de atuadores (WHITE et al., 2002).

1.1 Descrição do Problema

Levando-se em consideração o contexto apresentado, um projeto que visa a construção de um exoesqueleto para membro inferior destinado a pacientes hemiplégicos está sendo desenvolvido atualmente no laboratório LEIA (Laboratório de Sistemas Embarcados e Aplicações de Circuitos Integrados) da Universidade de Brasília. O principal objetivo do equipamento é auxiliar na locomoção de indivíduos hemiplégicos durante a execução de tarefas para, dessa forma, melhorar a autonomia dos mesmos.

O exoesqueleto possuirá dois conjuntos compostos por servo motor, caixa de redução e sensor de posição (*encoder*). Estes conjuntos serão fixados em uma estrutura mecânica de maneira a imprimir movimento ao joelho e ao quadril do usuário, tornando possível para ele reproduzir alguns movimentos realizados durante a marcha humana. Para tanto, será necessária a implementação de um sistema de controle de posição baseado na análise da marcha humana contextualizada para pacientes hemiplégicos e na intenção de movimento demonstrada pelo usuário do equipamento, que será captada por meio de um botão inserido em uma bengala eletronicamente instrumentada, que transmitirá informações ao controlador central do exoesqueleto através de uma comunicação sem fio. Dessa maneira, bastará ao usuário do equipamento pressionar o botão para que a estrutura mecânica se movimente.

Nesse sentido, este trabalho consiste na implementação de um sistema cujo objetivo é realizar uma interface entre o sistema de controle do exoesqueleto e os conjuntos dinâmicos compostos por atuador e transdutor de posição (*encoder*). Como forma de facilitar a reutilização da interface de controle desenvolvida, um dos principais componentes do sistema é um IP *core*, implementado em linguagem de descrição de *hardware* VHDL (*Very high-speed integrated circuit Hardware Description Language*) e testado em um *System on Chip* (SoC) Zybo, baseado na arquitetura Zynq da Xilinx que integra, entre outros componentes, um FPGA e um processador ARM *dual-core* Cortex-A9. Este módulo realiza o controle de velocidade, início e parada do atuador com base em parâmetros definidos pelo controlador central, além da leitura do *encoder*, para que possa ser realizado um controle de posição em malha fechada.

O sistema de interface estabelece uma comunicação entre o processador e o FPGA, o que permite que sistemas de controle menos complexos sejam embarcados também na camada de *software* do SoC. A transmissão de dados em alta velocidade possibilitada pelo barramento presente na plataforma *on-chip* garante o funcionamento eficaz dessa interface entre *hardware*/*software* e, dessa forma, a vantagem de execução de tarefas em tempo real conferida aos FPGAs pode ser aproveitada mesmo que o controle do exoesqueleto seja realizado em *software*. Completando a arquitetura geral do sistema, ilustrada na Fig. 1, tem-se ainda os circuitos acopladores, necessários para que seja possível realizar o envio e recebimento de dados entre o SoC, os atuadores e os sensores, visto que estes componentes

possuem tensões de funcionamento diferentes.

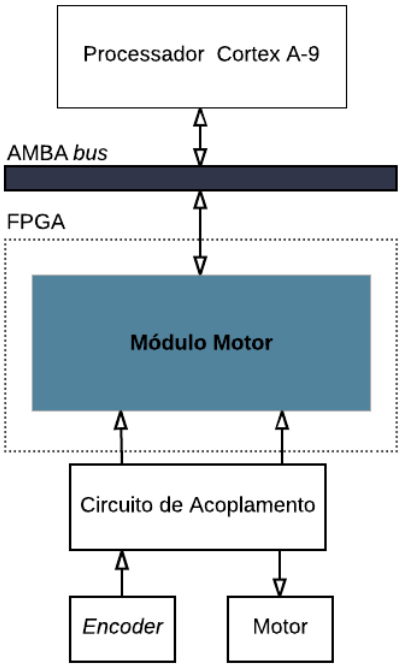


Figura 1 – Visão geral dos principais blocos e conexões do sistema de interface.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo deste trabalho é projetar e desenvolver uma interface de controle embarcada em uma plataforma SoC-FPGA Zybo que possibilite o acionamento de atuadores e a leitura de *encoders* por um sistema de controle de posição de um exoesqueleto destinado a membro inferior.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Embarcar um sistema operacional Linux em uma plataforma SoC-FPGA Zybo;
- Implementar um módulo reutilizável (IP *core*) para controle de servo motores e leitura de *encoders* incrementais de quadratura em linguagem de descrição de *hardware* VHDL;
- Utilizar o barramento AMBA AXI para estabelecer uma comunicação entre o processador ARM e o FPGA existentes na plataforma de desenvolvimento Zybo;

- Desenvolver circuitos acopladores que possibilitem a integração entre o FPGA presente na plataforma Zybo, os atuadores e os *encoders* destinados ao exoesqueleto;

1.3 Metodologia

A abordagem *top-down* adotada possibilitou o desenvolvimento paralelo das etapas que compõem a execução deste trabalho. Após a definição das caracterísitcas do sistema, as atividades foram subdivididas em implementações em *hardware*, implementações em *software* e prototipagem do sistema. Inicialmente, realizou-se a compilação do sistema operacional para embarcá-lo na plataforma Zybo. Posteriormente, foram seguidos os passos de desenvolvimento descritos a seguir.

- Projeto e implementação dos circuitos lógicos para leitura do *encoder* em quadratura e acionamento do servo motor, ambos descritos em VHDL;
- Simulação dos circuitos lógicos para leitura e acionamento com o *sofware* Vivado;
- Montagem e teste dos circuitos acopladores em bancada;
- Teste da integração entre *hardware* e *software* com o *Software Development Kit* (SDK) da Xilinx;
- Validação do sistema de interface utilizando o SDK;
- Customização do *hardware* da plataforma Zybo com a adição do módulo de interface desenvolvido;
- Desenvolvimento das placas de circuito impresso contendo os circuitos acopladores;
- Construção de uma bancada para facilitar os testes do sistema;
- Validação do sistema por meio de aplicações desenvolvidas no Linux embarcado na plataforma Zybo.

1.4 Organização do Trabalho

Este trabalho encontra-se subdividido em seis capítulos:

- Capítulo 1: este capítulo apresenta a contextualização dos temas abordados neste trabalho;
- Capítulo 2: neste capítulo são apresentadas as teorias necessárias para a implementação do sistema proposto;

- Capítulo 3: no terceiro capítulo é apresentada a arquitetura geral do sistema, a descrição técnica dos componentes utilizados para o seu desenvolvimento, além dos procedimentos experimentais realizados para testá-lo;
- Capítulo 4: o Capítulo 4 apresenta as implementações realizadas para o desenvolvimento do sistema;
- Capítulo 5: neste capítulo são apresentados os resultados finais obtidos com os testes do sistema de interface;
- Capítulo 6: capítulo de conclusões, considerações finais e sugestões para trabalhos futuros.

2 Referencial Teórico

2.1 Estado da arte

Atualmente, o desenvolvimento de tecnologias assistivas para auxílio à locomoção de deficientes físicos tem crescido e, neste contexto, o desenvolvimento de exoesqueletos têm se destacado. Estes equipamentos têm sido amplamente desenvolvidos, tanto em um contexto industrial ou militar para proporcionarem um aumento de força a seres humanos, quanto no contexto de tecnologias assistivas, com o objetivo de auxiliarem na reabilitação ou proporcionarem um aumento na mobilidade cotidiana de deficientes físicos (KIM et al., 2015). No âmbito de tecnologias assistivas, um exoesqueleto é um tipo de órtese dinâmica e, como tal, fornece força através de articulações com a finalidade de promover uma extensão dos movimentos articulares, necessário principalmente em pacientes que sofreram lesões cerebrais ou AVC (HSU; MICHAEL; FISK, 2008).

Além de aumentar a mobilidade de deficientes físicos, a utilização de exoesqueletos durante a reabilitação possibilita um aumento na repetição de movimentos de caminhada realizados, o que leva a um retorno mais rápido de resultados nesta fase devido ao treino induzido pelo qual o sistema nervoso do paciente é submetido. Além disso, a utilização de órteses ativas no período da reabilitação permite a obtenção de dados mais precisos e detalhados sobre o que ocorre com o corpo dos indivíduos durante essa fase(HSU; MICHAEL; FISK, 2008). Isto possibilita um melhor estudo de caso e colabora com o avanço de pesquisas nos campos da biomecânica e cinemática da marcha humana, necessárias para o projeto de um exoesqueleto (ARAÚJO, 2010).

Neste contexto, podem ser citados alguns trabalhos importantes relacionados ao desenvolvimento de exoesqueletos destinados a membros inferiores. Hassan et al. (2014) desenvolveu um sistema para análise de marcha de indivíduos hemiplégicos utilizando um exoesqueleto. Para tanto, foram utilizadas unidades inerciais de movimento (IMUs) e sensores de pressão dispostos na perna saudável e em uma bengala. Dessa forma, a bengala instrumentada, além de realizar seu papel natural de proporcionar maior equilíbrio ao indivíduo, auxilia também na estimação da intenção de movimento indicada pelo mesmo por meio de um botão e fornece valores para o controlador central do sistema, os quais auxiliam na análise de marcha. A Figura 2a mostra o sistema desenvolvido por Hassan et al. (2014).

Zhu et al. (2016) desenvolveu uma órtese dinâmica para indivíduos paraplégicos. O exoesqueleto desenvolvido é destinado às duas pernas e quando utilizado juntamente à duas muletas, permite que paraplégicos caminhem. Os comandos de início e parada

dos movimentos que permitem a caminhada são transmitidos pelo usuário por meio de um botão alocado na muleta, que se comunica com o controlador central por meio de uma comunicação sem fio. A Figura 2b mostra o exoesqueleto desenvolvido por Zhu et al. (2016).

Figura 2 – Exoesqueletos de membros inferiores: a. destinado a hemiplégicos desenvolvido por Hassan et al. (2014). b. destinado a paraplégicos desenvolvido por Zhu et al. (2016).

2.2 SoC (*System on Chip*)

Atualmente, existem plataformas de desenvolvimento conhecidas como SoCs (*Systems on Chip*) que, como o próprio nome sugere, oferecem ferramentas para implementação de sistemas de processamento em um único *chip*. Um SoC pode conter recursos como FPGA, microprocessador, memória, *Direct Memory Acess* (DMA), conversor analógico/digital (ADC), além de periféricos (PUTRA et al., 2016). Além de possibilitar um processamento em alta velocidade, o desenvolvimento de sistemas em uma plataforma *on chip* pode resultar em menor volume e consumo de energia, além de uma complexidade de *hardware* reduzida (ZHU; WANG; XU, 2011). A Fig. 3 mostra esquematicamente a estrutura básica de um SoC Zybo da Xilinx, desenvolvido com base na arquitetura Zynq 7000.

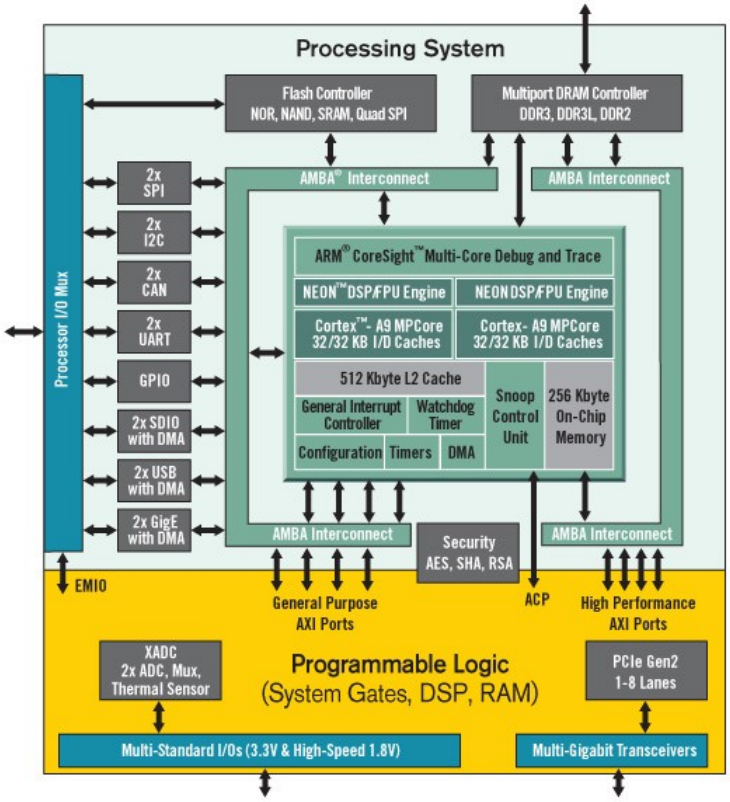


Figura 3 – Estrutura básica de um SoC Zybo. Fonte: (CROCKETT et al., 2014).

A integração entre *hardware* e *software* em um único *chip* possibilitada por um SoC colabora com o desenvolvimento de sistemas mais complexos (TRIMBERGER, 2015), visto que a execução de tarefas que exigem maior capacidade de processamento pode ser realizada na camada de *hardware*, enquanto que a camada de *software* pode ser utilizada para o controle e inicialização dos recursos do FPGA, assim como ocorre no trabalho desenvolvido por PUTRA et al., ou para a execução de tarefas que demandam menor capacidade de processamento. Isto ocorre pois a arquitetura SoC permite que o processador seja tratado como ponto central do sistema, fazendo com que o FPGA, bem como os demais recursos disponíveis se tornem ferramentas para a implementação de sistemas. A Fig. 4 mostra sistematicamente o modelo do sistema de *hardware* de um SoC.

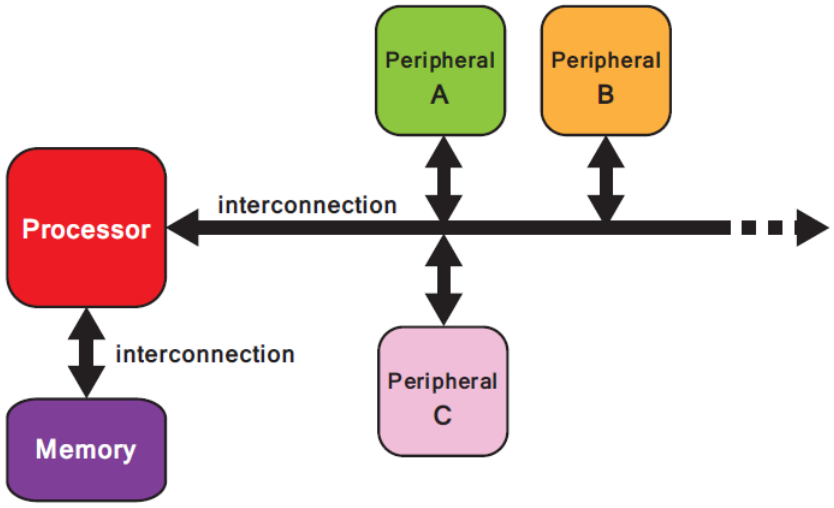


Figura 4 – Modelo do sistema de *hardware* de um SoC. Fonte: (CROCKETT et al., 2014).

Para estabelecer um caminho de dados entre as camadas de *software* e *hardware* são utilizadas interconexões, que podem ser ponto-a-ponto, nas quais cada periférico é conectado de maneira individual ao processador, ou por meio de barramentos, conhecidos também por *buses*. Nesta segunda abordagem, deve ser estabelecido um protocolo de comunicação para que o acesso aos dados pelos periféricos seja sistematizado (CROCKETT et al., 2014).

