UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ANA LUIZA GRACIANO BOSSOLANI BUCK

CONSTRUÇÃO DE ROBÔ HEXÁPODE COM CÓDIGO DE PROGRAMAÇÃO IMPLEMENTADO EM FPGA

MEDIANEIRA

2022

ANA LUIZA GRACIANO BOSSOLANI BUCK

CONSTRUÇÃO DE ROBÔ HEXÁPODE COM CÓDIGO DE PROGRAMAÇÃO IMPLEMENTADO EM FPGA

Construction of a hexapod robot with programming code implemented in FPGA

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Alberto Noboru Miyadaira.

Coorientador: Fernando Schutz.

MEDIANEIRA

2022

ANA LUIZA GRACIANO BOSSOLANI BUCK

CONSTRUÇÃO DE ROBÔ HEXÁPODE COM CÓDIGO DE PROGRAMAÇÃO IMPLEMENTADO EM FPGA

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 24/novembro/2022

_____

Alberto Noboru Miyadaira
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Fernando Schutz
Doutorado
Universidade Tecnológica Federal do Paraná

_____

Alex Lemes Guedes
Mestrado
Universidade Tecnológica Federal do Paraná

_____

Filipe Marangoni
Doutorado
Universidade Tecnológica Federal do Paraná

MEDIANEIRA

2022

AGRADECIMENTOS

Agradeço ao meu orientador, prof. Dr. Alberto Noboru Miyadaira, por sua ajuda, seus conselhos e paciência durante a orientação deste trabalho, assim como por ter me falado sobre o sistema FPGA pelo qual me facinou desde o princípio, e então me propiciado a elaboração deste projeto.

Agradeço também ao prof. Dr. Fernando Schutz pela coorientação e por todo suporte e atenção, principalmente às etapas do projeto que tangem a implementação, programação em descrição de hardware e FPGAs. Além disso, por ter me proporcionado a oportunidade de ingressar na Incubadora Tecnológica.

Agradeço aos Departamentos de Elétrica e de Computação pelos equipamentos, tempo e ajuda cedidos.

À minha família e amigos por prover todo o suporte e apoio durante minha graduação. E aos professores da Universidade, por todos os ensinamentos transmitidos em sala de aula, que acompanharam e me ajudaram durante esta jornada de graduação, sem à qual eu não teria conseguido.

RESUMO

A aplicação de robôs para realizar funções com segurança em locais de risco e difícil acesso estão cada vez mais comuns, onde requer-se estabilidade para realização da movimentação. Este trabalho teve como objetivo a construção de um robô hexápode biologicamente inspirado em formigas, com dois graus de liberdade em cada pata, e controle através de um computador e um dispositivo de lógica reprogramável. A linguagem de código de programação foi escrita em *SystemVerilog*, linguaguem de descrição e verificação de *hardware*, e implementado na placa *DE2-115*, cujo dispositivo FPGA é o *Cyclone IV EP4CE115F29C7N*. A tecnologia da capacidade da lógica reconfigurável, proporcionada pelo FPGA possibilita o sincronismo dos servomotores devido à flexibilidade da execução de comandos em paralelo, além de possuir uma reprogramação de forma rápida enquanto o sistema está *online*. Utilizou-se o total de doze servomotores, como atuadores, modelo MG996R, todos alimentados por uma fonte *SATE PRO-460* (em regime na linha de 5V), isolada do restante da placa. As partes que compõem a estrutura física do robô foram modeladas através do *software SolidWorks* e impressas em impressora 3D a fim de reduzir os custos e facilitar a reprodução de forma a permitir que o projeto possa ser utilizado em atividades de ensino e pesquisa. A linguagem de programação mostrou-se eficaz e altamente adaptável à outros possíveis projetos, de acordo com os módulos desenvolvidos de forma individualizada para controle dos movimentos.

Palavras-chave: robôs móveis; arranjos de lógica programável em campo; servomecanismos.

ABSTRACT

The application of robots to perform functions safely in risky and difficult to access places is becoming more and more common, where stability is required to execute the movement. This research aimed to build a hexapod robot biologically inspired by ants, with two degrees of freedom in each paw, and control through a computer and a reprogrammable logic device. The programming code language was written in *SystemVerilog*, *hardware* description and verification language, and implemented on the *DE2-115* board, whose FPGA device is the *Cyclone IV EP4CE115F29C7N* . The reconfigurable logic capability technology provided by the FPGA enables the synchronization of servomotors due to the flexibility of executing commands in parallel, in addition to having a quick reprogramming while the system is *online*. A total of twelve servomotors were used as actuators, model MG996R, all powered by a *SATE PRO-460* source (on the 5V line), isolated from the rest of the board. The parts that make up the physical structure of the robot were modeled using *SolidWorks software* and printed on a 3D printer in order to reduce costs and facilitate reproduction in order to allow the project to be used in teaching and research activities . The programming language proved to be effective and highly adaptable to other possible projects, according to the modules developed individually to control movements.

Keywords: mobile robots; field programmable gate arrays; servomechanisms.

LISTA DE ILUSTRAÇÕES

Figura 1 – Primeiro robô UNIMATE 13
Figura 2 – Robô inspirado em IA, Shaket 14
Figura 3 – Classificação segundo à anatomia 15
Figura 4 – Especificação da localização das juntas 16
Figura 5 – Hexápode *Hannibal* 17
Figura 6 – Robô SILO6 17
Figura 7 – Hexápode construído na Universidade São Judas Tadeu 18
Figura 8 – Kits de desenvolvimento FPGA: (a) Kit FPGA EE03, (b) Placa DE10-Pro . . 19
Figura 9 – Linguagens utilizadas para verificação (testbenchs) 20
Figura 10 – Estrutura do hexapóde elaborada no SolidWorks: (a) Vista superior, (b) Vista lateral 22
Figura 11 – Peças do hexapóde elaborada no *software SolidWorks* 22
Figura 12 – Servo Motor MG996R 23
Figura 13 – *Gráfico com informações técnicas do MG996R* 24
Figura 14 – SATE PRO-460 400W 24
Figura 15 – *Placa DE2-115 – Terasic* 25
Figura 16 – Recurso *Programming* disponível no *software Quartus Prime* para execução do programa 26
Figura 17 – Exemplo de sequência para programação dos movimentos 26
Figura 18 – Arranjo de pinos da Placa DE2-115 fornecido no Manual DE2-115 27
Figura 19 – Etapas da programação no *software Quartus Prime* 28
Figura 20 – Diagrama de bloco do código de programação 29
Figura 21 – Recorte dos módulos criados no software Quartus 30
Figura 22 – Parâmetros do módulo denominado "*controller*" 31
Figura 23 – Trecho do código de programação do módulo LUT e LUT_TURNLEFT 32
Figura 24 – RTL Viewer 33
Figura 25 – Peças fabricadas na impressora *3D FlyingBear Ghost 4S* 33
Figura 26 – (a)Partes um e dois da montagem da estrutura física; (b) Placa DE10-Pro . . 34
Figura 27 – Estrutura física finalizada 35
Figura 28 – Referêncial de posição e ângulo do servomotor 35

Figura 29 – Tabela de conversão para o duty cycle 36
Figura 30 – Tabela de instruções dos movimentos 36
Figura 31 – Bancada de testes: (a) conexão dos equipamentos, (b) conexão com o osciloscópio 38
Figura 32 – Resultados obtidos no osciloscópio:(a) sinais PWM sobrepostos, (b) sinais PWM separados 38
Figura 33 – Vista superior do robô hexápode implementado 39
Figura 34 – Implementação para testes: (a) passo 1: *reset* acionado, (b) passo 2: *rest* acionado, (c) passo 3: *forward* acionado, (d) passo 3: *forward* acionado . . . 39
Figura 35 – Vista superior dos movimentos da LUT_FORWARD: (a) instrução 1, (b) instrução 2 40
Figura 36 – Implementação final: (a) posição inicial, (b) momento um do movimento de avanço, (c) momento dois do movimento de avanço, (d) momento três do movimento de avanço 41

LISTA DE ABREVIATURAS E SIGLAS

AS	*Active Serial*.
FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
HDVL	*Hardware Description and Verification Language*
HVL	*Hardware Verification Language*
IEEE	Instituto de Engenheiros Elétricos e Eletrônicos.
IFR	Federação Internacional de Robótica.
JTAG	*Joint Test Action Group*
PWM	*Pulse Width Modulation*
RIA	*Robotics Industries Association*
RTL	*Register-Transfer Level*
SMA	*SubMiniature Version A*
SRI	Stanford Research Institute
SV	*SystemVerilog*
VHDL	*VHSIC Hardware Description Language*

SUMÁRIO

1 INTRODUÇÃO 11

1.1 Justificativa 11

1.2 Objetivos 12

1.2.1 Objetivo Geral 12

1.2.2 Objetivos Específicos 12

2 REVISÃO DE LITERATURA 13

2.1 Evolução da Robótica 13

2.2 Controle Através do FPGA 18

3 MATERIAIS E MÉTODOS 21

3.1 Componentes físicos 21

3.1.1 Modelagem da Estrutura Física 21

3.1.2 Atuador 22

3.1.3 Fonte de Alimentação 24

3.1.4 Controlador 24

3.2 Componentes Lógicos 26

3.2.1 Programação em SystemVerilog 27

4 RESULTADOS E DISCUSSÃO 29

4.1 Elaboração do código de programação 29

4.1.1 Visão macro do código de programação 29

4.1.2 Descrição dos módulos 31

4.2 Montagem da estrutura física do robô hexápode 33

4.2.1 Calibração dos duty cycles dos servomotores 35

4.2.2 Implementação e testes 37

5 CONCLUSÕES E PERSPECTIVAS 42

REFERÊNCIAS 44

APÊNDICE A CUSTOS ENVOLVIDOS NO PROJETO 47

1 INTRODUÇÃO

Uma área em grande crescimento atualmente é a robótica móvel. De acordo com o relatório *World Robotics*, elaborado pela Federação Internacional de Robótica. (IFR), no ano de 2018 a venda de robôs industriais chegou a 16,5 bilhões de dólares. A *Zion Market Research* estima que o mercado de robótica industrial chegará a US$ 62 bilhões de receita até 2024.

Os robôs estão difundidos em vários mercados, e em constante evolução, o que nos instiga cada vez mais à aplicar suas vantagens para realizar tanto tarefas do dia-a-dia, como robôs de limpeza, quanto operações mais complexas, como a exploração planetária. Cada funcionalidade possui adequações específicas que são mais eficientes para a finalidade pela qual foi projetado, sendo estas os tipos de controle, sensores, atuadores e processadores.

O planeta Terra possui os mais diversos tipos de ambientes, sendo eles hostis, irregulares e inóspitos, como zonas de guerra, exploração de montanhas, análises em áreas radioativas, respectivamente. Estes meios requerem mecanismos de locomoção adaptáveis para cada tipo de terreno. Nesses casos, geralmente torna-se complexo para um operador humano controlar os movimentos de um robô diretamente, portanto torna-se fundamental que eles sejam pré-programados.