2.2.1 *Field Programmable Gate Array* (FPGA)

FPGA é um dispositivo semicondutor constituído de uma matriz de blocos lógicos configuráveis (CLBs) unidos por meio de interconexões programáveis (XILINX, 2012b). O roteamento destas interconexões é realizado mediante a compilação de códigos que contém circuitos lógicos detalhados em linguagens de descrição de *hardware*, como VHDL e Verilog. Atualmente, um FPGA pode conter milhares de blocos lógicos, além de outros componentes, como BRAM - bloco de memória *Random Access Memory* (RAM), bloco de processamento de sinais digitais (DSP), *phase-locked loop* (PLL) ou *delay-locked loop* (DLL) para manipulação dos sinais de *clock*, além de blocos de entrada/saída (IOB). A adição destes e de outros componentes em um FPGA colabora para a elaboração de circuitos digitais de alta complexidade. A Fig. 5 mostra a arquitetura básica de um FPGA.

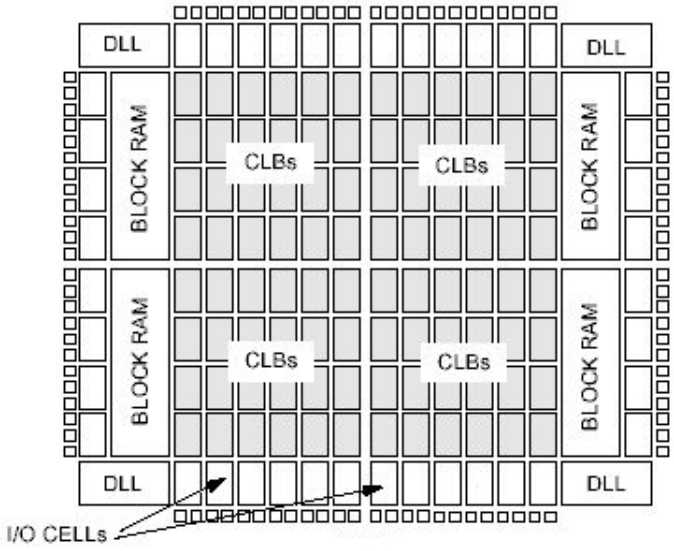


Figura 5 – Estrutura básica de um FPGA. Fonte: (INSTRUCTABLES, 2010).

O FPGA foi criado em meados de 1984 pela Xilinx (NATIONAL INSTRUMENTS, 2016). Antes do advento desta plataforma, os circuitos eletrônicos projetados eram dedicados a uma determinada aplicação e o tempo necessário para o desenvolvimento destes projetos era prolongado, visto que, após produzidos, os *chips* não podiam ser reprogramados para que erros pudessem ser corrigidos ou para que o sistema implementado fosse atualizado. Dessa maneira, uma das principais vantagens apresentada pelo FPGA em relação aos circuitos integrados dedicados a uma aplicação específica (ASICs) é o fato de permitirem reconfigurações e flexibilidade no projeto de circuitos lógicos digitais.

Em um FPGA, cada tarefa é processada em uma seção dedicada do *chip*. Dessa forma, desde que a capacidade de processamento da plataforma não seja ultrapassada, o desempenho de uma aplicação não é afetado com a adição de uma nova tarefa (NATIONAL INSTRUMENTS, 2016). No passado, o FPGA era utilizado em projetos que demandassem menor capacidade de processamento. Entretanto, o fato de atualmente poderem ultrapassar o desempenho de 500MHz, combinado à crescente densidade de CLBs possibilitada pelos avanços tecnológicos na indústria de semicondutores, além da adição de recursos adicionais de processamento e memória em um único *chip* fez com que os FPGAs se tornassem favoráveis para a elaboração de uma grande variedade de projetos (XILINX, 2012a).

2.2.2 Barramentos para comunicação em SoCs

Em um SoC, barramentos são utilizados para estabelecer uma comunicação entre os diversos periféricos e o seu processador. Os periféricos podem ser elementos adicionais de memória, elementos que complementem a atividade de processamento do processador principal, ou *cores* que realizam aplicações específicas.

Exemplos de barramentos comumente utilizados para estabelecer uma comunicação numa plataforma SoC são AMBA, Avalon, *Fast Simplex Link* (FSL), Wishbone, dentre outros. O barramento AMBA, por exemplo, é específico para plataformas que possuem processadores baseados na arquitetura ARM, enquanto que o Avalon é utilizado em plataformas *on Chip* fabricadas pela Altera.

2.2.2.1 Barramento AMBA

O barramento AMBA (*Advanced Microcontroller Bus Architecture*) é um protocolo de comunicação exclusivo aos processadores ARM; este padrão de comunicação realiza a interconexão de blocos em um sistema *on-chip*. Com este protocolo é possível realizar a conexão de periféricos ao processador central do sistema, o que propicia ao mesmo o controle de *cores* e demais periféricos.

Existem quatro tipos de barramentos AMBA: AMBA AHB, AMBA ASB, AMBA APB e AXI. Os barramentos AMBA AHB e AMBA ASB são aptos para sustentar a comunicação entre dispositivos de memória tanto externos quanto *on-chip*, assim como DMAs e processadores ARM de alta performance. Por outro lado, o barramento AMBA APB é ideal para estabelecer uma comunicação entre um processador e diversos dispositivos periféricos (ARM, 1999).

O barramento *Advanced eXtensible Interface* (AXI) é caracterizado por possibilitar a implementação de sistemas com alta frequência de processamento sem a necessidade de utilizar pontes complexas, além de possuir canais de leitura e escrita de dados distintos, o que promove um baixo custo de DMA (ARM, 2011). Os sinais básicos necessários para utilização do protocolo AXI são:

- Endereço para leitura;
- Dado de leitura;
- Endereço para escrita;
- Dado para escrita;
- Resposta de escrita.

O esquemático da Fig. 6 mostra sistematicamente um sistema implementado com base no protocolo AXI, o qual é formado por meio da conexão de *masters* (mestres) e *slaves* (escravos) em uma mesma interconexão.

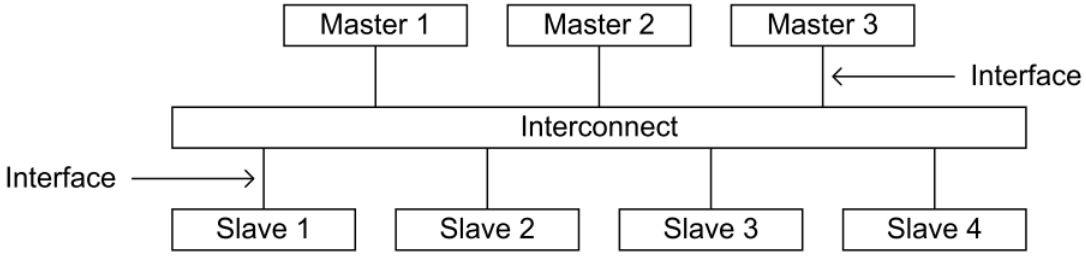


Figura 6 – Esquemático de um sistema baseado em protocolo de comunicação AXI. Fonte: (ARM, 2011).

2.3 Servo motores

Servo motor é um servomecanismo constituído de um motor e um controlador, o qual realiza medições de posição e velocidade do rotor com o objetivo de possibilitar um controle de movimento em malha fechada. Nesse contexto, geralmente são utilizados potenciômetros resistivos ou *encoders* incrementais como transdutores de posição (YANG; KE, 2000). Devido ao fato de possuírem um controle interno próprio, os servo motores promovem uma facilidade de utilização por parte do usuário, que fica responsável por fornecer ao componente apenas a velocidade desejada, que geralmente é definida por meio de um sinal modulado por largura de pulso (PWM) e o sentido de rotação do atuador. A Fig. 7 mostra um diagrama simplificado de funcionamento de um servo motor.

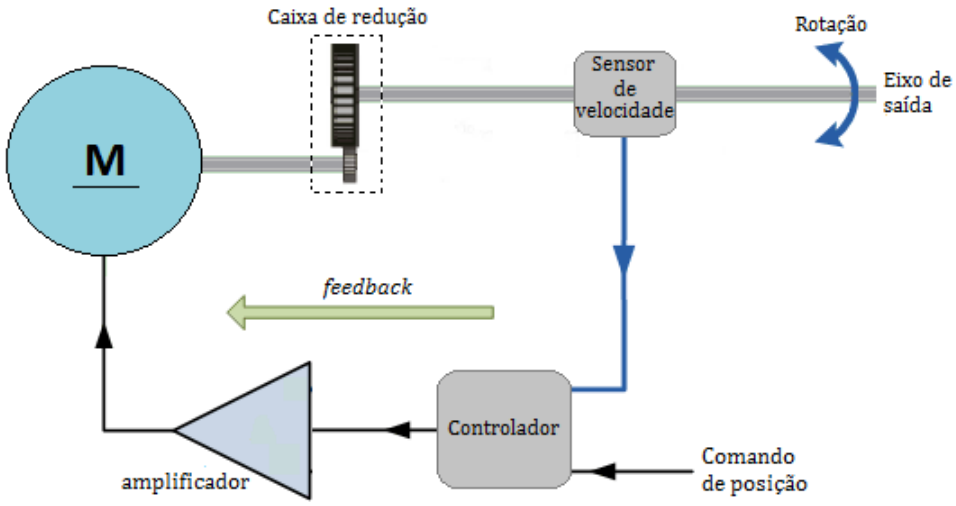


Figura 7 – Diagrama de funcionamento simplificado de um servo motor.

2.3.1 Caixa de redução de velocidade

A relação entre torque e velocidade para um servo motor de corrente contínua (DC) pode ser aproximada pela curva da Fig. 8.

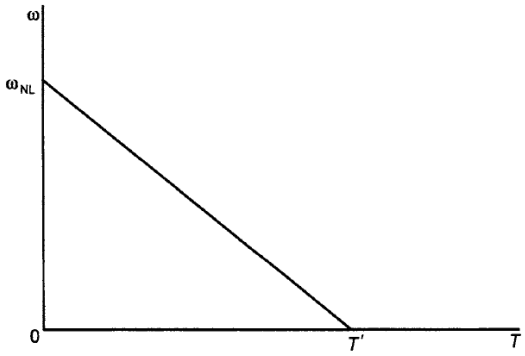


Figura 8 – Relação aproximada entre torque e velocidade para um servo motor. Fonte: (MORETON, 2000)

Neste diagrama, ω_{NL} representa a velocidade angular máxima alcançada pelo eixo de saída do motor sem que nenhum torque seja aplicado ao mesmo e T' o máximo torque suportado pelo eixo do motor quando o mesmo encontra-se em repouso, ou seja, quando $\omega = 0$. Por meio do diagrama, percebe-se que a velocidade e o torque em um servo motor DC são inversamente proporcionais. Por este motivo, geralmente é utilizada uma caixa de redução de velocidade, composta por um conjunto de engrenagens, com o intuito de reduzir a velocidade do eixo de saída do servo motor e dessa forma promover um aumento no torque fornecido pelo mesmo.

2.3.2 Sensor de velocidade

Um componente imprescindível em um servo motor é o sensor de velocidade. Este sensor possui a função de retornar ao controlador do sistema a velocidade atual do eixo de saída do motor, para que assim o mesmo possa calcular a posição e a velocidade do eixo e verificar se o comando de posição enviado anteriormente foi executado corretamente. Caso haja alguma diferença entre o sinal de posição enviado e a posição verificada com auxílio do sensor, então um sinal de erro é calculado e um comando é enviado novamente ao motor, o que caracteriza um sistema de controle em malha fechada. Sensores muito utilizados para realizar esta função em servo motores são os *encoders* incrementais. Dentre os tipos mais usuais de *encoders* incrementais encontram-se os ópticos e os indutivos.

Um *encoder* óptico é um sensor desenvolvido geralmente em um disco de material plástico transparente, o qual possui algumas partes transparentes e outras opacas (BRAGA, 2017). A Fig. 9 ilustra um *encoder* óptico acoplado ao eixo de um atuador com o intuito de aferir a velocidade do mesmo.

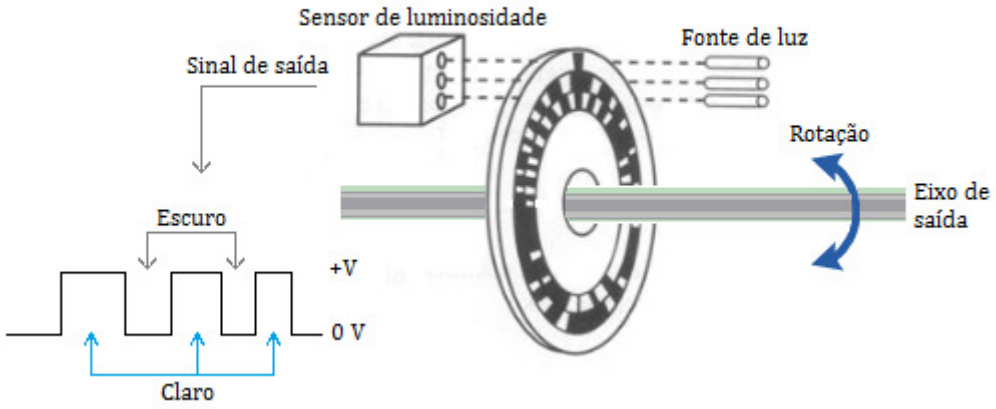


Figura 9 – *Encoder* disposto em um eixo movido por velocidade angular. Fonte: adaptado de (BRAGA, 2017).

Nesta configuração, o sensor é movido pelo eixo de saída de um atuador. Em uma face do *encoder* existe uma fonte de luz; disposto simetricamente do lado oposto existe um sensor de luminosidade. Devido à rotação da parte móvel do dispositivo, a luz fornecida pela fonte luminosa atravessa os pontos transparentes da estrutura. Entretanto, ao ser emitida sobre a superfície opaca do disco, a luz é bloqueada. Tanto o bloqueio quanto a emissão de luz são identificados pelo sensor de luminosidade. Quando o mesmo identifica alguma luminosidade, gera nível lógico alto, quando não, um nível lógico baixo é gerado. Dessa forma, um trem de pulsos digitais composto por '0' e '1' é obtido do sensor, como ilustrado na Fig. 9. Com este sinal, é possível estabelecer uma relação entre a quantidade de pulsos lida e a velocidade do rotor.

Um *encoder* pode ser classificado em: simples, de quadratura e absoluto. O *encoder* simples fornece apenas um sinal que permite, por meio de contagem de pulsos, calcular o deslocamento do rotor relativo a um ponto inicial. Um *encoder* em quadratura fornece dois sinais defasados em 90°, característica esta que justifica o próprio nome dado ao sensor (Gurley Precision Instruments, 2009); estes sinais são constituídos de pulsos provindos de dois canais, normalmente nomeados como canal A e canal B. A leitura de um só canal, como ocorre no *encoder* simples, permite apenas o cálculo da posição relativa e da velocidade do servo motor por meio da contagem dos pulsos provindos do sensor. Já no *encoder* de quadratura, a existência de outro canal permite determinar também o sentido de rotação do atuador. A Fig. 10 ilustra os sinais gerados pelo sensor de quadratura.