Os mecanismos de locomoção dos robôs com pernas são frequentemente inspirados em sistemas biológicos, que são muito bem-sucedidos em se mover por uma ampla área de ambientes (BÖTTCHER, 2006). Em especial, a configuração de seis pernas com três graus de liberdade permite uma mobilidade eficiente para uma aplicação flexível, podendo ser reconfigurada de acordo com a necessidade do projeto.

A reconfiguração da lógica é possível devido ao controle com *Field-Programmable Gate Array* (FPGA), que possui a vantagem de executar rapidamente comandos simultâneos de forma paralela, além de serem programados enquanto estão funcionando, pois o tempo de reprogramação é da ordem de microssegundos. Este curto espaço de tempo significa que o sistema nem notará que o *chip* foi reprogramado e pode haver um curto período de espera, onde o sistema não precisará ser desligado (BANJANOVIC-MEHMEDOVIC *et al.*, 2019).

A construção de um robô hexápode controlado por FPGA é o foco deste trabalho, atendendo a necessidade de acesso e navegação a diversos tipos de ambientes, tendo em vista a versatilidade da reprogramação para suprir diferentes aplicações sem a necessidade de interromper a operação do robô por longos períodos de tempo para a atualização do programa.

1.1 Justificativa

Os robôs hexápodes são eficientes em se locomoverem em ambientes perigosos e irregulares, onde é preciso manter a estabilidade, além de conseguir se deslocar em várias direções, portanto é um robô que possui uma grande quantidade de aplicações, tornando um projeto expansível aos mais diversos setores e funções. A motivação principal para o projeto é o

interesse em estudar o comportamento de controle com um dispositivo de lógica programável, o FPGA, tendo em vista os amplos recursos oferecidos pela lógica reconfigurável, principalmente a forma de reprogramar o robô tornando-o flexível sem precisar alterar seu *design*.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral a construção de um robô hexápode, com controle através de um dispositivo de lógica reprogramável com a linguagem de código de programação em *SystemVerilog*, explorando a capacidade da lógica reconfigurável, proporcionada pelo FPGA.

1.2.2 Objetivos Específicos

Os objetivos específicos propostos são:

i) Estudar a linguagem de programação SystemVerilog;

ii) Modelar em 3D a estrutura do robô hexápode;

iii) Montagem da estrutura física do robô hexápode;

iv) Planejar e executar os movimentos de avanço e de recuo do robô hexápode;

v) Elaborar e implementar o código de programação para o dispositivo FPGA;

vi) Avaliar o desempenho dos movimentos executados pelo robô.

2 REVISÃO DE LITERATURA

Neste Capítulo será apresentada uma breve evolução da robótica ao longo do tempo, com enfoque no sistema de locomoção de seis patas, ou seja, os hexápodes, e as principais vantagens da forma de controle com o FPGA.

2.1 Evolução da Robótica

Há muitas definições acerca do significado da palavra robô, e principalmente de sua origem. Os primeiros modelos de robôs, segundo Pieri (2002), surgiram na civilização grega, e imitavam os movimentos de seres humanos e animais. A partir do século IV a.C, as marionetes eram acionadas por sistemas de polias e pesos. A palavra robô tem sua origem na palavra tcheca *robota*, que significa trabalho forçado, e foi utilizada pela primeira vez em 1921, pelo dramaturgo Karel Capeck no conto de ficção *R.U.R (Rossum's Universal Robots)*.

A definição de um robô segundo o *Robotics Industries Association* (RIA), é mais abrangente e diz que um robô é um manipulador reprogramável e multifuncional projetado para mover materiais, partes, ferramentas ou dispositivos especializados através de movimentos variáveis programados para desempenhar uma variedade de tarefas (FERREIRA, 1991; BIANCHI, 2011). Outra definição, segundo Mataric *et al.* (2014, p. 19) "Um robô é um sistema autônomo que existe no mundo físico, pode sentir o seu ambiente e pode agir sobre ele para alcançar alguns objetivos". Sendo esta uma definição mais restritiva acerca do termo.

Segundo Pieri (2002), após a invenção do transistor em 1948, os robôs passaram a ser controlados por computadores. A primeira patente para um robô industrial controlado por computador foi registrada em 1954 por George Devol, que criou uma memória computadorizada e um sistema de controle chamado *Universal Automation*. Mais tarde, co-fundou a companhia de robôs industriais - *UNIMATION*. Em 1961 o primeiro robô *UNIMATE*, que usava comando numérico programável, foi instalado na linha de montagem da *General Motors*, dando inicio a era da automação industrial. A Figura 1 apresenta o robô citado acima.

Figura 1 – Primeiro robô UNIMATE

Fonte: Robotic Industries Association (2020).

Ainda assim, apesar do sucesso, esses robôs industriais tinham a desvantagem de possuir uma mobilidade restrita. Um manipulador fixo, como o da Figura 1, tem uma amplitude limitada de movimento que depende de onde ele está afixado. Portanto, um robô com capacidade de se movimentar seria capaz de deslocar-se por toda a indústria, sendo capaz de desempenhar outras funções com eficiência.

Uma definição genérica sobre o termo robô móvel, segundo Marchi (2001), um robô móvel é um dispositivo mecânico montado sobre uma base não fixa, que age sob o controle de um sistema computacional, equipado com sensores e atuadores que o permitem interagir com o ambiente. Os robôs móveis surgiram em 1968 carregando conceitos da mecânica e da robótica fixa.

A princípio, com o avanço nas áreas de sensores, processamento de imagens e inteligência artificial, dotar um robô móvel com capacidades para atuar em ambientes dinâmicos parecia ser algo simples, porém, logo percebeu-se a grande complexidade envolvida no desenvolvimento de sistemas móveis que fossem robustos e adaptáveis. Em 1969, Nilsson descreve o primeiro sistema robótico móvel que utiliza *quadtrees* (estrutura de dados em forma de árvore que é gerada pela decomposição de uma imagem bidimensional, codificando-a em quatro quadrantes) para representar o ambiente e grafos de visibilidade para o planejamento da trajetória.

O Shaket, apresentado na Figura 2 é um bom exemplo dos primeiros robôs inspirados em inteligência artificial, construído no Stanford Research Institute (SRI) por Charles Rosen (1917–2002) e sua equipe, no final dos anos 1960, possuía sensores de contato e uma câmera (MATARIC *et al.*, 2014; PIERI, 2002).

Figura 2 – Robô inspirado em IA, Shaket

Fonte: SRI International (2007).

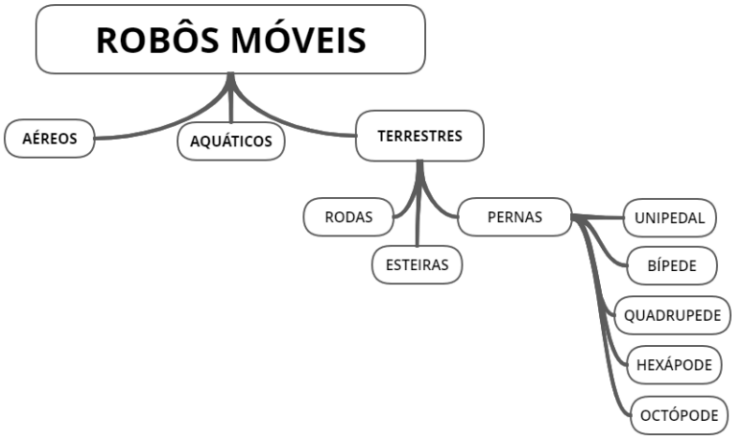
No entanto, esses robôs ainda eram lentos para processar as informações e consequentemente incapazes de se mover de forma satisfatória. Segundo Mataric *et al.* (2014), os robôs

do início dos anos 1970 e 1980 forneceram lições importantes, e na maioria dos casos, as lições tinham a ver com a necessidade de se mover mais rápido, de maneira mais robusta, e de pensar de forma a permitir tal ação. Em resposta a esses problemas, nos anos 1980, a robótica entrou em uma fase de desenvolvimento acelerado. Nessa década, novas perspectivas surgiram, e finalmente, se organizaram nos tipos de controle de rôbos que usamos nos dias atuais.

Um robô móvel pode ser categorizado de diversas formas, tendo em vista a grande diversidade de robôs móveis, que podem ser classificados, segundo Pieri (2002), de acordo com três aspectos: anatomia, tipo de controle e funcionalidade.

A anotomia pode ainda ser classificada em subgrupos, sendo: robôs aéreos, robôs aquáticos e robôs terrestres. Há uma grande variedade de maneiras possíveis de se mover e, portanto, a seleção da abordagem de um robô para a locomoção é um aspecto importante do design do robô móvel (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). A Figura 3 apresenta o diagrama de classificação de robôs segundo a anatomia.

Figura 3 – Classificação segundo à anatomia



Fonte: Adaptado de Siegwart, Nourbakhsh e Scaramuzza (2011).

Os sistemas biológicos conseguem se mover através de uma ampla variedade de ambientes hostis. Portanto, pode ser desejável copiar sua seleção de mecanismos de locomoção. Segundo Demasi (2012 apud BEKKER, 1961, p. 18), “há muito que se discutem as vantagens e desvantagens de sistemas movidos a rodas e pernas. Já na década de 60, Bekker questionava o futuro dos sistemas movidos a rodas apresentando algumas vantagens da locomoção por pernas baseado em estudos feitos com animais.” Atualmente, os robôs que se assemelham a sistemas biológicos de alguma forma, são denominados de robôs biomiméticos.

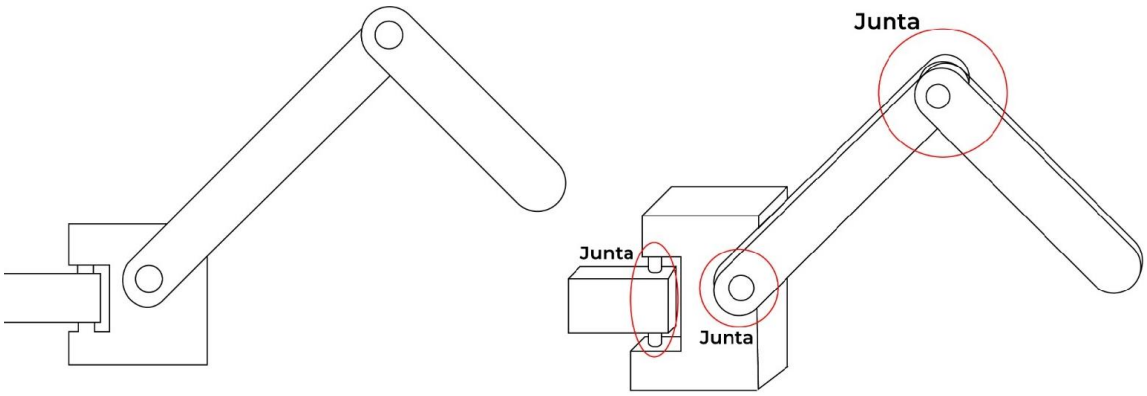
Os robôs móveis com patas podem superar obstáculos efetuando pontos de contato com o solo, atravessando descontinuidades nos terrenos, de acordo com a dimensão do robô e do obstáculo, podendo permanecer nivelados e estáveis. Portanto o sistema de locomoção de animais e insetos são mais versáteis e adaptados aos vários tipos de ambiente. No entanto, as pesquisas não estão restritas a representar o desenho estrutural desses sistemas biomecânicos em robôs, mas também aproximar, o quanto for possível, a estrutura operacional que controla a forma de locomoção deles (RABELO, 2015).