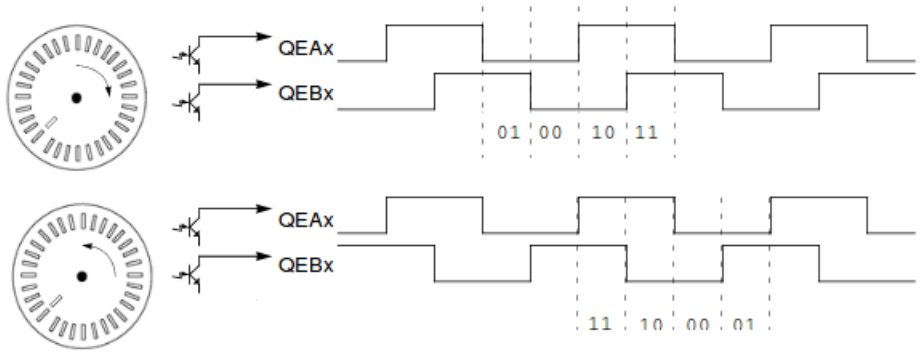


Figura 10 – Sinais gerados por um *encoder* de quadratura. Fonte: adaptado de (MICROCHIP, 2014).

O princípio óptico é utilizado para construção de um tipo mais preciso de *encoder*, o *encoder* absoluto. Um *encoder* absoluto possui várias trilhas contendo demarcações transparentes e opacas. Cada uma destas trilhas possui um sensor de luminosidade correspondente, o que faz com que cada posição seja referenciada por uma codificação digital. Apesar de fornecerem um valor de posição absoluta, o alto valor aquisitivo agregado a este tipo de sensor limita a sua utilização, fazendo com que os *encoders* incrementais sejam mais populares. A Fig. 11 ilustra a disposição dos elementos para o funcionamento de um *encoder* absoluto.

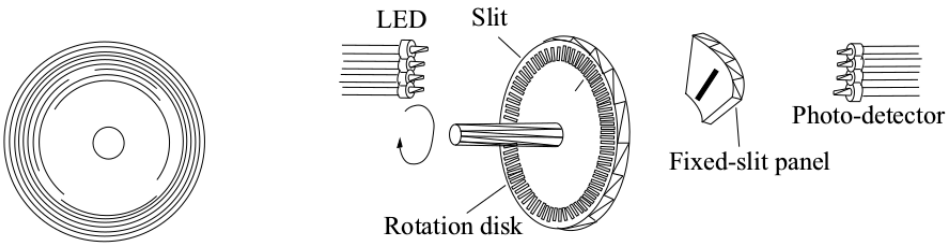


Figura 11 – Funcionamento do *encoder* absoluto. Fonte: (SUH et al., 2008).

Um *encoder* incremental, como o *encoder* simples e o *encoder* em quadratura, pode ser construído também utilizando-se os princípios de indução eletromagnética. Sensores deste tipo são conhecidos como *encoders* indutivos. Nestes dispositivos, o princípio da indução é utilizado para medir a posição relativa de um rotor com relação a um estator, entretanto, os sinais elétricos que fornecem são similares àqueles obtidos com um *encoder* óptico (ZETTLEX, 2017). A equação 2.1 fornece o ângulo de rotação, representado por $\Delta\theta$. Nesta equação, a variável n_R representa o número de pulsos por rotação do *encoder*. Este valor é fornecido no *datasheet* do dispositivo, pois depende das características de fabricação do sensor. A variável n_L representa o número de pulsos lidos pelo sensor.

$$\Delta\theta = 360^{\circ}\frac{n_L}{n_R} \tag{2.1}$$

A Equação 2.2 fornece a velocidade angular $\Delta\omega$ do disco do sensor. Para tanto, deve ser definido um intervalo de tempo Δt, que corresponde ao intervalo entre leituras realizadas pelo sensor infravermelho. Dessa forma:

$$\Delta\omega = \frac{\Delta\theta}{\Delta t} \tag{2.2}$$

3 Materiais e Métodos

Este capítulo apresenta a arquitetura geral da interface de controle implementada, as descrições técnicas dos principais equipamentos utilizados em seu desenvolvimento, além dos procedimentos experimentais reaizados para a validação do sistema.

3.1 Arquitetura Geral

O sistema implementado neste trabalho deve funcionar como uma interface entre os conjuntos dinâmicos (motores e *encoders*) e os controladores do exoesqueleto. Por esse motivo, o mesmo deve ser desenvolvido de maneira a facilitar o acesso a esses componentes. O diagrama da Fig. 12 ilustra de maneira geral os principais blocos e subsistemas que compõem a interface de controle implementada.

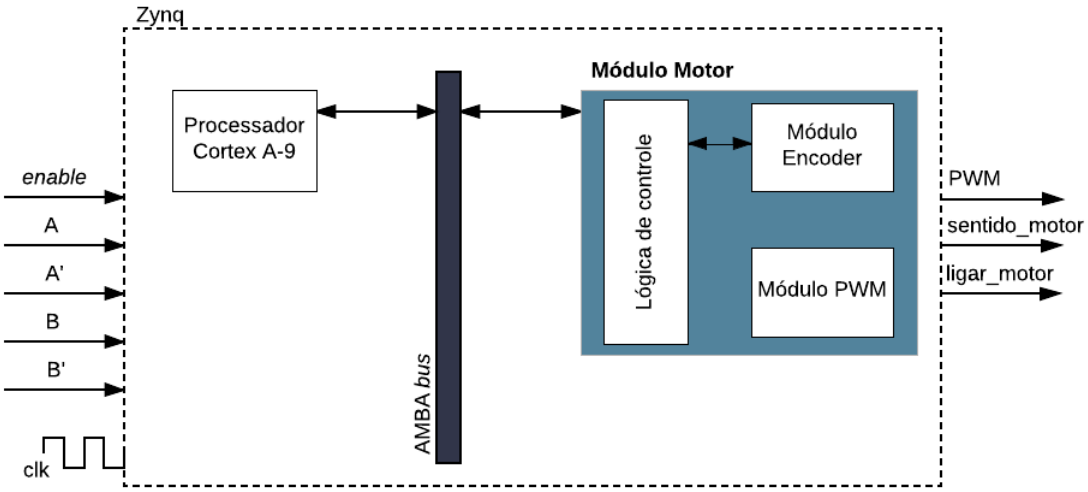


Figura 12 – Visão geral da interface de controle.

Foram definidos inicialmente os parâmetros de entrada e saída do sistema com base nos dados fornecidos pelo *encoder* e nos sinais necessários para o controle do motor. Além dos dois canais comuns, A e B, o *encoder* de quadratura utilizado fornece o complemento de ambos como forma de validar os sinais gerados. O motor, por sua vez, recebe comandos de *enable* (ligar_motor) e sentido de rotação (sentido_motor), além do sinal PWM para controle de velocidade. Com base nestas informações, as principais entradas e saídas definidas para o sistema encontram-se na Tabela 1.

Tabela 1 – Principais entradas e saídas do sistema.

A	Entrada
A'	Entrada
B	Entrada
B'	Entrada
ligar_motor	Saída
sentido_motor	Saída
PWM	Saída

A interface de controle é responsável por gerar um sinal PWM e por definir o sentido de rotação do motor (sentido_motor), baseando-se nos valores de *duty cycle* e sentido recebidos, seja pelo próprio FPGA ou por meio do barramento AMBA que estabelece uma comunicação entre HW/SW. Os sinais do *encoder* são lidos e processados pelo sistema, o que possibilita encontrar a posição relativa e a velocidade do motor. Além disso, o sistema é capaz de receber um valor de ângulo relativo a uma certa referência e realizar a movimentação do atuador até esta posição, retornando a cada momento valores de velocidade e posição relativa do eixo do motor, que podem ser visualizados em *software*.

Em FPGA, foi implementado um IP *core* responsável por gerar um sinal PWM com base no valor de *duty cycle* definido pelo usuário do sistema em *software* e por realizar a leitura do *encoder*, além de calcular os valores necessários para definição da velocidade e da posição relativa do motor. Um IP *core* nada mais é que um componente virtual que realiza alguma função específica. Alguns IP *cores* são distribuídos por empresas e organizações especializadas em seu desenvolvimento, como ARM, Altera e *OpenCores*. Estes componentes virtuais podem ser vendidos ou ainda distribuídos gratuitamente em comunidades de código aberto como a *OpenCores*.

O desenvolvimento da interface de controle na plataforma de *hardware*, no formato de um IP *core*, pode ser justificado pelo fato de o FPGA possibilitar a execução de tarefas em tempo real. Mesmo que o controle do atuador fosse realizado totalmente no processador ARM, para que a velocidade e a posição relativa fossem devidamente calculadas na camada de *software*, seria necessário utilizar um sistema operacional de tempo real (RTOS), devido aos requisitos de controle de tempo associados a este tipo de sistema operacional, que permitem uma pré-definição dos períodos de tempo associados à execução de determinadas tarefas.

Devido às funcionalidades apresentadas pela plataforma de *hardware*, tornou-se mais prático o desenvolvimento da interface de controle no FPGA, levando-se também em consideração o fato desta plataforma favorecer a reutilização do módulo implementado, o que permite um desenvolvimento mais ágil de sistemas utilizando-se os métodos de *design reuse*. O fato de o barramento AXI disponível na plataforma Zybo possibilitar uma comunicação ágil entre *cores* também colabora para que a interface com os atuadores e

encoders possa ser realizada na camada de *hardware* sem prejuízo ao funcionamento da interface, mesmo que o sistema de controle seja embarcado em *software*.

Completando a arquitetura geral do sistema, tem-se os circuitos optoacopladores, desenvolvidos para que os sinais de controle possam ser enviados entre os componentes. Estes circuitos são necessários pois a tensão típica de funcionamento do controlador ESCON 70/10 e do *encoder* MILE é de 5V, enquanto que os PMODs disponíveis na placa Zybo para conexão com periféricos funcionam com uma tensão de 3.3V.

O contexto abordado neste trabalho originou três grandes etapas de execução distintas: implementações em *hardware*, implementações em *software* e prototipagem do sistema.

3.2 Materiais

3.2.1 *System on Chip* Zynq Zybo

O SoC utilizado neste trabalho é uma placa Zybo (*Zynq Board*) produzida pela empresa Xilinx, mostrada na Fig. 13. Esta placa integra um processador dual-core ARM Cortex-A9 com operação de 650Mhz com um FPGA Xilinx 7-*series*.

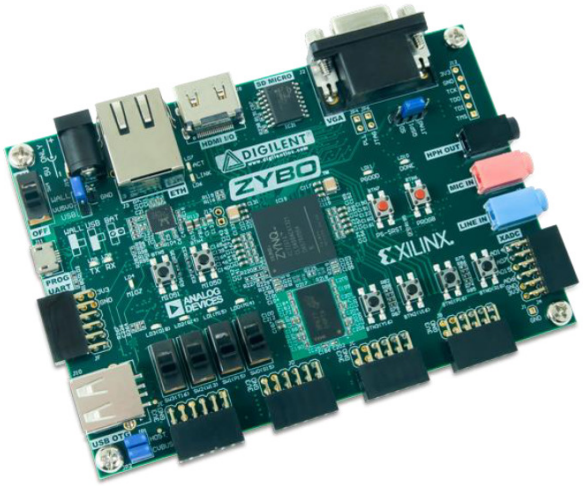


Figura 13 – Plataforma de desenvolvimento Zybo. Fonte: (DIGILENT, 2016).

Dentre os periféricos existentes neste SoC, encontram-se 6 *pushbottons*, 4 *switches* e 5 LEDs, além de 6 conectores PMOD - *Peripheral Module* utilizados para conexão de periféricos externos à placa. Esta placa fornece uma frequência externa de 125 MHz ao FPGA pelo pino L16. Este clock é totalmente independente daquele fornecido para a comunicação entre o PS – *Processor System* e o PL - *Programmable Logic*, cujo valor típico é de 100 MHz.

A tensão de alimentação recomendada para alimentação é de 5V, sendo que pode ser realizada por meio do próprio cabo micro USB utilizado para comunicação UART, ou por meio de uma fonte de alimentação externa, cujos valores recomendados são de 5V/2.5A, de acordo com o *datasheet* do SoC. Para configuração do modo de alimentação da placa, basta modificar o pino JP7 para a posição *wall* ou USB, sendo que *wall* corresponde à alimentação realizada com a fonte externa.

3.2.2 Motor Maxon EC90 *flat*

Neste trabalho, foi utilizado um motor *brushless* da marca Maxon acoplado à uma caixa de redução modelo GP62 e um *encoder* MILE, ambos da mesma marca e compatíveis com o motor EC90 *flat*. O motor acoplado ao *encoder* pode ser visto na Fig. 14 a seguir. A Tabela 2 a seguir mostra algumas características do motor.

Figura 14 – Motor Maxon EC90 *flat* acoplado ao *encoder* MILE. Fonte: (FARNELL,).

Tabela 2 – Características do motor EC90 *flat*. Fonte: adaptada de (MAXON, 2013).

Tensão nominal [V]	24
Corrente [mA]	538
Velocidade nominal [rpm]	2590
Torque nominal [mN.m]	444.4
Eficiência máxima (%)	84
Potência [W]	90
Diâmetro do motor [mm]	90

Este motor vem acompanhado de um controlador ESCON 70/10, mostrado na Fig. 15. Este controlador foi utilizado somente como interface para conversão dos comandos de controle em sinais de potências para os motores. Para realizar a configuração do controlador ESCON 70/10, utilizou-se o *software* ESCON Studio 2.2.

Os *encoders* MILE utilizados possuem resolução de 3200 pulsos por revolução e são caracterizados por possuírem boa acurácia para controle em alta e baixa velocidade.

Estes sensores fornecem dois sinais transmitidos por meio dos canais A e B, que possuem tensão típica de 5V, mesma tensão indicada para alimentação dos sensores.

Figura 15 – Controlador ESCON 70/10. Fonte: (MAXON MOTOR, 2017)

A Fig. 16 a seguir mostra a caixa de redução modelo GP62. A Tabela 3 mostra algumas características da caixa de redução GP62.

.

Figura 16 – Caixa de engrenagens Maxon GP62. Fonte: MAXON MOTOR.

Tabela 3 – Características da caixa de redução GP62. Fonte: adaptada de (MAXON MOTOR, 2012).

Redução	100:1
Eficiência máxima (%)	70
Peso [g]	1540
Torque máximo suportado pelo eixo [N.m]	50

3.3 Procedimentos Experimentais

Os procedimentos experimentais foram realizados de maneira a validar individualmente cada uma dos submódulos existentes no IP *core* ModuloMotor e, posteriormente, a interface de controle como um todo. O sinal PWM gerado pelo Módulo PWM foi validado

inicialmente com auxílio de um osciloscópio. Posteriormente, realizou-se um experimento no qual o valor de *duty cycle* foi modificado e o respectivo sinal foi adquirido com o controlador ESCON 70/10. Neste experimento, foram utilizados os circuitos optoacopladores para que os sinais pudessem ser enviados da plataforma Zybo para o controlador. Estes circuitos, por sua vez, foram anteriormente testados em bancada utilizando uma fonte de alimentação, um osciloscópio e um gerador de funções. Nesses testes, uma onda quadrada em alta frequência foi inserida no circuito com tensões de 3.3V e 5V e os respectivos sinais de saída foram observados com o osciloscópio.

O controlador ESCON 70/10 é configurado por meio do *software* ESCON *studio*, em que os parâmetros são definidos de acordo com as características dos sensores e motores utilizados. Com este *software*, é possível visualizar o valor de *duty cycle* do sinal PWM inserido em uma das portas digitais do controlador. Para o teste do Módulo PWM, em primeiro momento foi utilizado somente o FPGA sem qualquer conexão com o processador ARM por meio do barramento AXI. Dessa forma, para este experimento, o *duty cycle* do sinal PWM foi modificado diversas vezes no próprio FPGA. Após vários testes, foi possível validar o sinal PWM gerado pelo módulo. A Figura 17 mostra um dos valores obtidos durante os testes, visualizado por meio da interface existente no *software* ESCON *studio*.