Segundo Erden (2006), a locomoção por patas requer um sistema de controle mais complexo, além de possuir muitos atuadores em comparação as demais configurações, sendo necessário coordená-los continuamente, com isso o consumo de energia se torna alto, pois em um determinado momento da caminhada, algumas patas estão em movimento, enquanto outras estão fazendo o papel de suporte da base, o que reduz também a velocidade do robô.

As configurações de seis pernas têm sido extremamente populares na robótica móvel por causa da sua estabilidade estática durante a caminhada, reduzindo a complexidade do controle. Na maioria dos casos, cada perna tem três graus de liberdade, incluindo flexão do quadril, flexão do joelho e abdução do quadril. Esse tipo de locomoção se caracteriza por possuir um ou mais contatos pontuais entre o robô e o solo, utilizando as patas como suspensão, e tem como principal vantagem adaptabilidade e capacidade de fazer manobras em relevos acidentados, não importando a qualidade deste (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Um dos parâmetros mais importantes na modelagem do robô são os graus de liberdade, que determinam os movimentos que ele será capaz de realizar, estando associado ao número de variáveis posicionais que permitem definir a posição de todas as partes. No caso de três graus de liberdade, por exemplo, o robô consegue se mover no espaço de três dimensões ao todo. Cada grau de liberdade é representado por uma junta. A Figura 4 ilustra a localização das juntas, sendo três por pata, que permitem que ele se mova em várias posições.

Figura 4 – Especificação da localização das juntas



Fonte: Autoria Própria (2021).

Em 1961, os projetistas da *Space General Corp* desenvolveram um protótipo de hexápode, o Robô Lunar Caminhante, era um veículo com seis patas e dois braços, corpo triangular, câmera, painel solar e garra mecânica para exploração do solo lunar. O protótipo acabou não sendo utilizado para exploração lunar, porém serviu de base para as pesquisas de Morrison, que em 1968 construiu um robô de seis patas que podia caminhar alternando três patas por passo. Com a evolução dos sistemas de controle, os robôs consequentemente foram progredindo, e entre 1989 e 1990, desenvolvido pelo Laboratório *Mobot*, o robô autônomo *Hannibal* foi criado, com o intuito de servir de plataforma experimental para exploração planetária, dispondo

de 3 atuadores em cada pata, 60 entradas sensoriais e 8 microprocessadores (RABELO, 2015), conforme apresentado na Figura 5.

Figura 5 – Hexápode *Hannibal*

Fonte: MIT (1990).

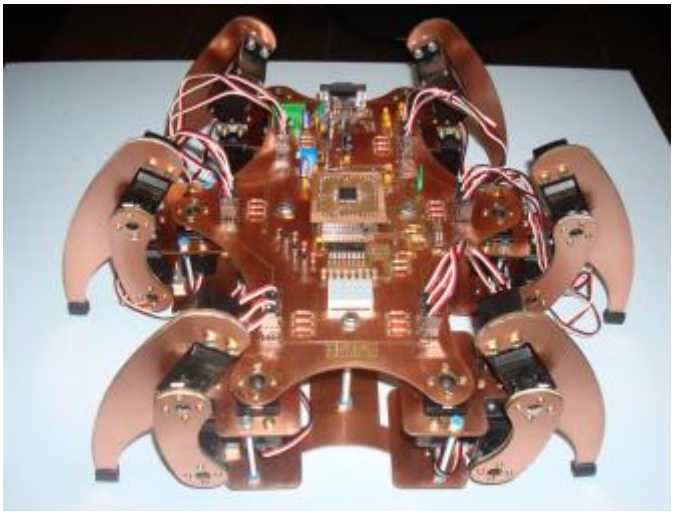
Um exemplo mais atual de hexápode, é o *SILO6*, criado pelo Instituto de Automação Industrial da Espanha em 2004, inicialmente com o objetivo principal de usá-lo como uma plataforma para pesquisas e estudos robóticos, porém foi utilizado com sucesso na detecção de minas terrestres. O projeto foi pensado como uma iniciativa para resolver um problema mundial causado pela remoção de minas terrestres que foram implantadas há várias décadas. Composto por cinco subsistemas principais: cabeça do sensor, manipulador, localizador, o próprio robô ambulante e o controlador, sendo o sensor composto por um conjunto de detecção de minas (ROBOTICSTODAY, 2018), conforme ilustração da Figura 6.

Figura 6 – Robô SILO6

Fonte: Instituto de Automática Industrial (2004).

No Brasil, um robô hexápode foi premiado no Congresso Internacional da Instituto de Engenheiros Elétricos e Eletrônicos. (IEEE), criado em 2008, de forma experimental, possui três graus de liberdade, funciona por controle e tem uma ligação com uma fonte de alimentação externa à estrutura (RABELO, 2015), conforme apresentado na Figura 7.

Figura 7 – Hexápode construído na Universidade São Judas Tadeu



Fonte: Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) (2010).

2.2 Controle Através do FPGA

As principais preocupações na última década em sistemas automatizados em tempo real tem sido o projeto de arquiteturas de *hardware (Systems-on-Chip)*, bem como o desenvolvimento de algoritmos viáveis para execução em plataformas de *hardware*, como um campo programável *Gate Array (FPGA)*.

O FPGA permitiu aos desenvolvedores criar projetos grandes, testá-los, e fazer modificações com facilidade e rapidez, acarretando flexibilidade no projeto do circuito, além da execução paralela de comandos ser mais eficiente quando comparado ao sistema convencional baseado em microcontroladores (BANJANOVIC-MEHMEDOVIC *et al.*, 2019).

A implementação do controle através do FPGA tem uma certa superioridade sobre um microprocessador, segundo Reichenbach *et al.* (2011, p. 17),

> O FPGA tem a capacidade de ser reprogramado em tempo real, o que significa que não há necessidade de nenhum tempo de inatividade para o controlador, os FPGAs que estão no mercado atualmente suportarão hardware com mais de 1 milhão de portas, além de operar mais rápido do que um microprocessador.

A forma de programar é descrita a nível de *hardware* e não de *software*, portanto as *Hardware Description Language* (HDL), ou linguagens de descrição de *hardware*, foram desenvolvidas para descrever como o *hardware* se comporta. As linguagens tradicionais são um processo sequencial, enquanto a HDL é um processo paralelo. As duas formas mais comuns usadas no desenvolvimento de projetos em FPGA são:

i) *VHSIC Hardware Description Language* (VHDL) direto ou codificação *Verilog* (Linguagem de descrição de *hardware* de circuito integrado de velocidade muito alta);

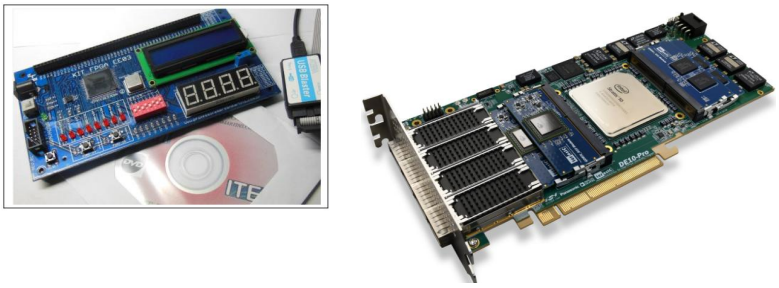
ii) *SystemVerilog* considerada uma *Hardware Description and Verification Language* (HDVL), que combina os recursos da linguagem de programação *Verilog* e VHDL com o da linguagem de verificação de hardware *VERA*, bem como C++ (SUTHERLAND; MILLS, 2003).;

iii) Geradores de código HDL específicos. Embora o HDL *Code Generator* existente (por exemplo, da ferramenta de *software MATLAB*) simplifique o procedimento, pois não é necessário dominar a linguagem VHDL (BANJANOVIC-MEHMEDOVIC *et al.*, 2019).

Atualmente existem vários kits de desenvolvimento FPGA disponíveis para aquisição, desde os mais simples até os mais completos, por exemplo o kit de desenvolvimento FPGA EE03 e o kit de desenvolvimento FPGA DE10-Pro, apresentados na Figura 8. O produzido pelo Instituto de Tecnologia Emerson Martins é baseado no modelo FPGA Intel EP4CE6E22C8N da família *Cyclone IV*, apresentado na Figura 8(a) e pode ser considerado um kit de desenvolvimento simples de acordo com a quantidade de recursos que possui em comparação com o kit de desenvolvimento FPGA DE10-Pro produzido pela Terasic, baseado no modelo FPGA Intel Stratix 10 GX/SX e apresentado na Figura 8(b).

Figura 8 – Kits de desenvolvimento FPGA: (a) Kit FPGA EE03, (b) Placa DE10-Pro

(a) Kit FPGA EE03 **(b) Placa DE10-Pro**



Fonte: (a) Martins (2022), (b) Terasic (2020).

Uma vantagem excepcional da linguagem de programação descrita em nível de *hardware* é a possibilidade de verificação do projeto, e que segundo (SILVA, 2018): "Pode-se dizer que a verificação é uma tarefa ainda maior que o próprio projeto quando se trata do desenvolvimento de um circuito integrado".

Para efetuar a verificação funcional é fundamental aplicar alguma linguagem de verificação de *hardware*, diferente das linguagens HDL citadas anteriormente (VHDL e *Verilog*) que se tornam desapropriadas para tal verificação. Logo, o termo *Hardware Verification Language* (HVL) vem para definir as linguagens utilizadas para verificação de hardware, em que se enquadra a linguagem SystemVerilog, que possui as funcionalidades necessárias para implementar a verificação.

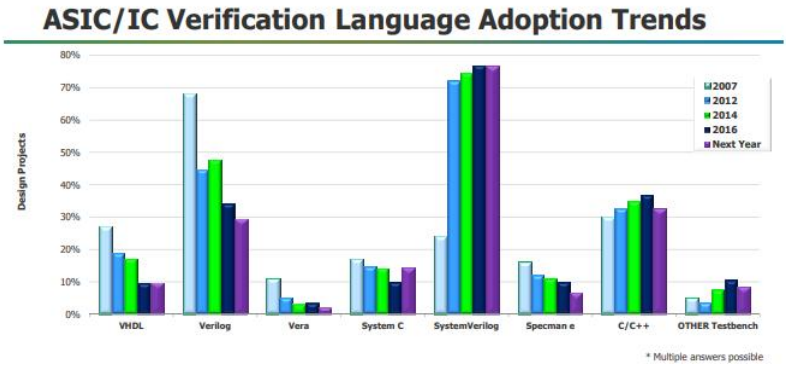
Algumas das tecnologias de verificação adotadas, citadas por Foster (2015) são a cobertura de código, cobertura funcional, declarações, verificação aleatória restrita, verificação formal

de propriedades, emulação e prototipagem de FPGA, ademais a linguagem SystemVerilog é a primeira linguagem padrão da indústria nesse âmbito e a que possui maior taxa de adoção.

A forma mais comum de realizar a verificação de um projeto que possui a particularidade de utilização de um circuito integrado é por meio de *testbenchs*. Uma definição de *testbench* segundo (SILVA, 2007) é: "O testbench é o ambiente através do qual o dispositivo a ser verificado será inserido, de forma que ele receba estímulos e que as respostas sejam comparadas com o resultado ideal", ou seja, o arquivo *testbench* nada mais é que um arquivo de simulação do código de programação, onde é pré-estabelecido os possíveis valores a serem atribuidos às variáveis de entrada e observar a resposta gerada a partir dos mesmos para as saídas, de acordo com a escala de tempo atribuida.

Para (FOSTER, 2015) ao analisar a tendência de adoção das linguagens e bibliotecas de classe base de verifição (testbenchs): "as taxas de adoção para todas as linguagens usadas para criar testbenches estão em declínio ou estáveis, com exceção do SystemVerilog". Através da Figura 9 é possível constatar que a linguagem SystemVerilog possuia a maior adoção para a geração de ambientes de verificação até 2014.

Figura 9 – Linguagens utilizadas para verificação (testbenchs)



Fonte: (FOSTER, 2015).

Na seção anterior foi apresentada uma variedade de robôs hexápodes, que utilizam diferentes linguagens de programação e métodos de controle. Este é um fator importante no desenvolvimento de protótipos robóticos de modo à realizar diferentes tipos de funções, em diferentes tipos de ambientes de forma versátil. Atrelado a isso, o FPGA possui vantagens em termos de alta velocidade de processamento e grande capacidade de memória, além de ser descrito a nível de hardware é possível utilizar-se dos recursos de verificação, que certamente contribui para a identificação das imprecisões, facilitando no aprimoramento do código. Neste projeto optou-se pela linguagem SystemVerilog, que possui a vantagem de criação do projeto, simulação, teste e implementação sem precisar utilizar outras linguagens, sendo que a utilização de outras linguagens para integração dos recursos é amplamente difundida atualmente.

3 MATERIAIS E MÉTODOS

O presente trabalho desenvolveu uma pesquisa de aplicação prática para o desenvolvimento do robô hexápode, em que neste capítulo são abordadas as considerações sobre as decisões acerca do projeto e as ferramentas utilizadas, além de ser abordado trechos do código de programação essenciais para o funcionamento do robô.

3.1 Componentes físicos

Os robôs hexapódes biologicamente inspirados geralmente são adaptações de insetos e as escolhas mais típicas abordadas em artigos acadêmicos e também para comercialização, que representam a categoria de robôs biomiméticos são as formigas, aranhas e escorpiões. Para escolha do inseto a ser adaptado, considerou-se o chassi do robô e as vantagens de cada design, por exemplo, o formato circular possui patas distribuídas em simetria axial, e tem a vantagem de se movimentar em qualquer direção sem a necessidade de rotacionar o robô para mover-se lateralmente, já a simetria retangular tem a vantagem de ser menos robusta, o que facilita a locomoção em locais estreitos, além de facilitar o comando da locomoção em linha reta. Ademais, considerou-se também a arquitetura das patas e o tamanho máximo e mínimo das peças, em que todos os fatores foram considerados de acordo com os recursos disponíveis para a construção e implementação do projeto. O robô proposto neste trabalho é um sistema biologicamente inspirado em formigas, possuindo um grau de liberdade por pata, ou seja, um atuador por pata. As próximas subeseções apresentam pormenorizadamente os componentes físicos utilizados para desenvolver o projeto.

3.1.1 Modelagem da Estrutura Física

A estrutura física foi modelada através do *software SolidWorks*, um *software* de desenho 3D e 2D para modelagem de peças, que disponibiliza recursos para criação do processo de design e simulação, onde é possível visualizar a quantidade de peças e dimensioná-las individualmente. A Figura 10 apresenta o modelo da estrutura elaborado (*design*) para o chassi do robô. A espessura da peça da estrutura do corpo possui 4 mm, enquanto o restante possui das peças 3 mm de espessura.

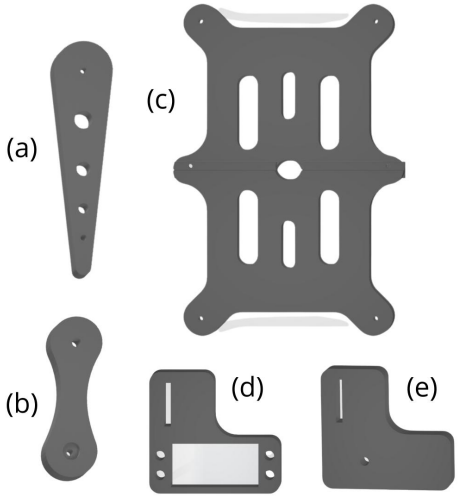
Figura 10 – Estrutura do hexapóde elaborada no SolidWorks: (a) Vista superior, (b) Vista lateral

(a) Vista superior **(b) Vista lateral**

Fonte: Autoria Própria (2022).

A estrutura apresentada acima é composta por cinco peças. A Figura 11 apresenta as peças desenvolvidas no *software SolidWorks* de forma particular, totalizando as cinco peças que foram ser denominadas como (a) coxa, (b) panturrilha, (c) corpo, (d) peça superior e (e) peça inferior, sendo as duas últimas para acomodação dos servomotores, que representam as articulações.

Figura 11 – Peças do hexapóde elaborada no *software SolidWorks*



Fonte: Autoria Própria (2022).

3.1.2 Atuador

Um fator imprescindível na estrutura do corpo do robô são os atuadores e seus respectivos mecanismos de acionamento e fonte de alimentação, sendo estes definidos a partir da particularidade pelo qual o robô foi elaborado, um projeto que necessite de locomoção em longas distâncias deve possuir um sistema de alimentação acoplado ao robô com capacidade de fornecimento de energia por longos períodos de tempo assim como um sistema de comunicação isolado, para que o mesmo consiga cumprir com seu objetivo.

Considerando a aplicabilidade deste projeto, o tipo de atuador escolhido é o servomotor elétrico, onde além de ser mais barato, possui uma forma de controle mais simples. Como o sistema dimensionado inicialmente possuia dois graus de liberdade por pata, são necessários doze atuadores no total.

Para o projeto, optou-se pelo servomotor digital com alto torque e engrenagem metálica *MG996R*, que possui três pinos, sendo os pinos vermelho e marrom para alimentação, e o pino laranja é o pino que receberá o sinal *Pulse Width Modulation* (PWM), usado para controlar o ângulo de rotação do servomotor. A Figura 12 apresenta a imagem do servomotor.

Figura 12 – Servo Motor MG996R

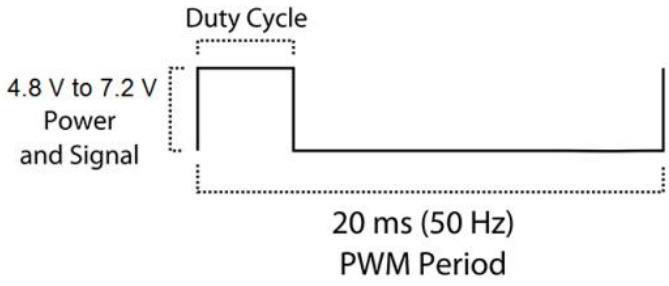
Fonte: All Datasheet (2022)..

Quanto a tensão de operação varia entre 4,8 V a 7,2 V, de acordo com o *datasheet*, e possui um torque de 9.4 kgf·cm quando submetido a tensão de 4.8 V, e 11 kgf·cm quando submetido a tensão de 6 V.

Servomotores requerem uma onda quadrada de diferentes larguras de pulso para operar, o sinal PWM. O servomotor é controlado pelo sinal PWM, onde pulsos PWM são gerados para cada motor, forma frequentemente utilizada para se controlar cada motor de forma individual.

A frequência e o período do sinal, além da largura de pulsos, são características que definem a onda gerada, e desta forma, o sinal para controlar o servomotor. O sinal PWM possui um *duty cycle*, onde é determinado com que frequência o sinal muda de nivel lógico, ou seja, nível lógico alto para nível lógico baixo. A Figura 13 apresenta o gráfico com as informações fornecidas pelo datasheet.

Figura 13 – *Gráfico com informações técnicas do MG996R*



Fonte: All Datasheet (2022).

Neste projeto, o servomotor escolhido possui um período de 20 ms, e uma frequência de 50 Hz, logo o *duty cycle*, tempo em alto, é entre 1 e 2 ms, ou seja, entre 5 e 10% (ALL DATASHEET, 2022). Por exemplo, quando o período for 1 ms o servo estará na posição 0º, quando for 1,5 ms o servo estará na posição 90º, e quando for 2 ms o servo estará na posição 180º. Esses valores podem variar na prática.

3.1.3 Fonte de Alimentação

Para doze atuadores, é necessário uma fonte de tensão somente para alimentar os servomotores. Estimou-se a corrente e tensão necessárias para alimentar os doze atuadores, e optou-se pela alimentação realizada por meio de uma fonte externa, sendo ela uma *SATE PRO* de 400 W, em pico, o que fornece um intervalo seguro de operação considerando as estimativas prévias serem inferiores a potência fornecida por esta fonte em questão. A Figura 14 apresenta a imagem da fonte de alimentação utilizada.

Figura 14 – SATE PRO-460 400W

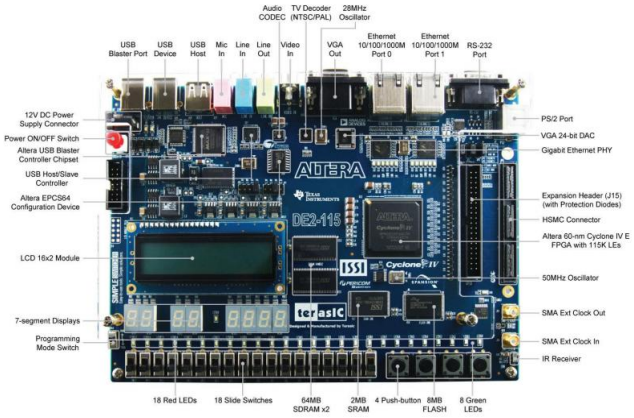
Fonte: Satellite (2021).

3.1.4 Controlador

O controlador FPGA para este projeto é o *Cyclone IV*, da Altera. Ele possui um tamanho sucinto, além de possuir software de programação e verificação gratuitos. O *software Quartus*

Prime e suas extensões foram utilizados para elaboração da programação e simulações. A Figura 15 apresenta a placa de desenvolvimento DE2-115, fabricada pela empresa *Terasic*, que é utilizada neste projeto, e disponibilizada pelo Departamento de Ciências da Computação do Campus Medianeira desta instituição.