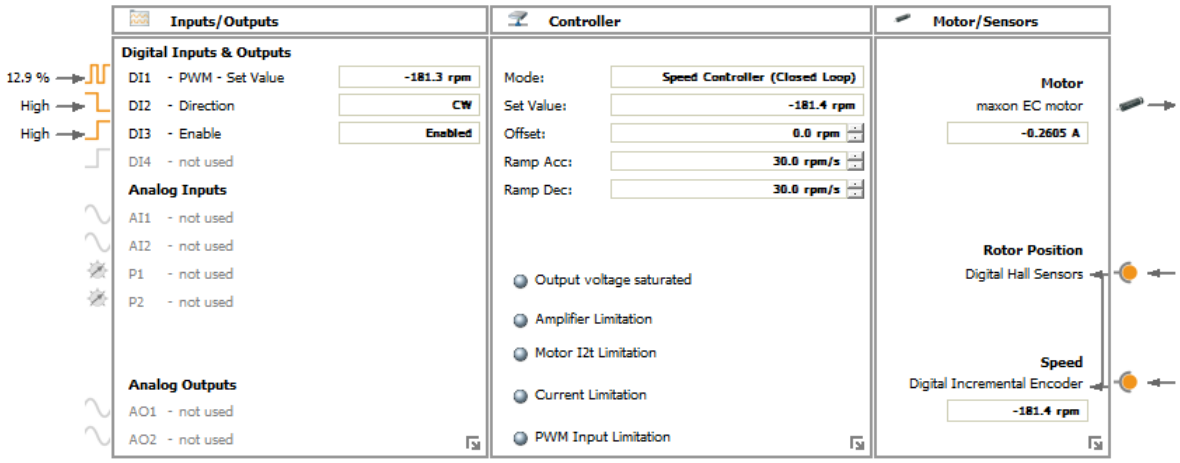


Figura 17 – Valor obtido para um sinal PWM com *duty cycle* de 13%.

Por meio do mesmo experimento descrito anteriormente, foi possível realizar os testes de acionamento do atuador e a leitura dos sinais do *encoder*, além de ter sido possível validar os valores de velocidade calculadores com base na leitura dos sinais provindos do *encoder*. A Figura 18 mostra as formas de onda produzidas pelo sensor, obtidas com um osciloscópio.

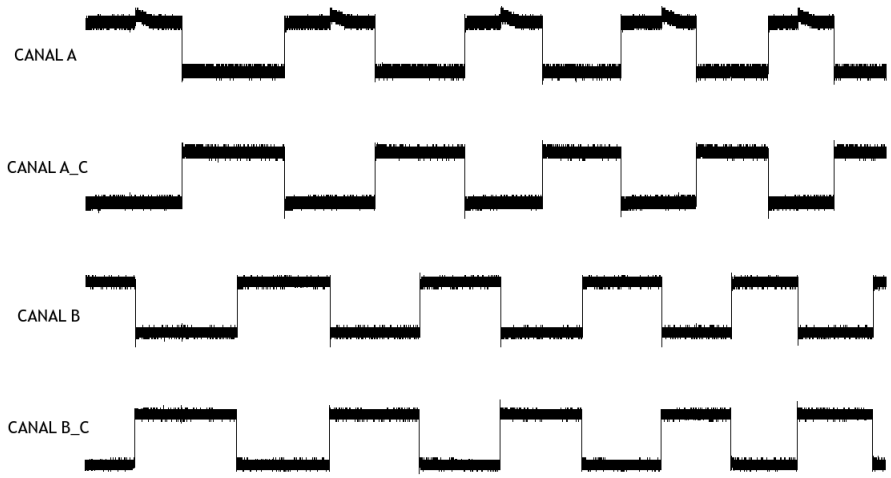


Figura 18 – Sinais do *encoder* obtidos com osciloscópio.

Uma forma encontrada para calibrar o *encoder* foi utilizando o próprio *software* ESCON *studio*, que possui a função de *auto-tuning*. Por meio desta, é possível observar os resultados ideais e os obtidos com a calibração, como ilustra a Fig. 19. Após validado o comportamento do sistema, foi possível realizar as leituras e os devidos cálculos de velocidade a fim de compará-los aos obtidos com o controlador ESCON, valores estes que puderam ser observados pela interface do *software*, assim como mostra a Fig. 17. Já os valores de velocidade foram calculados em *software* com o SDK.

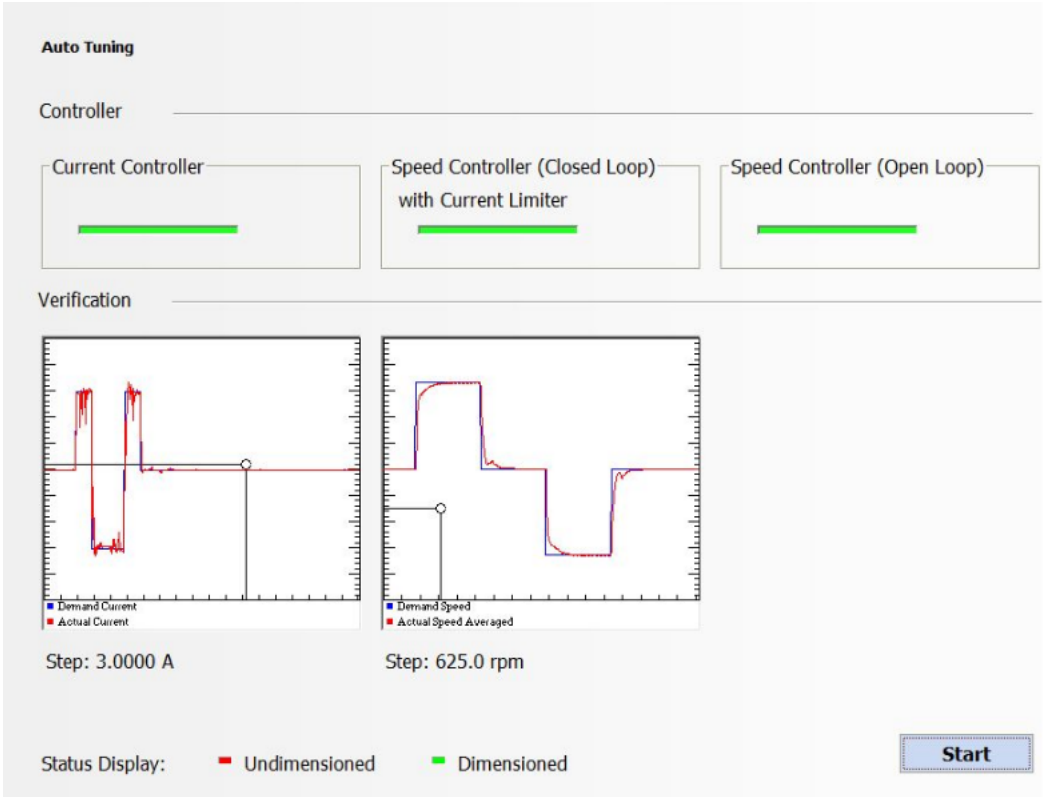


Figura 19 – Processo de calibração de um sistema qualquer, semelhante ao realizado neste trabalho. Fonte: (KAMINSKI, 2017).

Na etapa de testes com o SDK, foram realizados acionamentos do atuador por meio do envio do valor de *duty cycle* do ARM para o FPGA utilizando o SDK. A quantidade de pulsos baseada no sinal em quadratura calculada pelo Módulo Encoder era enviada ao *software* para que o cálculo de velocidade do motor fosse realizado. Este valor era validado de acordo com o *duty cycle* definido inicialmente e de acordo com o valor mostrado no *software* ESCON *studio*. Finalizada a validação dos módulos utilizando o SDK, foram desenvolvidas as lógicas de controle de posição e os testes posteriores foram realizados com aplicações desenvolvidas em linguagem C no Linux embarcado no processador ARM. Estas aplicações consistiam em rotinas de execução, nas quais eram definidos ângulos de deslocamento para o eixo do motor e o resultado era observado com o auxílio de um transferidor, como descrito no Capítulo 5.

4 Desenvolvimento

Neste capítulo serão descritas as implementação em *hardware*, as implementação em *software*, além da etapa de prototipagem do sistema. Os códigos desenvolvidos tanto em *hardware* quanto em *software* encontram-se no repositório do GitLab: https://gitlab.com/anapaulachavierrr/Exoesqueleto.

4.1 Implementações em *hardware*

O módulo de controle dos motores foi dividido em três componentes: Módulo Encoder, Módulo PWM e Lógica de Controle. Após projetados os módulos, foram realizadas simulações comportamentais no *software* Vivado versão 2015.3.

4.1.1 Módulo PWM

O Módulo PWM tem o objetivo de receber um valor de *duty cycle* e a partir disso gerar um sinal PWM. Este valor é recebido por meio do barramento AMBA AXI, especificamente pelo registrador *slv_reg5*. O diagrama da Figura 20 mostra os componentes que compõem este módulo.

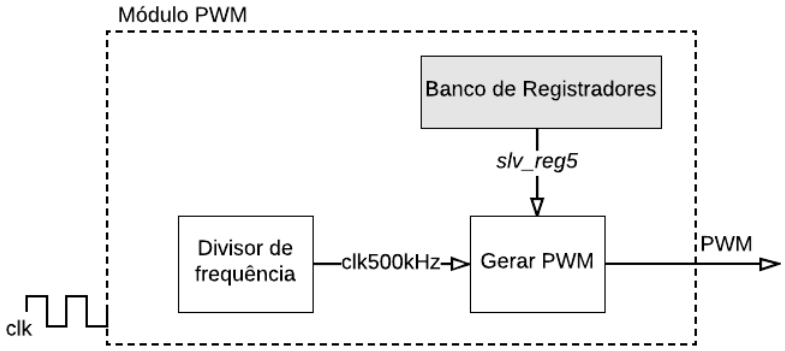


Figura 20 – Visão geral do Módulo PWM.

De acordo com o *datasheet* do controlador ESCON 70/10, utilizado como módulo de potência para o acionamento dos atuadores, a resolução de leitura do sinal PWM é de 0.1%, em uma faixa de *duty cycle* que vai de 10% a 90%. Levando em consideração esta informação, o módulo para gerar o sinal PWM foi projetado de maneira a fornecer um sinal com uma resolução igual a 0.1%. Isso significa que um período do sinal possui um *duty cycle* máximo de:

$$duty = \frac{100}{0.1} = 1000 \tag{4.1}$$

Além disso, a faixa de frequência de leitura do sinal PWM pelo controlador ESCON 70/10 vai de 10 Hz à 5 kHz. Nesse sentido, o módulo foi projetado para gerar um sinal com 500 Hz de frequência. A Fig. 21 a seguir mostra os blocos operacional e de controle utilizados para gerar o sinal. Estes blocos fazem referência ao componente "Gerar PWM"mostrado na Fig. 20. No diagrama, foram desconsiderados os eventos temporais de *clock* e alguns sinais de inicialização do sistema como forma de simplificar a máquina de estados.

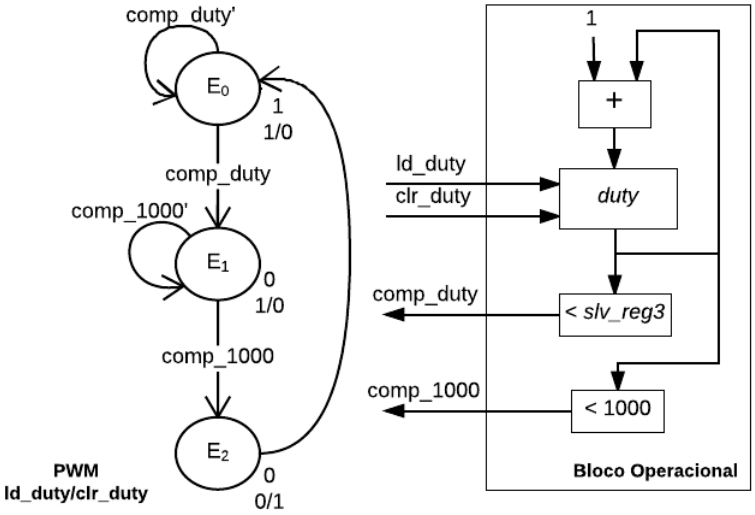


Figura 21 – Bloco operacional e de controle do componente Gerar PWM.

O valor de *duty cycle* é definido por meio do registrador *slv_reg5*. Este valor limita o tempo em que o sistema deve permanecer no estado E0, no qual o sinal PWM se encontra em nível lógico alto. Quando o registrador local *duty* assume um valor igual ao *duty cycle* desejado, a máquina de estados migra para o estado E1 e permanece neste até que o valor de *duty* seja menor que 1000. Este estado faz com que o sinal PWM passe a ter nível lógico baixo. Como definido anteriormente, 1000 corresponde ao valor máximo de *duty cycle*, ou seja, 100%. Dessa forma, a frequência do sinal PWM de saída será igual a:

$$frequência = \frac{500000}{1000} = 500Hz \tag{4.2}$$

A Figura 22 ilustra, para um *duty cycle* qualquer, a relação entre o sinal PWM e os respectivos estados de operação. Nesta, o estado E2 foi desconsiderado, pois o sistema permanece nesse estado por apenas um ciclo de *clock* a cada período do sinal PWM gerado.

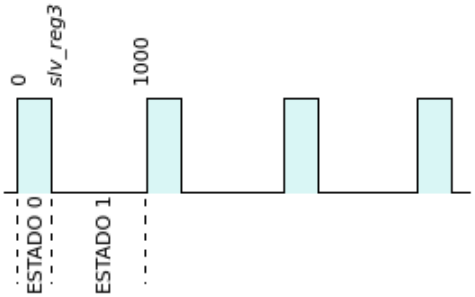


Figura 22 – Sinal PWM.

A Figura 23 mostra o sinal gerado em uma das simulações do Módulo PWM.

Figura 23 – Simulação do Módulo PWM.

4.1.2 Módulo Encoder

O Módulo Encoder foi projetado para gerar um sinal em quadratura baseado nos canais A, B, A' e B' do sensor de posição. Este sinal é utilizado pela lógica de controle para determinar os valores de posição relativa e velocidade dos servo motores. A implementação desse módulo foi baseada no IP *core* para leitura de *encoders* incrementais de quadratura disponível na plataforma OpenCores (2009). Esquematicamente, o módulo encontra-se descrito na Figura 24.

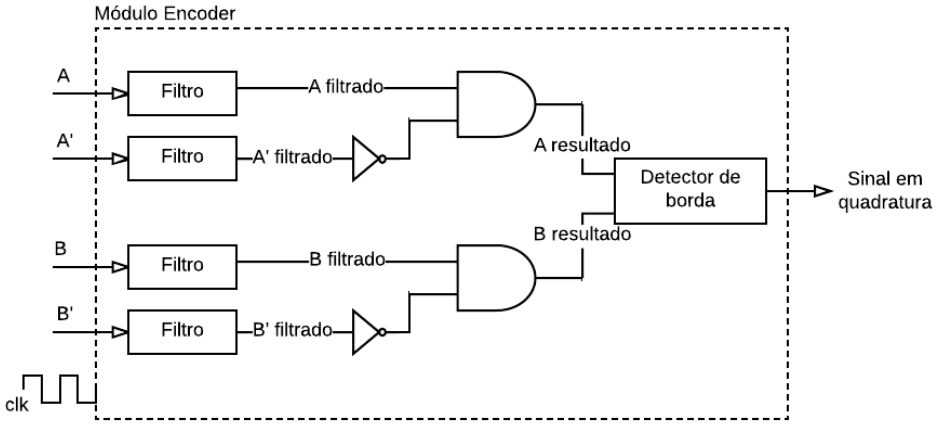


Figura 24 – Visão geral do Módulo Encoder.