Figura 15 – *Placa DE2-115 – Terasic*



Fonte: Terasic (2017).

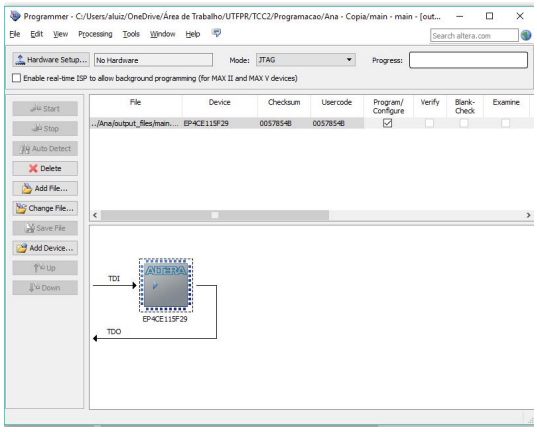
A Placa DE2-115 contém o *chip* FPGA Cyclone IV modelo EP4CE115F29C7N, e inclui um oscilador que produz um sinal clock de 50 MHz, além de possuir dois conectores *SubMiniature Version A* (SMA) para conectar fontes de clock externa à placa ou interna. A placa inclui um conector de energia de 12 V para alimentação, e o cabo USB Blaster, que é utilizado para gravar a programação através da interface USB do computador. A placa foi disposta pelo departamento de computação do campus da Universidade. Ela possui dimensão robusta, sendo a estrutura física projetada para alocação da placa.

De forma mais aprofundada, a placa possui o sistema *NIOS II*, um microcontrolador embarcado, possuindo processador, conjunto de periféricos que incluiem comunicação serial, processamento de sinais e interfaces, além de memória, no mesmo *chip* (FONTOURA; NASCIMENTO; GIOPPO, 2013). Isto se mostra uma grande vantagem, levando em consideração ser um microcontrolador embarcado, ele pode ser reconfigurado de inúmeras maneiras e integrado ao resto do circuito, já que a empresa Altera concede sua arquitetura.

O FPGA pode ser configurado utilizando-se do cabo USB Blaster, das seguintes formas: *Joint Test Action Group Joint Test Action Group* (JTAG) e *Active Serial Active Serial*. (AS) modos de programação. A forma utilizada neste projeto é a JTAG, e no método de programação JTAG, o fluxo de bits de configuração é baixado diretamente no Cyclone IV E. Desta forma, o FPGA manterá a programação enquanto a alimentação for mantida à placa (TERASIC, 2017).

Para comunicação JTAG, utiliza-se o recurso *Programming* disponível no *software Quartus Prime*, onde o *USB-Blaster* deve estar conectado ao computador para ser selecionado. A Figura 16 apresenta o recurso *Programming*, onde utiliza-se o modelo do dispositivo FPGA utilizado juntamente ao modo de comunicação e ao arquivo a ser baixado em formato .sof, podendo ser outro formato caso haja a preferência de manter o código na memória flash da placa.

Figura 16 – Recurso *Programming* disponível no *software Quartus Prime* para execução do programa

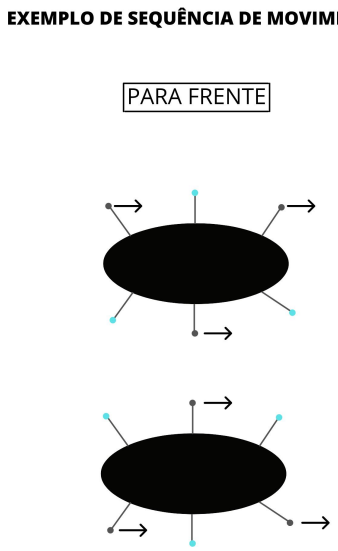


Fonte: Autoria Própria (2022).

3.2 Componentes Lógicos

Para elaboração da programação levou-se em consideração a sequência de movimentos e o interesse em manter o processo simples e didático, desta maneira os processos foram desenvolvidos de acordo com a sequência de movimentos ilustrado na Figura 17.

Figura 17 – Exemplo de sequência para programação dos movimentos



Fonte: Autoria Própria (2021).

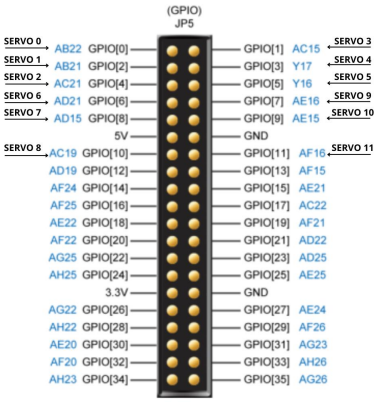
O hexápode é estaticamente estável, logo, quando mantém a sincronia do movimento de manter três patas em um ponto fixo enquanto movimenta outras três patas ele possui estabilidade estática, vantagem atribuída aos modelos biomiméticos.

3.2.1 Programação em SystemVerilog

De forma geral, a programação é desenvolvida na linguagem Systemverilog, que permite a verificação funcional do projeto, de maneira a implementar módulos para controle dos servomotores, de forma em que seja possível ser utilizado para qualquer servomotor e necessidade, alterando as constantes com base no valor das entradas genéricas no código de programação.

Na elaboração a saída PWM é o sinal enviado para o controle do servomotor, ele deve passar por um pino da Placa DE2-115, e se conectar à entrada do sinal PWM no servomotor. A Placa DE2-115 conta com uma expansão de conectores de quarenta pinos, e fornece alimentação corrente contínua de 5 V e de 3,3 V, além de dois pinos de referência de tensão 0 V. A Figura 18 apresenta a imagem da distribuição de pinos da expansão de conectores JP5, onde estão relacionados os servomotores com os respectivos pinos de saída da GPIO. A maioria dos projetos precisam de um conversor de tensão, pois a maioria dos FPGAs usa nível lógico com saída em 3,3 V, enquanto a maioria dos servosmotores tem seu funcionamento em 5 V, tanto na alimentação quanto no sinal.

Figura 18 – Arranjo de pinos da Placa DE2-115 fornecido no Manual DE2-115

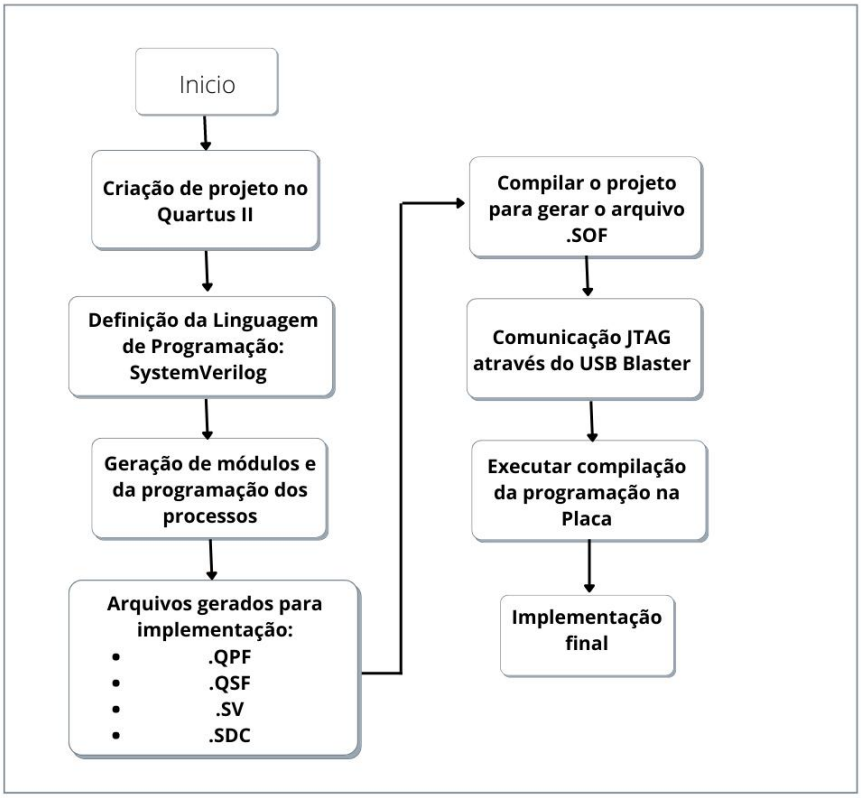


Fonte: Terasic (2017).

Uma vez que o programa foi simulado e verificado, o arquivo do circuito é gerado e programado para o *chip* FPGA através do procedimento de programação JTAG Ali *et al.* (2013), em que deve ser ligada a Placa DE2-115, e em sequência conectado à entrada JTAG por meio do cabo USB Blaster para comunicação com o computador e, por fim, realizada a gravação da programação.

De modo geral, a programação será elaborada de acordo com as etapas apresentadas na Figura 19 no *software Quartus Prime* para a implementação final.

Figura 19 – Etapas da programação no *software Quartus Prime*



Fonte: Autoria Própria, 2022.

Ao fim de todas as simulações e execuções, o protótipo será testado e modificado de acordo com a necessidade do sistema na prática.

4 RESULTADOS E DISCUSSÃO

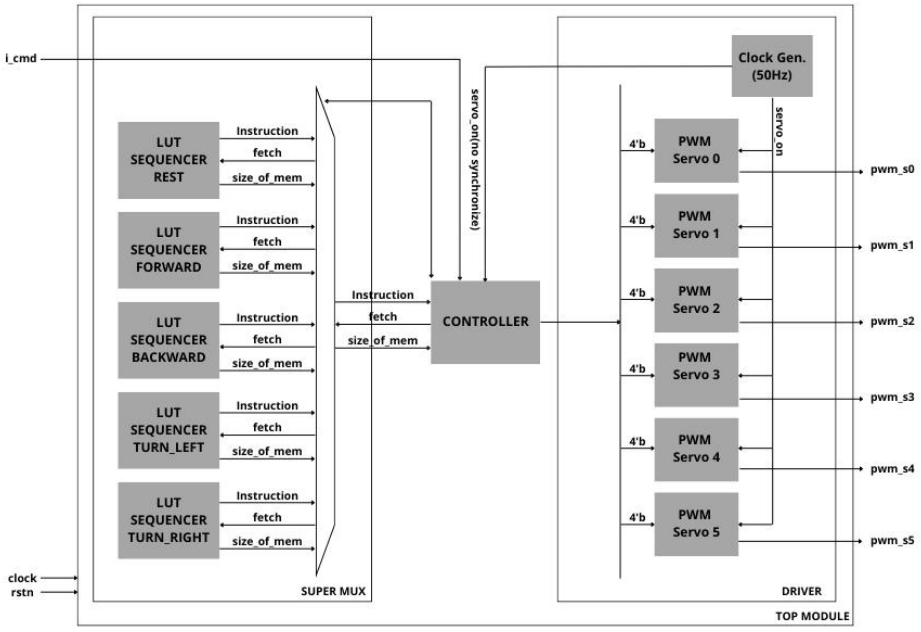
Neste Capítulo serão apresentados os resultados obtidos através da aplicação da metodologia descrita no Capítulo 3 para a construção da estrutura física, composta pelos atuadores e Placa DE2-115, fonte de alimentação e a verificação e execução do código da programação, para enfim a execução do protótipo final e ajustes.