O sinal em quadratura esperado encontra-se ilustrado na Figura 25. Para a implementação do mesmo, realizou-se a detecção das bordas de subida e descida dos sinais A_resultado e B_resultado com o componente "Detector de Borda". Estes, por sua vez, são obtidos por meio de uma operação lógica entre os canais A, B e seus respectivos complementos A' e B'.

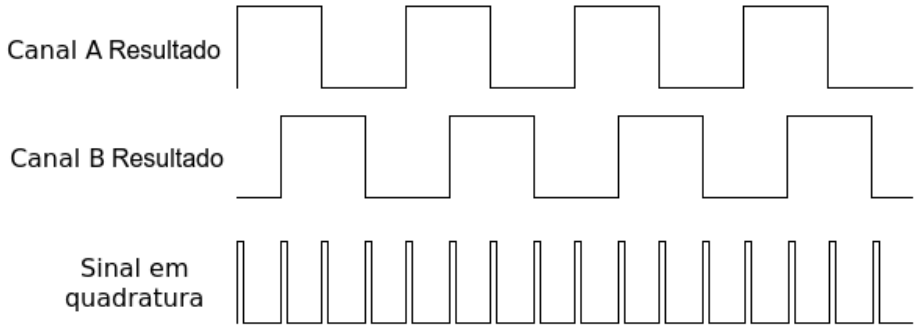


Figura 25 – Sinal em quadratura esperado após implementação do Módulo Encoder.

Utilizando os sinais A' e B', esperava-se diminuir a ocorrência de ruídos, que nestes casos assemelham-se a picos. Nesse sentido, implementou-se também um simples filtro digital composto por *flip flops*. Os componentes para filtragem e detecção de bordas encontram-se descritos a seguir.

4.1.2.1 Filtro

O circuito da Figura 26 apresenta esquematicamente o módulo em VHDL implementado para a filtragem dos sinais do *encoder*.

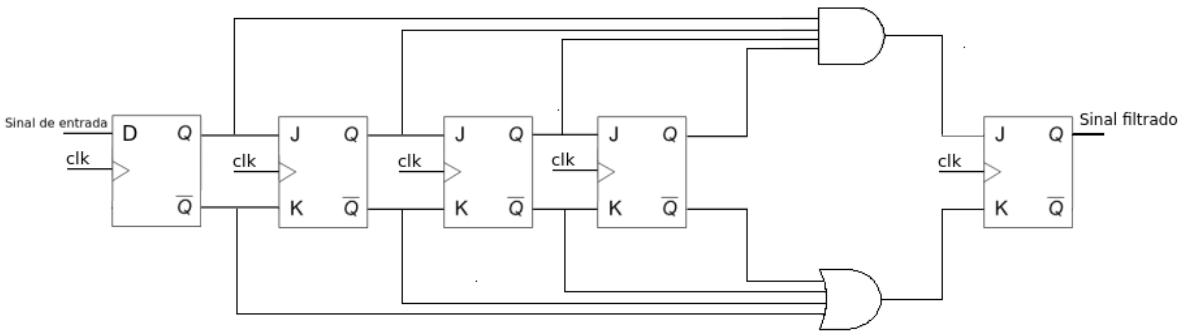


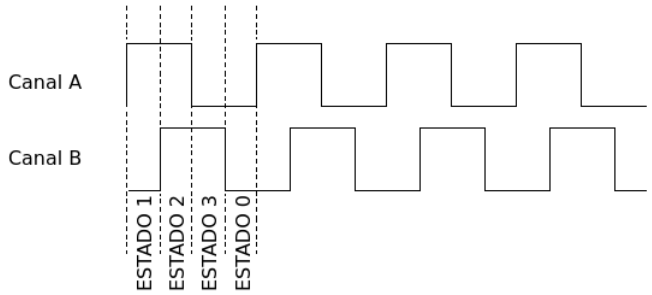
Figura 26 – Filtro de sinais utilizando *flip-flops*.

Com este filtro, um sinal em nível lógico alto ou baixo deve passar por quatro *flip flops* antes de ser considerado válido. Como um ruído assemelha-se a um pico, é válido considerar que o mesmo não se mantém em nível lógico alto por quatro ciclos de *clock*. Dessa forma, um ruído intermitente no sinal de entrada desapareceria na saída do filtro. Entretanto, para que o filtro funcione corretamente, é necessário garantir que a frequência de operação dos *flip flops* não seja tão rápida a ponto de o ruído permanecer em nível lógico alto por quatro ciclos de *clock* consecutivos.

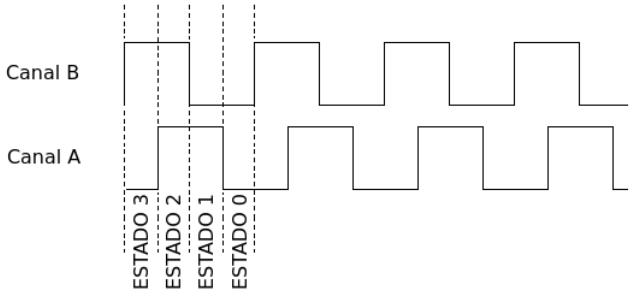
Dessa forma, para o projeto do filtro em questão, considerou-se a frequência máxima que os sinais do *encoder* podem alcançar, que nesse caso é de aproximadamente 267 kHz. Assim, o sinal *clk* escolhido possui um período que é aproximadamente 50 vezes o período correspondente à frequência de 267 kHz.

4.1.2.2 Detector de Borda

O componente para detecção de borda recebe os sinais A_resultado e B_resultado, gerados por meio da operação lógica entre A e A', B e B', respectivamente, e por meio disso gera o sinal em quadratura. Os dois sinais, A_resultado e B_resultado são defasados de 90°, devido às características dos *encoders* de quadratura de possibilitarem a detecção do sentido de rotação do motor. Assim, para o projeto deste submódulo, considerou-se as duas situações mostradas na Figura 27.



(a) Sinal B defasado de A.



(b) Sinal A defasado de B.

Figura 27 – Possibilidades de comportamento dos sinais A e B.

A máquina de estados da Figura 28 mostra, de forma simplificada, a relação entre as entradas A e B, a saída *sinal_quad* e a sequência de estados correspondente. Estas relações foram encontradas considerando as duas possibilidades de ocorrência dos sinais A e B mostradas na Figura 27. Os estados E0, E1, E2 e E3 da máquina de estados correpondem, respectivamente, aos estados ESTADO0, ESTADO1, ESTADO2 e ESTAD03 da Figura 27.

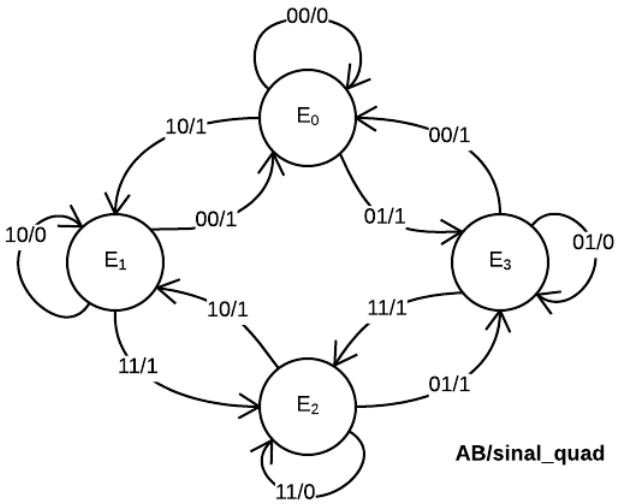


Figura 28 – Esquemático da máquina de estados implementada em VHDL.

A Figura 29 mostra uma das simulações realizadas para teste do Módulo Encoder. Nela, podem ser vistos os sinais A_res e B_res, referentes aos sinais A_resultado e B_resultado, respectivamente, e o sinal em quadratura sinal_quad.

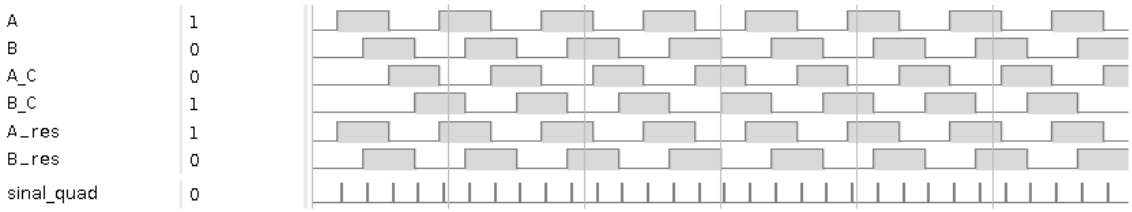


Figura 29 – Simulação do Módulo Encoder.

4.1.3 Lógica de Controle

Foi implementado um controle de posição em malha fechada para controlar a posição relativa do motor. Utilizou-se o sinal interno *sinal_quad*, gerado pelo Módulo Encoder, para definir o deslocamento correto do motor por meio de contagem de pulsos. A Figura 30 mostra esquematicamente a visão geral do módulo Lógica de Controle.

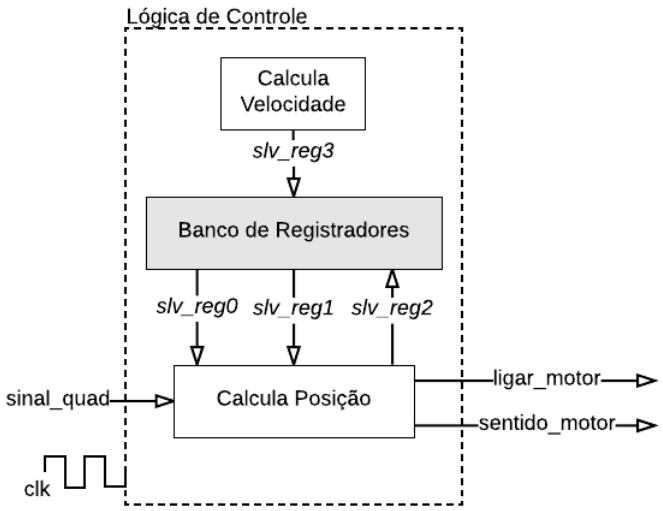


Figura 30 – Visão geral do módulo Lógica de Controle.

O módulo funciona de acordo com os valores recebidos por meio dos registradores *slv_reg0* e *slv_reg1*, descritos na Tabela 4.

Tabela 4 – Relação dos valores contidos nos registradores.

slv_reg0	sentido_motor
slv_reg1	posicao
slv_reg2	posicao_validada
slv_reg3	velocidade_validada

Por meio de uma relação entre o ângulo relativo no qual o motor se encontra e o valor contido em *slv_reg1* é definido o quanto o motor deve se deslocar. Nessa relação, é considerado também o sentido em que o atuador deve girar, definido pelo valor contido em *slv_reg0*. O componente **Calcula Posição** é responsável por esta lógica de controle e por retornar o valor da posição em que o motor se encontra durante o deslocamento por meio do registrador *slv_reg2*.

O componente **Calcula Velocidade** retorna a quantidade de pulsos lidos em um determinado período de tempo por meio do registrador *slv_reg3*. Com este valor, é possível calcular a velocidade na qual o motor está girando. Para as implementações deste trabalho, foi considerado um período de 10 ms.

4.2 Implementações em *software*

Para as implementações em *software*, foi seguido o tutorial da DIGILENT (2014).

4.2.1 Xilinx *Software Development Kit* (SDK)

Para realizar a interface entre o módulo em FPGA e o *software*, inicialmente utilizou-se o *Software Development Kit* (SDK) da Xilinx. Nessa etapa, os testes realizados se deram principalmente com a geração de um sinal PWM e com a leitura dos sinais do *encoder*. O objetivo era validar individualmente cada um dos módulos implementados em FPGA antes de acrescentá-los à customização do *hardware*.

Para a implementação do *design* do *hardware*, utilizou-se a interface gráfica disponível no *software* Vivado. Nesta interface, os IP *cores* implementados são interligados com as entradas e saídas físicas, bem como com o sistema de processamento Zynq por meio de um diagrama de blocos. A Figura 31 mostra o esquemático do *design* de *hardware* exportado para o SDK em um dos testes realizados. Neste, foi implementada uma aplicação em linguagem C em que foram definidos valores de *duty cycle* para o sinal PWM e calculada a velocidade de rotação do motor com base nos sinais gerados pelo ModuloMotor implementado em FPGA.

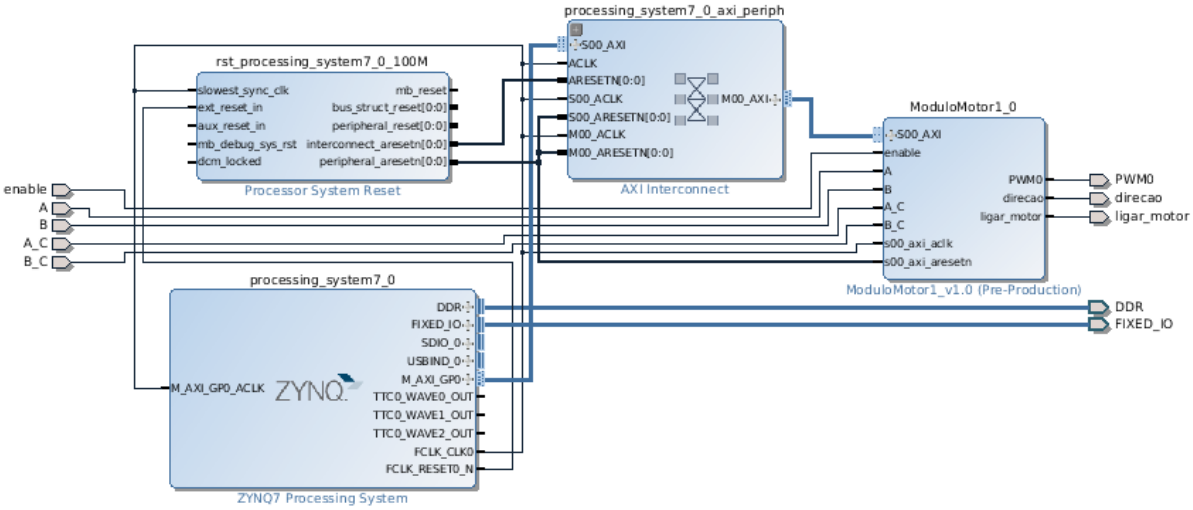


Figura 31 – *Design* de *hardware* para teste em SDK.

4.2.2 Customização de *hardware* e compilação do *kernel* Linux

Para acrescentar os módulos desenvolvidos no *hardware* do FPGA, realizou-se a etapa de customização de *hardware*. A Figura 32 mostra, de forma simplificada, a arquitetura completa da plataforma Zynq e a adição do ModuloMotor desenvolvido neste trabalho.

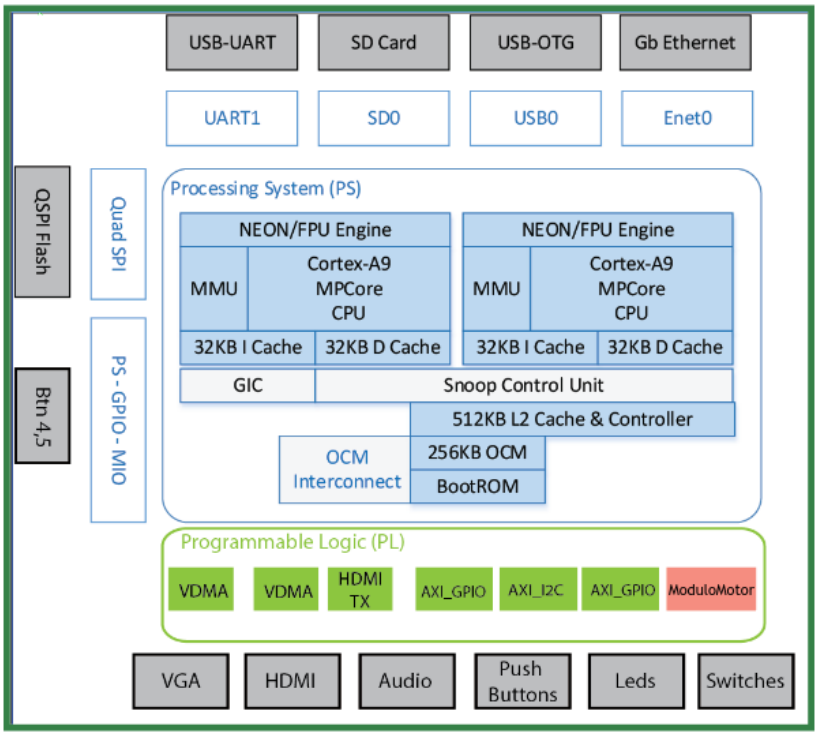


Figura 32 – Adição do ModuloMotor na arquitetura do FPGA contido na plataforma Zybo. Fonte: Adaptada de DIGILENT (2014).

Para que os módulos implementados em FPGA pudessem ser acessados diretamente pelo processador ARM, foi necessário realizar a compilação de um *kernel* Linux utilizando as configurações padrão da plataforma Zybo. Para a execução desta etapa, foram seguidos os passos contidos no tutorial (DIGILENT, 2014) com algumas modificações.

Utilizou-se um micro SD *card* de 16GB classe 10 da marca Kingston para a inserção dos arquivos compilados: **BOOT.bin**, **devicetree.dtb** e **u-boot.elf**. Este SD *card* foi dividido em duas partições, uma como FAT32 com 1GB de memória, na qual foram inseridos os três arquivos resultantes da compilação do *kernel* e a outra como ext4, na qual foi inserido o sistema de arquivos Linaro. Posteriormente, o cartão SD foi inserido na placa, o que possibilitou acessar o sistema de arquivos da mesma por meio de uma comunicação serial com um *baud rate* de 11500, como pode ser visto na Fig. 33.

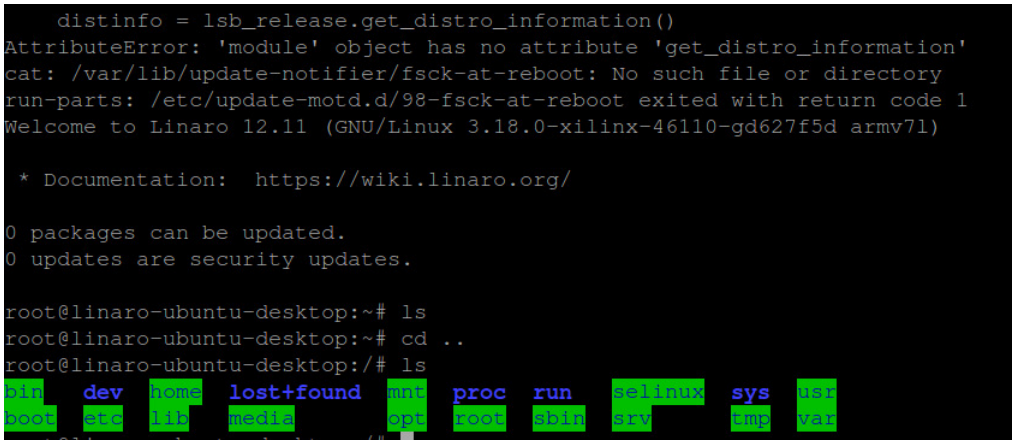


Figura 33 – Estrutura de arquivos do SoC Zybo acessada via comunicação serial.

4.2.3 *Direct Memory Acess* (DMA)

Os módulos descritos em VHDL são acessados pelo ARM por meio de *drivers* de dispositivo, pois são considerados periféricos para o processador. Como a implementação de *device drivers* requer um conhecimento específico sobre o *kernel* Linux, uma alternativa mais simples pode ser utilizada para realizar a transferência de dados entre *hardware* e *software*. O barramento AXI suporta o acesso a dispositivos por meio de *Direct Memory Access* (DMA). Este componente pode ser configurado e adicionado ao *hardware* do FPGA por meio da interface em alto nível disponível no *software* Vivado.

A Figura 34 mostra esquematicamente a conexão do bloco AXI DMA realizada em baixo nível para transmissão de dados entre o FPGA e o processador.

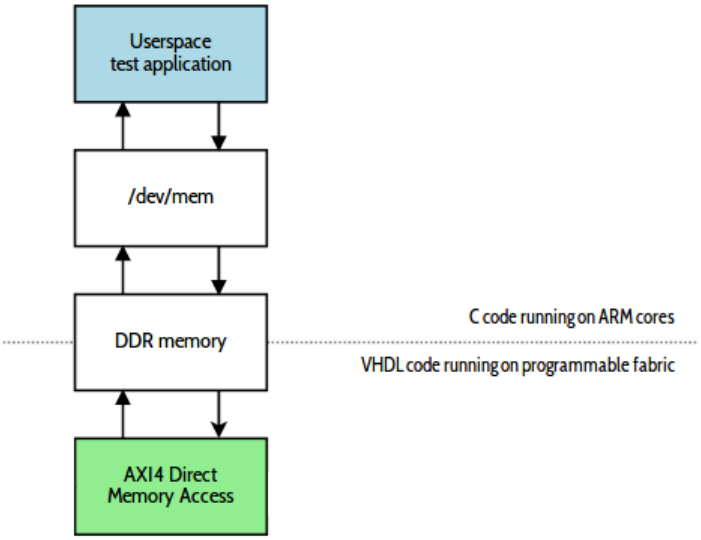


Figura 34 – Fluxo de acesso à DMA. Fonte: adaptada de Võsandi (2014).

Com esta abordagem, a leitura e a escrita de dados nos registradores existentes para

comunicação são realizadas utilizando o endereço físico atribuído ao módulo em FPGA na etapa de customização do *hardware*. Segundo Andersson (2011), existem vantagens e desvantagens em utilizar esta forma de comunicação.

Prós:

- Não necessita implementação de *device drivers*;
- Ideal para prototipagem rápida e validação de IP.

Contras:

- Nenhuma manipulação de interrupção é possível;
- Não possui proteção contra acesso simultâneo.

Devido aos objetivos iniciais desse trabalho, optou-se por utilizar a comunicação por DMA, por possuir uma implementação mais rápida e por facilitar a fase de testes dos módulos.

4.2.4 Aplicações em linguagem C

Em *software*, foi desenvolvida uma biblioteca para facilitar o acesso e a manipulação do módulo descrito em VHDL. As funções contidas nessa biblioteca realizam cálculos de posição e velocidade com base nos dados recebidos do IP *core*. Caso estes valores não sejam modificados pelo usuário, são utilizados como padrão os parâmetros dos componentes utilizados neste trabalho, descritos na Seção 3.2.2.

O envio de dados para o módulo implementado em *hardware* foi realizado de acordo com as definições da Tabela 4 e considerando que o valor de *duty cycle* é enviado por meio do registrador *slv_reg5*. O usuário do sistema define valores de velocidade em rotações por minuto (rpm) e ângulo de deslocamento relativo em graus, além do sentido de rotação do motor. Estes valores são convertidos por uma função, respectivamente, em *duty cycle* e em pulsos do sinal em quadratura que devem ser lidos pelo Módulo Encoder para completar o deslocamento até o ângulo definido. A seguir, encontram-se descritas as equações utilizadas para as conversões.

$$duty\_cycle[\%] = duty\_min + \frac{[velocidade(duty\_max - duty\_min)]}{vel\_max} \tag{4.3}$$

Na Equação 4.3, **duty_min** é o valor mínimo e **duty_max** o valor máximo de *duty cycle* aceito pelo controlador ESCON 70/10, **vel_max** é a máxima velocidade de rotação do motor utilizado e **velocidade** o parâmetro de entrada definido pelo usuário.

$$qtd\_pulsos = \frac{angulo[ppr * reducao * 4]}{360} \tag{4.4}$$

Na Equação 4.4, **ppr** representa o valor de pulsos por revolução do *encoder*, **reducao** é a taxa de redução da caixa de engrenagens acoplada ao motor e **angulo** o valor inserido pelo usuário. A Figura 35 ilustra a relação entre ângulo e quantidade de pulsos de acordo com as características dos componentes utilizados neste trabalho.

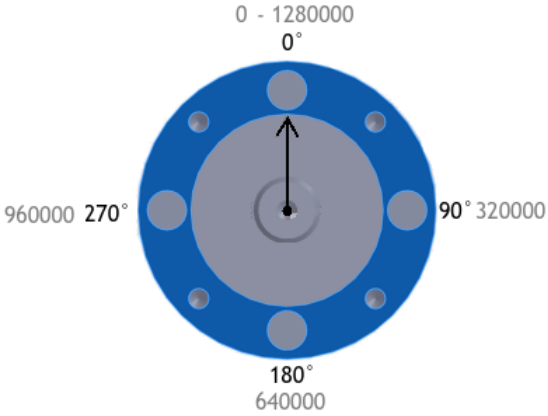
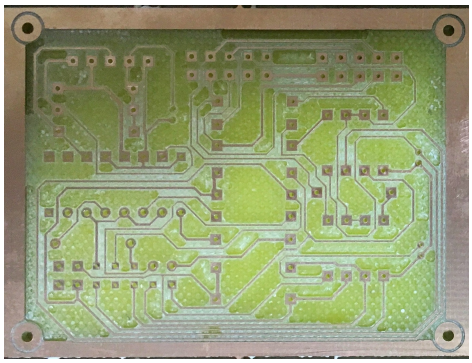


Figura 35 – Ilustração da relação entre pulsos e ângulo de deslocamento para o conjunto de componentes utilizados.

4.3 Prototipagem

Na etapa de prototipagem, foram desenvolvidas as placas de circuito impresso (PCBs) do sistema e uma estrutura física para testes. Na primeira versão da PCB, as trilhas foram impressas por um processo térmico; na segunda versão, as trilhas foram desenhadas por uma fresadora LPKF ProtoMat S103 pertencente ao Instituto Federal de Brasília (IFB) *campus* Taguatinga, como ilustrado na Figura 36. Os *layouts* dos dois circuitos foram desenvolvidos com o *software* Proteus.

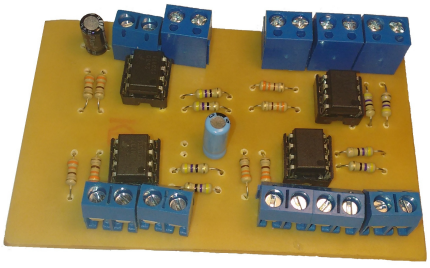
(a) Fresadora utilizada para o processo.



(b) Trilhas desenhadas com a fresadora.

Figura 36 – Processo de fabricação da PCB com fresadora.

A primeira placa implementada se mostrou pouco prática para conexão dos componentes; a segunda, por outro lado, permitiu uma conexão mais rápida e eficiente, pois foram utilizados conectores *latch* para que a conexão entre o controlador ESCON, o *encoder* e a Zybo fosse realizada por meio de cabos *flat*. O esquemático do circuito optoacoplador implementado encontra-se no Anexo B. A Figura 37 mostra as duas versões de PCB fabricadas.



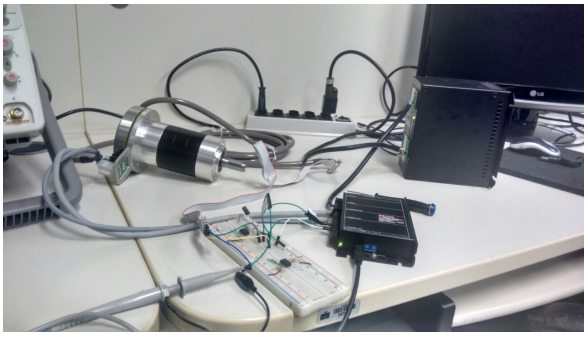
(a) Primeira versão da PCB.

(b) Segunda versão da PCB.

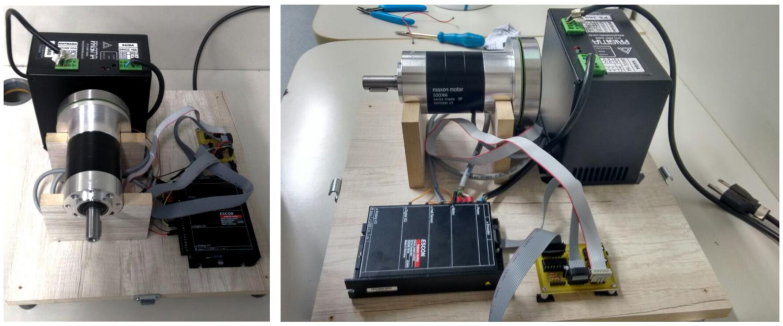
Figura 37 – PCBs desenvolvidas.

A Figura 38a mostra a bancada de testes antes da construção de um suporte para os equipamentos utilizados. Como forma de viabilizar uma validação mais precisa do controle de posição implementado e tornar mais prático o período de testes, foi desenvolvida uma

plataforma em que ficam alocados o controlador ESCON 70/10, o conjunto de motor, *encoder* e caixa de redução, a fonte para alimentação do motor e o circuito acoplador. O resultado dessa montagem pode ser visto na Figura 38b.



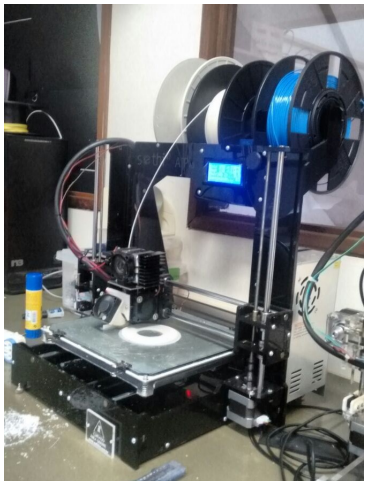
(a) Bancada de testes inicial.



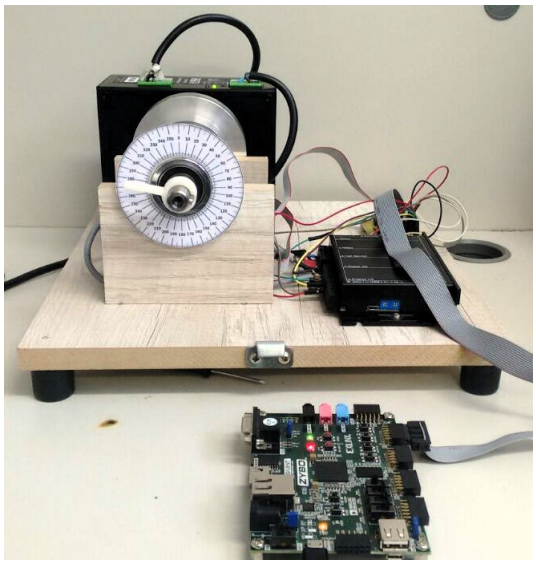
(b) Bancada de testes após construção do suporte.

Figura 38 – Bancadas de teste.