4.1 Elaboração do código de programação

4.1.1 Visão macro do código de programação

A principal intenção deste estudo é a elaboração de um código de programação que possa ser facilmente adaptável a outros projetos, principalmente voltados para futuros estudos acadêmicos, diante disto o código de programação foi elaborado visando um modelo expansível e de fácil reconfiguração, dividido em módulos, de forma breve, o módulo denominado "*CONTROLLER*"tem como objetivo receber o comando de movimento previamente determinados nas denominadas *LUTs*, ou seja, uma memória que armazena os comandos pré-definidos de instruções de acionamento dos servomotores através do *duty cycle*) e que está implementada nas chaves *switchs* da Placa DE2-115, de forma a enviar o sinal para as saídas, os servomotores. A Figura 20 apresenta o diagrama de bloco da visão macro do projeto, conforme descrito acima.

Figura 20 – Diagrama de bloco do código de programação



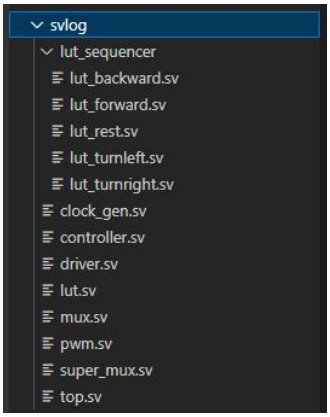
Fonte: Autoria Própria, 2022.

Observando a Figura 20, é possível notar a divisão dos módulos em três módulos destacados, denominados de "*TOP MODULE*", "*DRIVER*" e "*SUPER MUX*", onde o módulo denominado de "*TOP MODULE*" faz a conexão lógica (descrita no código de programação) entre o módulo denominado "*DRIVER*" e o módulo denominado "*SUPER MUX*" com o módulo "*CONTROLLER*", como citado anteriormente, o módulo controle responsável pela interligação da entrada com a saída.

Definiu-se no total treze módulos que foram denominados, conforme a Figura 21. O tópico denominado de "*lut_sequencer*" são os módulos respectivos as instruções de movimentos pré estabelecidos, sendo cinco módulos, denominados de: "*lut_backward*", "*lut_forward*", "*lut_rest*", "*lut_turnleft*" e "*lut_turnright*", ou seja, as instruções respectivas dos movimentos de recuo, de avanço para frente, de posição inicial, de deslocamento lateral para a esquerda e de deslocamento lateral para direita.

Além de outros oito módulos, que serão abordados de forma mais específica, denominados como: "*clock_gen*", "*controller*", "*driver*", "*lut*", "*mux*", "*pwm*", "*super_mux*" e "*top*". A Figura 21 apresenta a tela hierarquica do código de programação completo, composta por tais módulos citados acima, o nível mais alto sendo o módulo "*top*".

Figura 21 – Recorte dos módulos criados no software Quartus



Fonte: Autoria Própria, 2022.

O módulo denominado "*top module*" tem como interface representando as entradas portanto: o sinal *i_cmd*, que é responsável pelo sinal de comando do projeto (através da chave *switch*), o sinal "*clk*" e o "*rstn*" sendo os dois últimos responsáveis pela sincronicidade do código de programação. Representando as saídas: os seis sinais PWM denominados "pwm_[S0..S5]", que estão ligados fisicamente a seis servomotores MG996R.

Descrevendo de forma prática o funcionamento do diagrama de bloco apresentado na Figura 22: O sinal denominado de *i_cmd*, que está atribuído fisicamente com as *switches* da Placa DE2-115, selecionará um dos módulos denominado de "*lut_sequencer*" de acordo com a posição selecionada nas *switches*, através do módulo "*controller*".

Supondo que seja escolhida através da *switch*, a denominada "*lut_forward*" (movimento de avanço para frente), por exemplo, para tal o controller que intermedia o módulo denominado de "*super_mux*" vai buscar a primeira instrução contida no módulo da "*lut_forward*", e esse bloco

irá retornar o tamanho da memória contida, ou seja, quantas instruções ela possui, e o tamanho da memória. Essa instrução, que será abordada posteriormente, e consiste na configuração de *duty cycles* definida no módulo para cada servomotor de forma individual. Esse tamanho da memória, denominado como "*size_of_mem*", retorna a variável denominada "*fetch*", que endereça a memória, ou seja, o *fetch[0]* estará referenciado à *instruction[0]*, de acordo com a quantidade de instruções que o módulo tem pré definido.

Após o controlador receber o comando para escolha do movimento para frente, ele irá comunicar as variáveis definidas no módulo de controle com as definidas nos módulos de sequência de movimentos, neste caso, com a denominada "*lut_forward*", e então executando a variável *fetch* das instruções e relacionando-as com as pré definições da máquina de estado, enviando assim as especificações de *duty cycle* para o sinal PWM de cada servomotor particularmente, atrelado ao módulo denominado "*clock_gen*" que recebe a frequência de 50 MHz gerada pelo clock da placa de implementação, e divide a frequência para 50 Hz de forma a gerar os sinais de saída PWM com a frequência de funcionamento dos servomotores, 50 Hz, com variação de duty cycle.

A seguir, será abordado de forma concisa o que compreende cada bloco de forma individual.

4.1.2 Descrição dos módulos

O módulo denominado como "*controller*" é responsável pelo controle do circuito, além de ser o módulo mais complexo em termos de programação. Este módulo é uma máquina de estado do tipo *Moore*, em que o código registra o estado atual e computa na lógica combinacional o próximo estado. Possui um *delay* responsável por manter a configuração pré definida selecionada até que a mesma finalize a execução de modo a não perder o estado correto do sistema. O *delay*, inserido como uma variável de fácil ajuste no módulo, mantém a configuração do módulo de sequência de movimento selecionada, ou seja, o *delay* é responsável por manter a instrução até que ela termine de ser executada, e então buscar a próxima instrução, que também será executada respeitando o *delay*, em *looping* até que termine a execução da última instrução selecionada, e receba um novo comando. Na Figura 22 é exposto as variáveis do módulo citadas anteriormente.

Figura 22 – Parâmetros do módulo denominado "*controller*"

PARÂMETROS - CONTROLLER

```
module controller #(
    parameter CLK                   = 50000000,          // 50M Hz
    parameter DELAY_MS              = 40,                // delay in MS (multiple of 20ms)
    parameter longint DELAY_VALUE   = CLK*DELAY_MS/1000,
    parameter MAX_NUM_OF_SERVOS     = 'd6,
    parameter NBW_DUTY_SELECTOR     = $clog2(16),        // size of LUT
    parameter NBW_ACCUMULATOR       = 32,                // result of f_clk/f_target
    parameter MAX_NUM_OF_INSTRUCTIONS = $clog2(5),
    parameter MAX_NUM_OF_STATES     = $clog2(5)
```

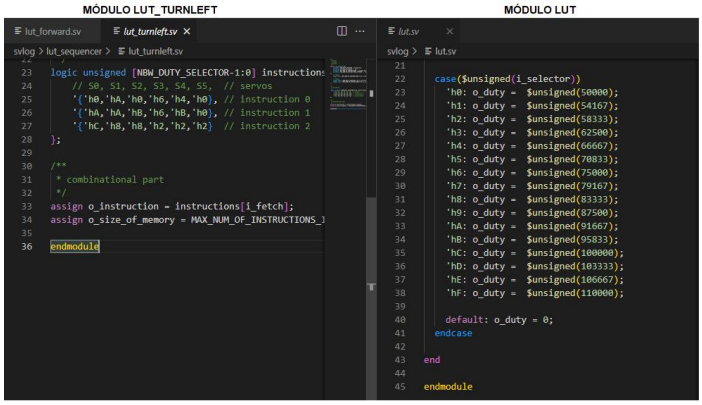
Fonte: Autoria Própria, 2022.

O módulo denominado "*pwm*" recebe como parâmetro o clock do dispositivo (50MHz), e o clock de operação dos servomotores (50Hz), o resultado da relação entre eles, o máximo valor que o duty cycle pode assumir, e a quantidade de instruções, ou seja, ele possui a função de gerar os sinais de saída em 50Hz, conforme o duty cycle determinado no modulo denominado "*lut*".

O módulo denominado como "*driver*" basicamente relaciona o módulo denominado como "*clock_gen*" e o módulo denominado como "*pwm*" de forma a gerar o número de sinais PWM respectivamente atrelado ao número de saídas de sinal para os servomotores. Já o módulo denominado como "*clock_gen*" é um divisor de frequência, possui a função de gerar o sinal de *clock* de saída com frequência de 50 Hz.

Existem seis módulos denominados como "*luts*" neste projeto, os módulos de sequência de movimento e um módulo denominado como "*lut*". O módulo "*lut*" pode ser dito como um módulo de atribuição do o valor de *duty cycle*, ou seja, ele atribui à uma variável (h[0..F]) referente à posição angular do servomotor o seu respectivo valor de *duty cycle*, logo ele retorna ao módulo PWM a quantidade de ciclos que o sinal PWM deve contar para alterar o estado do sistema, do nível lógico 1 para 0. Os outros cinco módulos definidos são os módulos de sequência de movimentos, onde é definido em seus parâmetros as instruções de acordo com os valores de *duty cycle* configurados no módulo módulo "*lut*". A Figura 23 expõe as variáveis dos módulos *lut* e "*lut_turnleft*", de modo a observar a forma em que estes módulos estão relacionados e de visualizar o descrito acima.

Figura 23 – Trecho do código de programação do módulo LUT e LUT_TURNLEFT



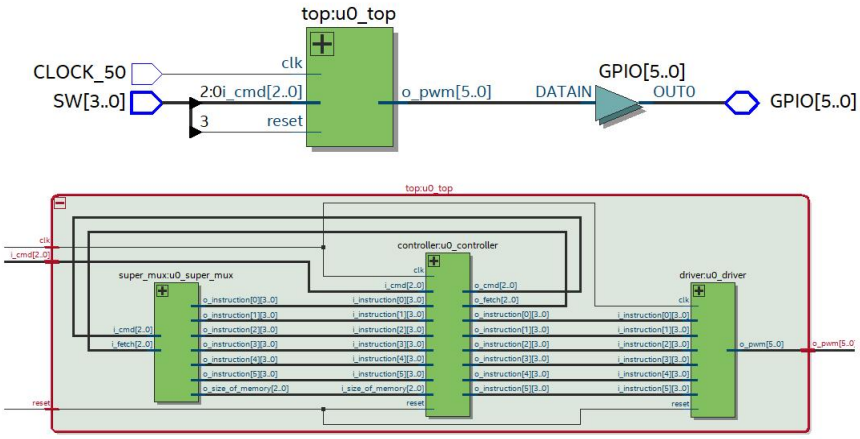
Fonte: Autoria Própria, 2022.