Foram impressas duas peças em uma impressora 3D, como mostra a Figura 39a, para tornar mais precisa a visualização do ângulo relativo no qual o eixo do motor se encontra. O dimensionamento de ambas pode ser visto no Anexo A. Com a montagem ilustrada na Figura 39b, foram adquiridos os resultados finais deste trabalho.



(a) Processo de impressão das peças.



(b) Montagem final do suporte.

Figura 39 – Suporte para alocação dos componentes.

5 Resultados

Para a validação final do sistema, foi utilizado o suporte físico com os componentes já alocados, o que facilitou a utilização de um transferidor para validação visual da posição relativa do eixo do motor. A partir disso, foram definidas várias rotinas de execução em *software* que enviavam comandos de controle de posição e velocidade como descrito na Seção 4. Além da validação visual, foi realizada uma leitura constante dos valores de posição em *software* utilizando as funcionalidades da biblioteca desenvolvida. Essa leitura possibilitou uma análise mais precisa, pois foi possível estabelecer uma relação entre o ângulo no qual a referência se encontrava e a respectiva quantidade de pulsos lidos. A Figura 40 ilustra uma das rotinas de teste executadas.

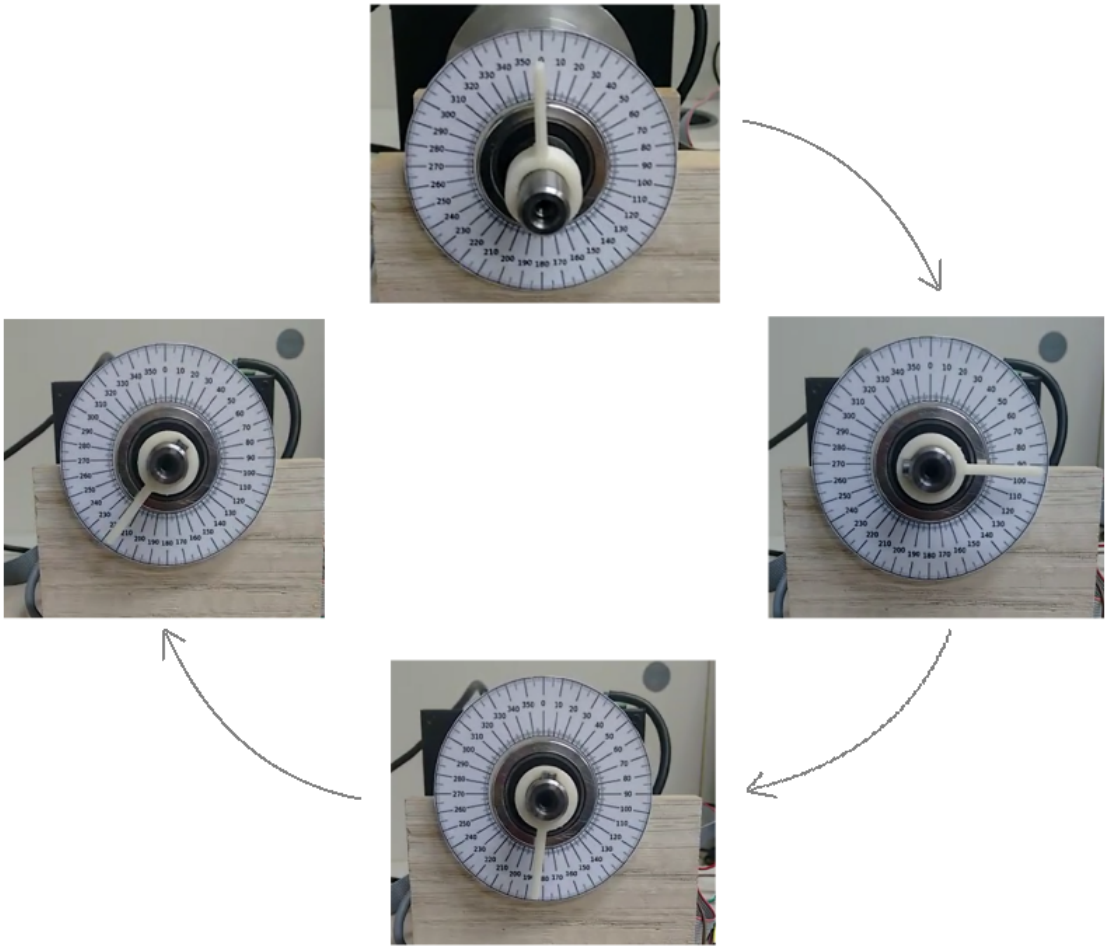


Figura 40 – Exemplo de rotina de teste executada para validação do sistema.

Nesta rotina específica, o valor de velocidade foi de 400 rpm, que de acordo com a Eq. 4.3 equivale a um *duty cycle* de 16,4 %. Os dados esperados e os dados adquiridos experimentalmente encontram-se descritos na Tabela 5.

Tabela 5 – Relação entre ângulo e quantidade de pulsos obtida teórica e experimentalmente.

Posição	**Valor teórico**	**Valor experimental**
0°	0	0
90°	320000	337070
180°	640000	674327
210°	746666,66	787198

A diferença existente entre os valores teóricos e experimentais é justificada pelo fato de não ter sido aplicado nenhum algoritmo de controle que considerasse o comportamento dinâmico do motor, ou seja, não foi levado em consideração o tempo de aceleração e frenagem do atuador.

A Tabela 6 mostra o relatório de utilização dos recursos do FPGA para a implementação dos subcomponentes do ModuloMotor anteriormente à inserção das configurações do barramento necessárias para estabelecer uma comunicação com o ARM. Na Tabela 7 encontra-se o resultado obtido após implementação do ModuloMotor, contendo todas as configurações necessárias para utilização do protocolo AMBA AXI disponível na plataforma de desenvolvimento Zybo.

Tabela 6 – Recursos de *hardware* utilizados para a implementação dos submódulos.

Componente	**FF**	**FF (%)**
Modulo_PWM	41	0.12
Modulo_Encoder	25	0.07
logica_combinacional	5	0.01

Tabela 7 – Recursos de *hardware* utilizados após implementação do ModuloMotor.

Componente	**FF**	**FF (%)**
ModuloMotor	782	2.22

Percebe-se que o número de *Flip Flops* (FFs) utilizados pelo FPGA aumentou consideravelmente com a adição das configurações do barramento. Isto ocorre devido aos vários sinais internos que são utilizados pelo sistema *on-chip* para controlar o acesso ao barramento AXI.

6 Conclusão

Neste trabalho, foi projetado e implementado um sistema de interface para acionamento de servo motores e leitura de *encoders* com um FPGA. Para o desenvolvimento desse sistema, foram realizadas implementações em *software* e em *hardware* utilizando uma plataforma *System on Chip* Zynq Zybo. Posteriormente, foi utilizado o barramento AMBA AXI disponível nesta plataforma para estabelecer uma comunicação entre *hardware* e *software*, que foi crucial para os testes do sistema, pois permitiu uma mudança rápida de parâmetros e a aplicação de rotinas de teste que conferiram agilidade ao desenvolvimento do projeto. Além de auxiliar na fase de testes, esta integração possibilitará a utilização do sistema de interface por controladores de posição do exoesqueleto que estejam embarcados em *software* ou em *hardware*.

As vantagens encontradas na utilização de uma plataforma *on-chip* para o desenvolvimento deste trabalho residem também no fato de que o barramento já fornecido pelo SoC confere maior agilidade na implementação de sistemas, além de uma utilização otimizada dos recursos de *hardware*. Caso fosse utilizada alguma plataforma que possui um processador, mas que não possui recursos de *hardware* reconfigurável, seria necessário embarcar um sistema operacional de tempo real para que fosse possível realizar o devido controle do servo motor, já que um sistema operacional comum não possui os requisitos de restrição de tempo provenientes dos RTOSs. Isto ocorre porque, apesar de o servo motor utilizado neste trabalho não necessitar de um controle em tempo crítico, o cálculo de posição relativa e velocidade dependem de uma exatidão na contagem do tempo que não está associada aos sistemas operacionais comuns.

Além das característcas de tempo real apresentadas pelos FPGAs, estas plataformas favorecem o *design reuse*, muito importante para uma posterior utilização do sistema implementado neste trabalho. Estas caracteríticas, associadas ao fato de que posteriormente sistemas de controle do exoesqueleto serão embarcados em plataformas de *hardware* reconfigurável, favoreceram para a escolha de um SoC para o desenvolvimento da interface de controle proposta. De acordo com os resultados obtidos, o sistema pode ser utilizado no contexto para o qual foi projetado, entretanto, algumas melhorias podem ser implementadas em trabalhos futuros. Como citado no Capítulo 5, não foi desenvolvido neste trabalho nenhum tipo de algoritmo de controle que considerasse as características dinâmicas do servo motor utilizado. Além disso, foi construído apenas um suporte físico para testes, devido ao tempo disponível. Dessa forma, sugere-se que em trabalhos futuros o sistema possa ser devidamente validado por meio de uma interface com sistemas de controle embarcados tanto em *software* (SW), quanto em *hardware* (HW).

Como descrito no Capítulo 4, apesar de a comunicação entre HW/SW ser facilitada com a utilização de DMA, este tipo de conexão apresenta algumas limitações, como o fato de não possuir proteção contra acesso simultâneo. Como forma de garantir maior segurança na utilização do barramento, sugere-se a implementação de um *driver* de dispositivo que realize a comunicação entre HW/SW, devido às vantagens que este tipo de comunicação possui. Para o contexto do exoesqueleto, a plataforma *on-chip* deve apresentar autonomia para que possa ser acoplada à estrutura móvel do equipamento. Dessa forma, sugere-se que seja implementado em trabalhos posteriores um mecanismo para que o sistema possa ser alimentado por meio de baterias.

Referências

ANDERSSON, S. *Userspace access via /dev/mem*. 2011. Disponível em: <http://svenand.blogdrive.com/archive/150.html#.WiWsx_bJ08p>. Citado na página 51.

ARAÚJO, M. V. de. *Desenvolvimento de Uma Órtese Ativa Para os Membros Inferiores Com Sistema Eletrônico Embarcado*. 2010. 136 p. Tese (Dissertação em Engenharia Eletrônica), UFRN (Universidade Federal do Rio Grande do Norte). Citado na página 22.

ARM. *AMBA ™ Specification(Rev. 2.0)*. 1999. Citado na página 27.

ARM. *AMBA AXI and ACE Protocol Specification*. 2011. 1–306 p. Disponível em: <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022d/index.html>. Citado 3 vezes nas páginas 9, 27 e 28.

BRAGA, N. *Como funcionam os encoders (MEC128)*. 2017. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/5454-mec128>. Citado 3 vezes nas páginas 9, 29 e 30.

CONDIE, E.; CAMPBELL, J.; MARTINA, J. *Consensus Conference On The Orthotic Management Of Stroke Patients, Netherlands, 2003.pdf*. Ellecom, Netherlands, 2003. 272 p. Citado na página 17.

CROCKETT, L. et al. *The Zynq Book*. 1. ed. Glasgow, Scotland: [s.n.], 2014. 484 p. Citado 3 vezes nas páginas 9, 24 e 25.

DIGILENT. *Embedded Linux ® Hands-on Tutorial for the ZYBO ™*. 2014. 1–37 p. Citado 3 vezes nas páginas 10, 47 e 49.

DIGILENT. ZYBO™ FPGA Board Reference Manual. p. 1–26, 2016. Citado 2 vezes nas páginas 9 e 35.

FARNELL. *Motor Brushless Maxon EC90 flat 323772*. Acessado em 28 de maio de 2017. Disponível em: <http://uk.farnell.com/maxon-motor/323772/motor-brushless-dc-90w-24v-3190rpm/dp/1909092>. Citado 2 vezes nas páginas 9 e 36.

Gurley Precision Instruments. Understanding Quadrature. 2009. Disponível em: <http://www.gurley.com/Encoders/Understanding_Quadrature.pdf>. Citado na página 30.

HASSAN, M. et al. Wearable gait measurement system with an instrumented cane for exoskeleton control. *Sensors (Basel, Switzerland)*, v. 14, n. 1, p. 1705–1722, 2014. Citado 3 vezes nas páginas 9, 22 e 23.

HSU, J. D.; MICHAEL, J. W.; FISK, J. R. *AAOS Atlas of Orthoses and Assistive Devices*. 4ª ed.. ed. [S.l.]: Mosby, 2008. Citado 2 vezes nas páginas 17 e 22.

IBGE. Censo Demográfico 2000: características gerais da população - resultados da amostra. p. 173, 2000. Citado na página 17.

IBGE. CENSO DEMOGRÁFICO 2010. Características gerais da população, religião e pessoas com deficiência. p. 211, 2012. Citado na página 17.

INSTRUCTABLES. *A Beginners Guide to Programmable Logic Devices*. 2010. Disponível em: <http://www.instructables.com/id/A-Beginners-Guide-to-Programmable-Logic-Devices/>. Citado 2 vezes nas páginas 9 e 26.

KAKURAI, S.; AKAI, M. Clinical experiences with a convertible thermoplastic knee-ankle-foot orthosis for post-stroke hemiplegic patients. *Prosthetics and orthotics international*, v. 20, n. 3, p. 191–194, 1996. Citado na página 17.

KAMINSKI, S. Maxon Motor & Motor Controller Manual. 2017. Disponível em: <http://edge.rit.edu/edge/P17105/public/Subsystem_Build_Test_Documents/Motor_Guide.pdf>. Citado 2 vezes nas páginas 9 e 39.

KAWAMOTO, H. et al. *Development of single leg version of HAL for hemiplegia*. 2009. 5038–5043 p. Citado na página 17.

KIM, H. et al. Locomotion control strategy of hydraulic lower extremity exoskeleton robot. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, v. 2015-August, p. 577–582, 2015. Citado na página 22.

MASSOCO, D. Z. d. S.; LUCINIO, L. A.; SANTOS, R. M. dos. Hemiplegia : uma revisão bibliográfica. *III Encontro Científico do GEPro, Grupo de Estudo de Produção*, 2013. Disponível em: <http://geprofatecjahu.com.br/anais/2013/24.pdf>. Citado na página 17.

MAXON. *Maxon Fl At Motor*. [S.l.], 2013. 360 p. Disponível em: <http://www.farnell.com/datasheets/1792645.pdf?_ga=2.47604718.898815476.1496528828-180704932.1490840510>. Citado 2 vezes nas páginas 11 e 36.

MAXON MOTOR. *maxon motor - Online Shop*. Acessado em 28 de maio de 2017. Disponível em: <http://www.maxonmotor.com/maxon/view/product/gear/planetary/gp62/110505?etcc_cu=onsite&etcc_med=Header%20Suche&etcc_cmp=mit%20Ergebnis&etcc_ctv=Layer&query=110505>. Citado 2 vezes nas páginas 9 e 37.

MAXON MOTOR. *Planetary Gearhead GP 81 mm, 20–120 Nm*. [S.l.], 2012. 2012 p. Disponível em: <http://www.maxonmotor.com/medias/sys_master/root/8821070135326/16-356-EN.pdf>. Citado 2 vezes nas páginas 11 e 37.

MAXON MOTOR. *Escon 70/10*. [S.l.], 2017. Citado 2 vezes nas páginas 9 e 37.

MICROCHIP. Section 15. Quadratrure Encoder Interface (QEI). 2014. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/70208C.pdf>. Citado 2 vezes nas páginas 9 e 31.

MORETON, P. *Industrial Brushless Servomotors*. 1. ed. Newnes, 2000. 186 p. Disponível em: <http://www.sciencedirect.com/science/book/9780750639316>. Citado 2 vezes nas páginas 9 e 29.

NATIONAL INSTRUMENTS. *Introducao a tecnologia FPGA - National Instruments*. 2016. Acessado em 15 de abril de 2017. Disponível em: <http://www.ni.com/white-paper/6984/pt/>. Citado na página 26.

NUNEN, M. P. M. van et al. Recovery of walking ability using a robotic device in subacute stroke patients: a randomized controlled study. *Disability and Rehabilitation: Assistive Technology*, Amsterdam, v. 10, n. 2, p. 141–148, 2015. Citado na página 17.

OPENCORES. Quadrature Decoder (for optical Encoders). 2009. Disponível em: <https://opencores.org/project,quad_decoder>. Citado na página 43.

PUTRA, A. P. et al. System-on-Chip Architecture for High-Speed Data Acquisition in Visible Light Communication System. Bandung, Indonesia, p. 63–67, 2016. Citado 2 vezes nas páginas 23 e 24.

SUH, S.-H. et al. *Theory and Design of CNC Systems*. 1. ed. [S.l.]: Springer-Verlag London, 2008. Citado 2 vezes nas páginas 9 e 31.

TRIMBERGER, S. M. Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology. *Proceedings of the IEEE*, v. 103, n. 3, p. 318–331, 2015. Citado na página 24.

VÕSANDI, L. *AXI Direct Memory Access*. 2014. Disponível em: <https://lauri.vosandi.com/hdl/zynq/xilinx-dma.html>. Citado 2 vezes nas páginas 10 e 50.

WHITE, H. et al. Clinically prescribed orthoses demonstrate an increase in velocity of gait in children with cerebral palsy: a retrospective study. *Developmental medicine and child neurology*, v. 44, n. 4, p. 227–232, 2002. Citado na página 17.

XILINX. *FPGA vs. ASIC*. 2012. Acessado em 29 de abril de 2017. Disponível em: <https://www.xilinx.com/fpga/asic.htm>. Citado na página 26.

XILINX. *Xilinx® : What is an FPGA? Field Programmable Gate Array*. 2012. Disponível em: <https://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm>. Citado na página 25.

YANG, S. M.; KE, S. J. Performance evaluation of a velocity observer for accurate velocity estimation of servo motor drives. *IEEE Transactions on Industry Applications*, v. 36, n. 1, p. 98–104, 2000. Citado na página 28.

ZETTLEX. A Comparison Between Optical Encoders and Inductive Encoders. p. 1–8, 2017. Disponível em: <http://www.azosensors.com/article.aspx?ArticleID=744>. Citado na página 31.

ZHU, M.; WANG, X.; XU, M. The analysis for system-on chip application on full authority digital engine control system. *2011 International Conference on Electric Information and Control Engineering, ICEICE 2011 - Proceedings*, p. 2728–2732, 2011. Citado na página 23.

ZHU, Z. et al. Design of a Wearable Lower Limb Exoskeleton for Paralyzed Individuals. *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, p. 3, 2016. Citado 3 vezes nas páginas 9, 22 e 23.

Apêndices

APÊNDICE A – Primeiro Apêndice

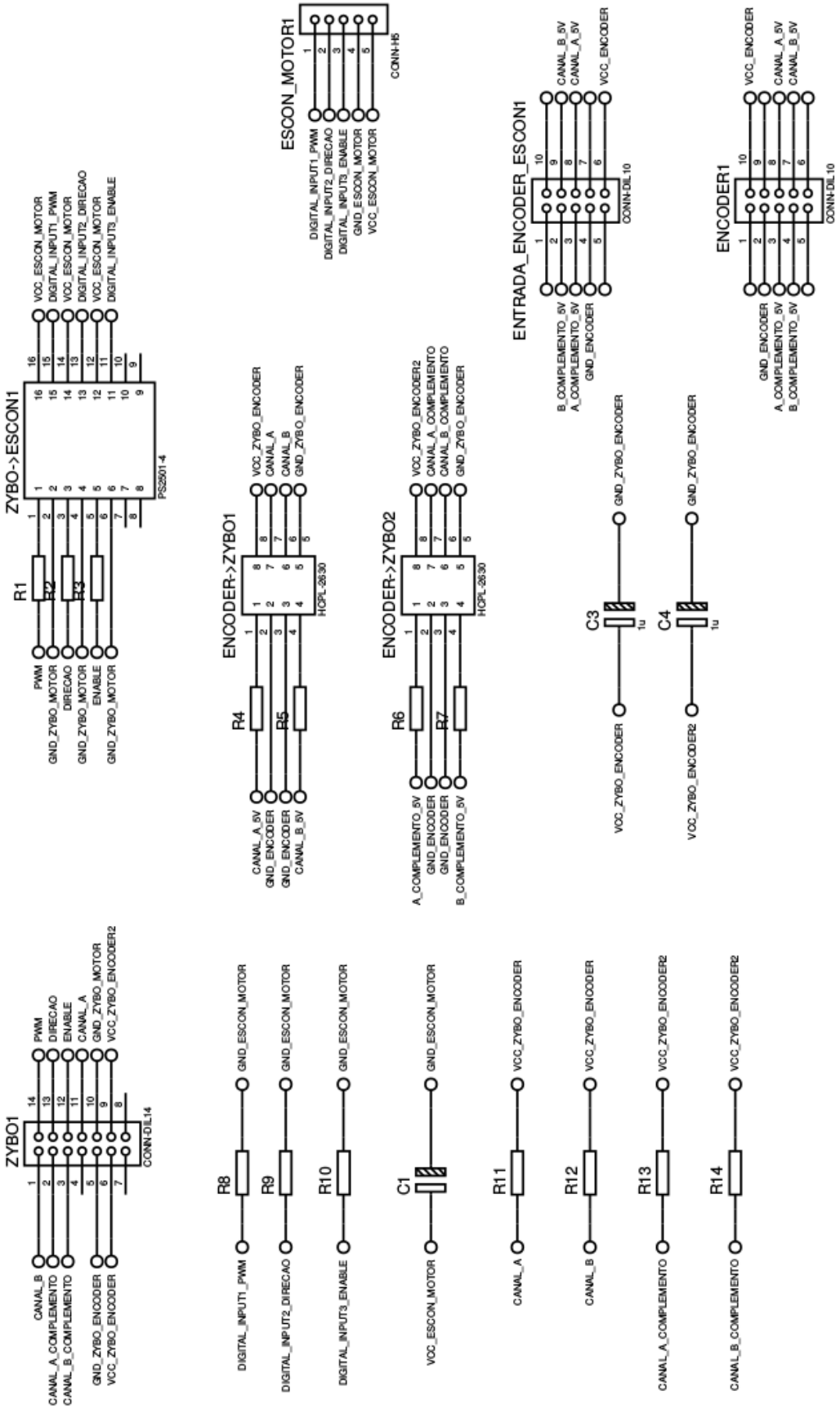


Figura 41 – Esquemático dos circuitos de acoplamento.

Layout das trilhas da segunda versão da PCB.

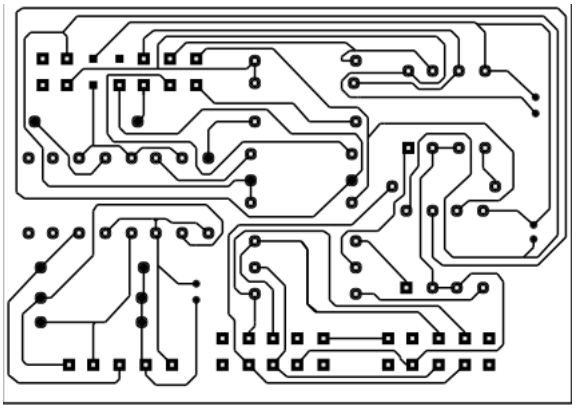


Figura 42 – *Layout* da segunda versão da PCB.

APÊNDICE B – Segundo Apêndice

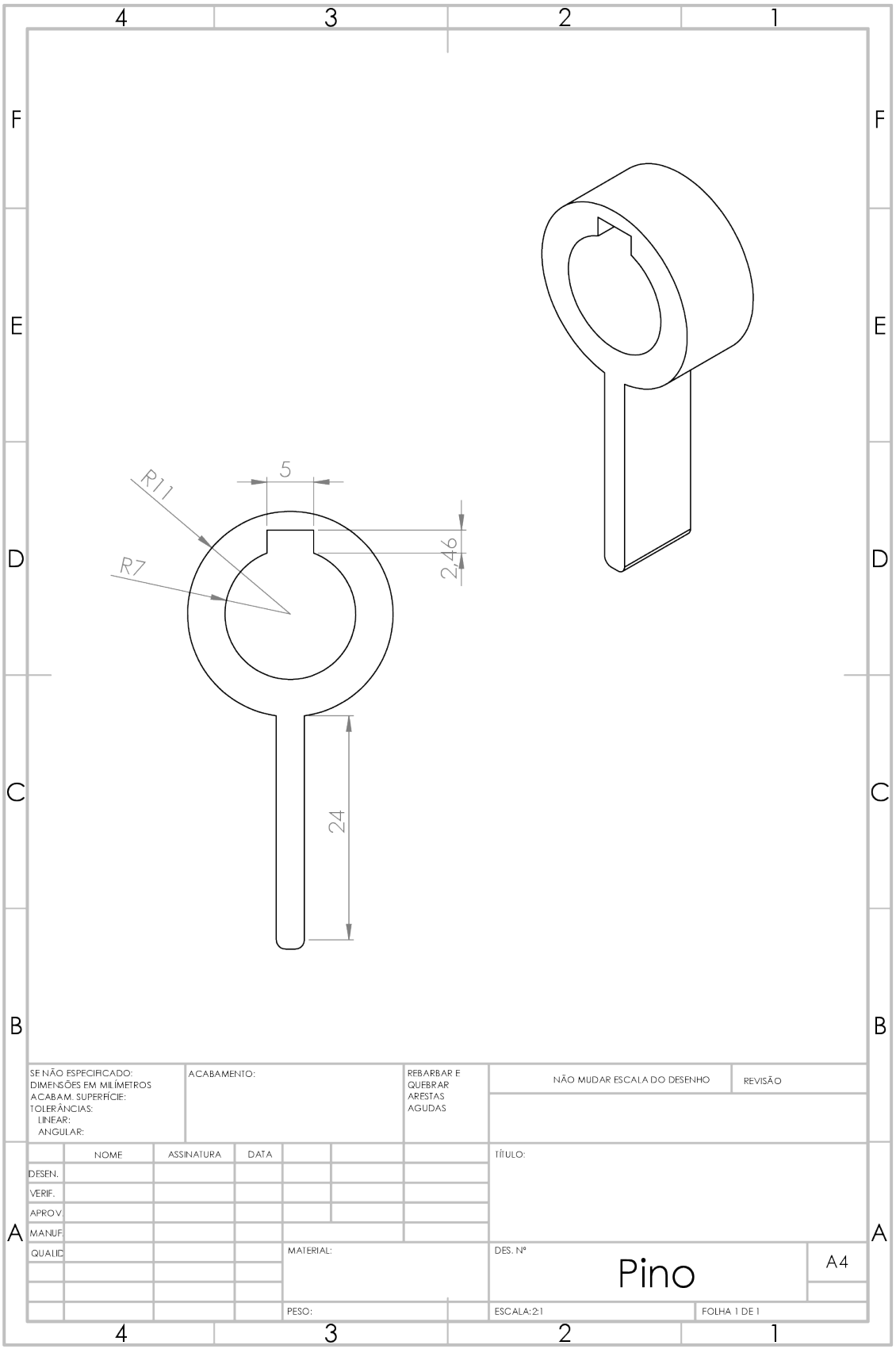


Figura 43 – Referência para transferidor.

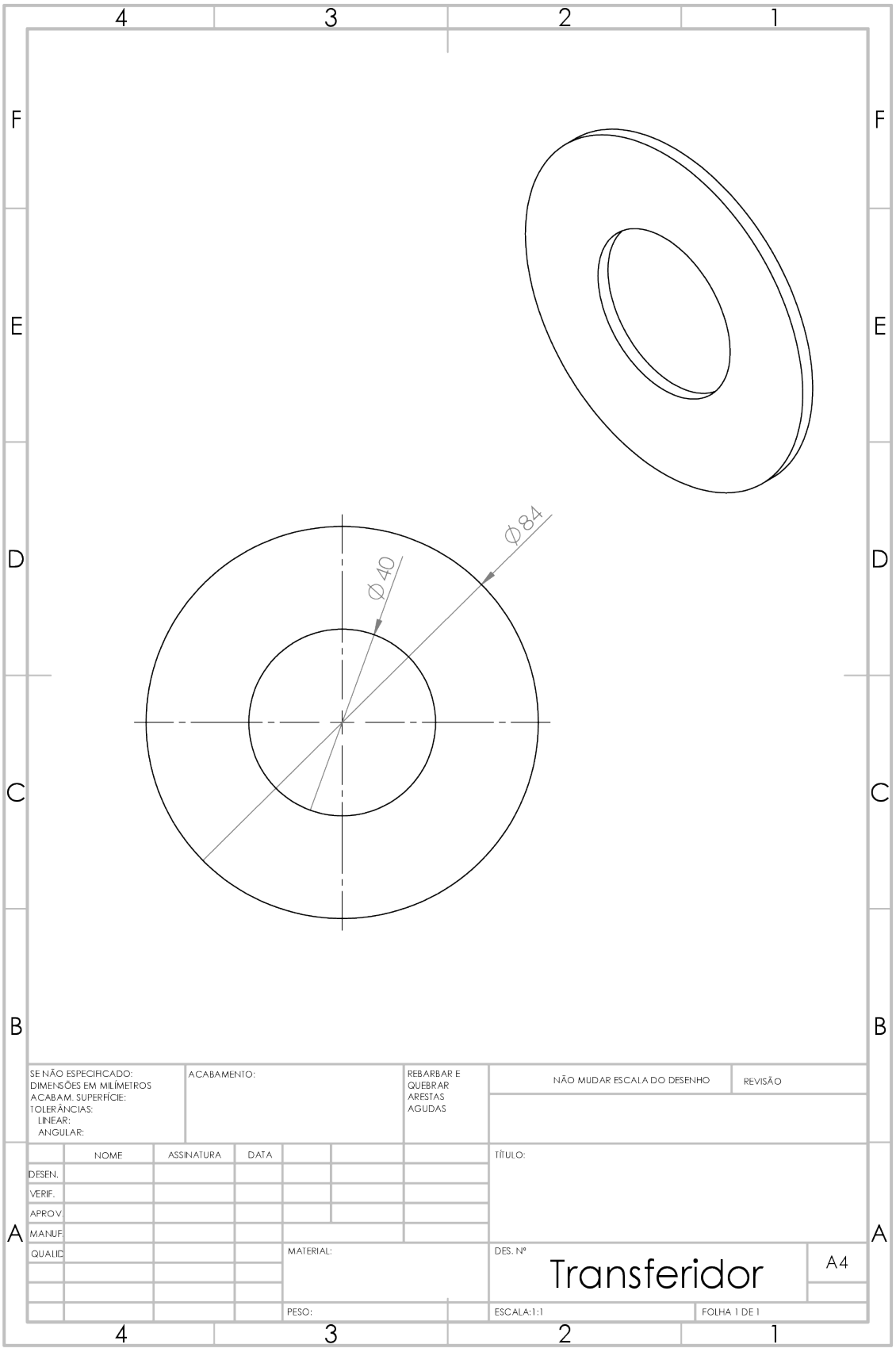


Figura 44 – Molde para fixação do transferidor.