O módulo denominado como "*super_mux*" engloba os módulos de sequência de movimento e dentro dele ocorre a instanciação dos mesmo. Já o módulo denominado como "*mux*" é diretamente relacionado com o módulo de controle e a entrada denominada "*i_cmd*", onde ele tem como função retornar o sinal da instrução e o tamanho da memória contida no módulo de sequência de movimento.

A Figura 24 exibe o *RTL Viewer*, uma ferramenta do *software Quartus Prime Lite*, em que após a compilação do código de programação, é gerada a visualização das conexões lógicas

realizadas pelo cógido compilado, ou seja, ela exibe a visualização do *Register-Transfer Level* (RTL) (interpretação do hardware descrito em HDL).

Figura 24 – RTL Viewer



Fonte: Autoria Própria, 2022.

4.2 Montagem da estrutura física do robô hexápode

A estrutura foi impressa em uma impressora 3D *FlyingBear Ghost 4S*, disponibilizada pelo Departamento de Engenharia Elétrica do câmpus Medianeira. O material utilizado na impressão é o filamento de PLA de 1,75 mm, disponível no laboratório do departamento.

A impressão foi realizada peça a peça, devido à capacidade da impressora 3D disponibilizada, onde a peça que representa o corpo foi dividida em duas partes, logo além das duas peças do corpo foram impressas seis peças referente a coxa, seis peças referente a parte superior e seis referente a inferior para acomodação dos servomotores, seis peças referente a panturrilha, sendo a coxa responsável pelo segundo grau de liberdade. A Figura 25 apresenta as peças impressas.

Figura 25 – Peças fabricadas na impressora *3D FlyingBear Ghost 4S*

Fonte: Autoria Própria, 2022.

Para montagem foram utilizados parafusos, arruelas e porcas para fixação dos atuadores e demais peças presentes na estrutura, além disso, para melhor adequação da saída do atuador, ou seja, do mecanismo que realiza o movimento, foi acoplado um encaixe que acompanha o servomotor.

A montagem da estrutura física foi realizada em três partes. A primeira parte consiste na fixação das peças correspondentes a panturrilha e a coxa. A panturrilha foi fixada à coxa por meio de parafusos,arruelas e porcas. A segunda parte consistiu na fixação dos servomotores com suas respectivas peças (inferior e superior) e, em sequência, a fixação da peça que acompanha o servomotor para acoplamento da saída à pata montada anteriormente. A Figura 26 (a) apresenta as duas partes da montagem. A terceira parte da montagem da estrutura física constistiu na fixação das duas peças do corpo, que foram elaboradas com uma superfície rebaixada para facilitar a colagem, tendo em vista as limitações relativas à dimensão da peça corpo onde foi necessário dividi-lá em duas peças para se adequar ao tamanho da base da impressora. A Figura 26 (b) apresenta a imagem do corpo após a colagem das duas peças.

Figura 26 – (a)Partes um e dois da montagem da estrutura física; (b) Placa DE10-Pro

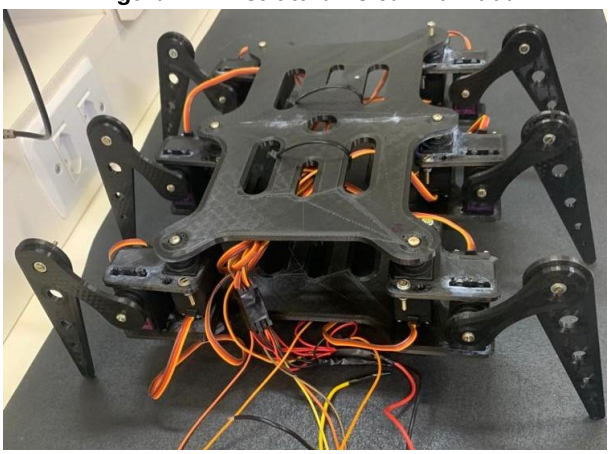
(a) Partes um e dois da montagem da estrutura física

(b) Peça corpo após colagem

Fonte: Autoria Própria, 2022.

Em sequência, foi fixado à estrutura superior do corpo os adaptadores que acompanham o servomotor, e as patas. Para organização dos fios e circuito, optou-se por não utilizar *protoboard*, a fonte de energia foi adaptada através de *jumpers* para realizar toda a ligação, e utilizou-se de abraçadeira para fixação na peça do corpo. A Figura 27 apresenta a imagem da estrutura física finalizada, conforme descrito.

Figura 27 – Estrutura física finalizada

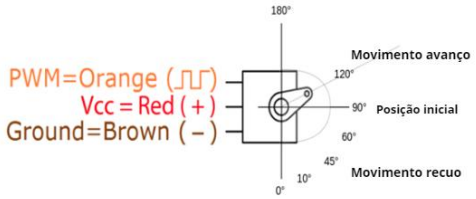


Fonte: Autoria Própria, 2022.

4.2.1 Calibração dos duty cycles dos servomotores

Os servomotores utilizados neste projeto são do tipo *closed loop*, ou seja, o ângulo varia de 0º a 180º de acordo com o referencial, o que para esta aplicação é suficiente. Para referenciar as posições dos servomotores, foi definido um referencial conforme apresenta a Figura 28, onde em 90º foi definido como posição inicial, a condição $90 + X$, sendo X um valor entre 0º e 90º, como movimento de avanço e $90 - X$ como movimento retrógrado, em relação aos servomotores do primeiro grau de liberdade, o responsável pela movimentação do robô hexápode.

Figura 28 – Referêncial de posição e ângulo do servomotor



Fonte: Autoria Própria, 2022.

Portanto, elaborou-se uma planilha no *software Excel* onde inicialmente foi realizada a calibração dos servomotores, no laboratório do departamento, utilizando-se de um gerador de sinais de onda quadrada configurado em 50 Hz de frequência e tensão de 3,3 V (simulando a tensão fornecida pela GPIO, extensão da placa no pino de saída dos sinais) alimentando o fio do sinal do servomotor e um transferidor para estimar o ângulo de saída. Na coluna denominada "*DUTY CYCLE* (ms)" estão os valores obtidos através da calibração realizada com o gerador de sinais e cada servomotor de forma individual, para o movimento de avanço, logo a coluna denominada "REFERÊNCIA" estão as posições para tal movimento, sendo elas: posição inicial ou "posição 0"(90º) e posição de avanço (135º), e convertido para o valor em *duty cycle*.

A Figura 29 apresenta a tabela criada para conversão do *duty cycle* para ser inserido no código de programação, mais especificamente nos módulos de sequência de movimentos e

no módulo denominado como "*lut*", afim de simplificar e organizar o procedimento de calibração dos servomotores e a inserção dos dados no código. Os próximos passos na calibração é repetir o processo anterior alterando para os ângulos referentes aos movimentos de avanço, recuo, deslocamento para direita e deslocamento para esquerda.

Figura 29 – Tabela de conversão para o duty cycle

REFERÊNCIA	LUT	ÂNGULO	DUTY CYCLE (ms)	CONVERSÃO DUTY CYCLE
S0 - Posição 0	0	0	1,38	69000
S0 - Posição Avanço	1	15	1,70	85000
S1 - Posição 0	2	30	1,06	53000
S1 - Posição Avanço	3	45	1,44	72000
S2 - Posição 0	4	60	1,30	65000
S2 - Posição Avanço	5	75	1,66	83000
S3 - Posição 0	6	90	1,66	83000
S3 - Posição Avanço	7	105	1,24	62000
S4 - Posição 0	8	120	0,9800	49000
S4 - Posição Avanço	9	135	0,76	38000
S5 - Posição 0	10	150	1,08	54000
S5 - Posição Avanço	11	165	0,86	43000
0	12	180	0,86	43000
~~0~~	~~13~~	~~195~~	~~0,93~~	~~46333~~
~~0~~	~~14~~	~~210~~	~~0,99~~	~~49667~~
~~0~~	~~15~~	~~225~~	~~1,06~~	~~53000~~

*Para obter os dados da tabela acima utilizou-se a calibração do servo a partir de um gerador de sinais;

ÂNGULO	0º	135º	45º
EM %	DUTY CYCLE	DUTY CYCLE	DUTY CYCLE
Servo 0	6,9	8,5	5,3
Servo 1	5,3	7,2	2,9
Servo 2	6,5	8,3	4,7
Servo 3	8,3	6,2	10,3
Servo 4	4,9	3,8	7,2
Servo 5	5,4	4,3	7,5

Fonte: Autoria Própria, 2022.

Para as instruções contidas nos módulos das sequências de movimentos utilizou-se também de tabelas, contidas na mesma planilha criada para a calibração dos servomotores, que em relação ao código de programação possui fácil reprogramação para adaptabilidade em outros projetos. A instrução, que pode ser única ou não, é facilmente adicionada ao módulo da sequência de movimentos que pode ser observada na Figura 23. A Figura 30 apresenta a tabela criada para inserção das instruções para ser inserido no código de programação, afim de simplificar e organizar o procedimento de instruções.

Figura 30 – Tabela de instruções dos movimentos

LUT FORWARD	Instrução 0	Instrução 1
Servo 0	1	0
Servo 1	2	3
Servo 2	5	4
Servo 3	6	7
Servo 4	9	8
Servo 5	10	11

LUT REST	Instrução 0
Servo 0	0
Servo 1	2
Servo 2	4
Servo 3	6
Servo 4	8
Servo 5	10

LUT BACKWARD	Instruction 0	Instruction 1
Servo 0	9	6
Servo 1	6	9
Servo 2	9	6
Servo 3	6	9
Servo 4	9	6
Servo 5	6	9

LUT TURNLEFT	Instrução 0	Instrução 1	Instrução 2
Servo 0	0	10	12
Servo 1	10	10	8
Servo 2	0	11	8
Servo 3	6	6	2
Servo 4	4	11	2
Servo 5	0	0	2

LUT TURNRIGHT	Instrução 0	Instrução 1	Instrução 2
Servo 0	12	4	0
Servo 1	6	4	4
Servo 2	12	5	4
Servo 3	0	0	8
Servo 4	8	5	8
Servo 5	12	12	8

Fonte: Autoria Própria, 2022.

Com relação a calibração do servomotor, cada dispositivo em razão da forma que a engrenagem foi encaixada na estrutura física, sofreu uma variação na relação de ângulo e *duty cycle* correspondente. Logo surge a necessidade de calibração individual de cada servomotor, considerando as relações de *duty cycle* e ângulo de saída individual de cada um, ou seja, cada servomotor possui um valor distinto de *duty cycle* para um determinado ângulo correspondente, a posição inicial, por exemplo, de cada um dos servomotores possui um valor distinto. A calibração é fundamental para o funcionamento correto do código de programação, de forma que haja precisão nos movimentos pré definidos desejados.

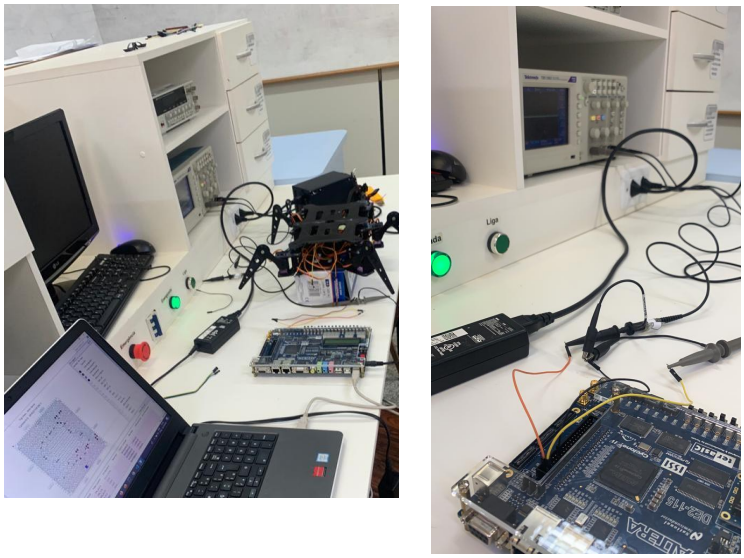
É imprescindível ressaltar que o módulo PWM recebe um valor de *duty cycle*, que está atrelado ao módulo denominado como "*lut*". Essa é uma vantagem do código de programação ter sido criado para ser facilmente reconfigurado, ampliando as possibilidades de aplicação.

4.2.2 Implementação e testes

Após os ajustes e atualizações no código de programação, realizou-se testes com o protótipo onde na implementação dos *duty cycles* foram redefinidos de acordo com a calibração individual de cada servomotor, e ajustados para a execução funcional do movimento. A Figura 31 apresenta a imagem do laboratório onde foi realizada a calibração dos servomotores, implementação na FPGA para os testes e ajustes. A Figura 31(a) exibe a bancada de testes, onde utilizou-se de um computador, cabo para comunicação com a FPGA *USB-Blaster, jumpers* e o osciloscópio, a Figura 31(b) mostra a ligação da extensão GPIO da placa DE2-115 com o osciloscópio, onde aplicou-se às saídas referentes aos servomotores das duas patas frontais, em razão da estabilidade, um encontra-se na posição inicial enquanto o outro executa o movimento de avanço, 90º e 135º graus respectivamente, de forma sucessiva.

Figura 31 – Bancada de testes: (a) conexão dos equipamentos, (b) conexão com o osciloscópio

(a) conexão dos equipamentos **(b) conexão com o osciloscópio**



Fonte: Autoria Própria, 2022.

Na Figura 32 é possível observar os resultados obtidos no osciloscópio, a Figura 32(a) apresenta os sinais dos dois pinos de saída para os servomotores S0 e S4, de forma que como os sinais possuem os mesmos valores, variando apenas o *duty cycle* e portanto, estão sobrepostos. Na Figura 32(b) os sinais foram separados e ampliados, uma função do osciloscópio, para que ficasse visível.

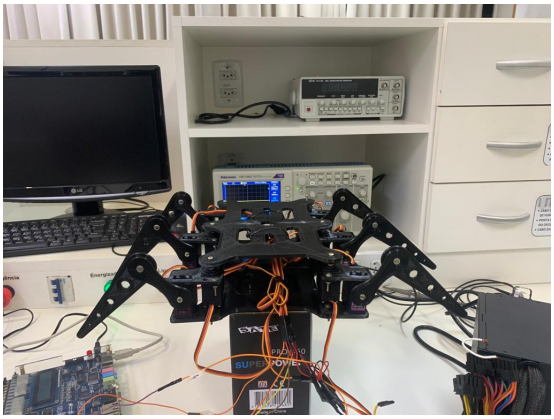
Figura 32 – Resultados obtidos no osciloscópio:(a) sinais PWM sobrepostos, (b) sinais PWM separados

(a) conexão dos equipamentos **(b) conexão com o osciloscópio**

Fonte: Autoria Própria, 2022.

Após os testes com o osciloscópio, a programação foi implementada na FPGA e conectada no protótipo do robô, de modo a avaliar a amplitude dos passos e ajustá-lo. A Figura 33 exibe a vista superior do robô implementado na bancada para os testes de amplitude.

Figura 33 – Vista superior do robô hexápode implementado

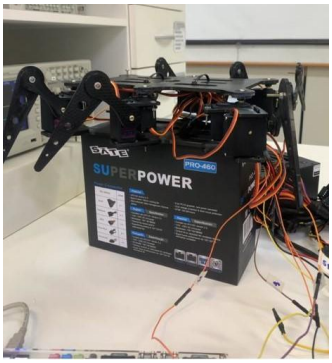


Fonte: Autoria Própria, 2022.

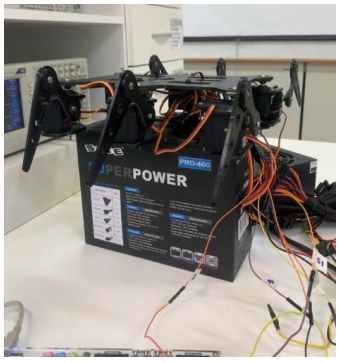
A Figura 34 demonstra a posição dos servomotores quando o comando "*rest*" e "*forward*" são acionados, ou seja, posição inicial e movimento de avanço. Na Figura 34(a) observa-se a posição dos servomotores quando apenas o botão de reset está acionado, na Figura 34(b) seleciona-se através do botão a *LUT_REST*, na Figura 34(c) seleciona-se através do botão a *LUT_FORWARD*, e na Figura 34(d) não é efetuado nenhum comando, logo a *LUT_FORWARD* continua sendo executada.

Figura 34 – Implementação para testes: (a) passo 1: *reset* acionado, (b) passo 2: *rest* acionado, (c) passo 3: *forward* acionado, (d) passo 3: *forward* acionado

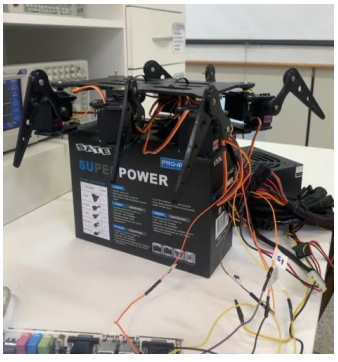
(a) Passo 1: *reset* acionado



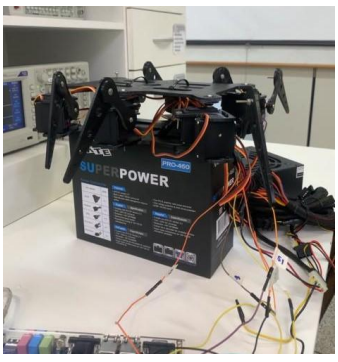
(b) Passo 2: *rest* acionado



(c) Passo 3: *forward* acionado



(d) Passo 3: *forward* acionado



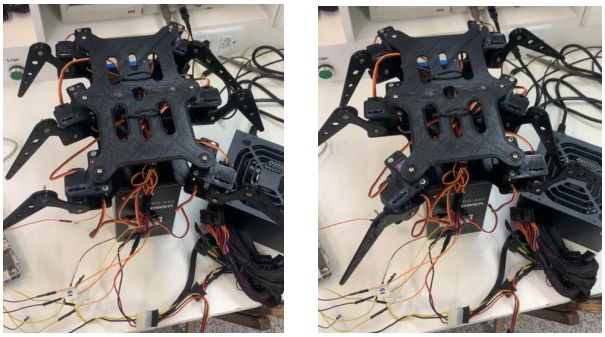
Fonte: Autoria Própria, 2022.

Analisando a Figura 34(a), passo um na implementação, coloca-se a *switch* referente ao *reset* em posição alta e o sinal de *clock* é acionado, neste momento os servomotores estão em uma posição não definida, já na Figura 34(b) segundo passo da implementação, coloca-se a *switch* referente a *LUT_REST* em posição alta e o sinal de saída do PWM é acionado e estão configurados para a posição inicial, em 90º, onde comparando as figuras é possível observar a amplitude da movimentação dos servomotores. A Figura 34(c) terceiro passo da implementação, coloca-se a *switch* referente a *LUT_FORWARD* em posição alta e o sinal de saída do PWM é acionado, neste momento a posição dos servomotores S0, S2 e S4 estão em 135º enquanto os servomotores S1, S3 e S5 estão em 90º, completando a primeira instrução, enquanto na Figura 34(d) não é executado nenhum novo comando, ocorre a execução da segunda instrução da *LUT_FORWARD*, em que os servomotores S0, S2 e S4 estão em 90º enquanto os servomotores S1, S3 e S5 estão em 135º, finalizando assim o movimento de avanço para frente, que é executado em looping até ser inserido um novo comando.

A Figura 35 mostra a vista superior do robô realizando as intruções da *LUT_FORWARD*, referente ao movimento de avanço para frente. A Figura 35(a) exibe a execução da instrução 1, enquanto a Figura 35(b) exibe a execução da instrução 2, citadas anteriormente.

Figura 35 – Vista superior dos movimentos da LUT_FORWARD: (a) instrução 1, (b) instrução 2

(a) conexão dos equipamentos **(b) conexão com o osciloscópio**

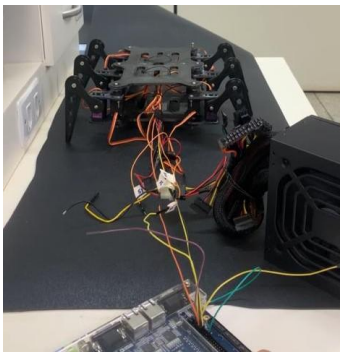


Fonte: Autoria Própria, 2022.

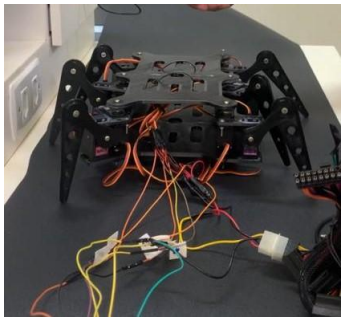
Após os testes, utilizou-se de um tapete emborrachado para aumentar o atrito das patas com a superfície de modo a favorecer a movimentação do robô. A Figura 36 demonstra a execução dos movimentos do robô hexápode após os ajustes e testes, onde na Figura 36(a) o robô encontra-se na posição inicial, logo a lut sequencer executada é a de posição inicial (*lut_rest*, na Figura 36(b) é executado módulo de sequências de movimentos denominada *LUT_FORWARD*, onde a movimentação é iniciada. Na Figura 36(c) e na Figura 36(d) não ocorre nenhum comando novo, a execução da *LUT_FORWARD* continua a ocorrer, executando as duas instruções em looping.

Figura 36 – Implementação final: (a) posição inicial, (b) momento um do movimento de avanço, (c) momento dois do movimento de avanço, (d) momento três do movimento de avanço

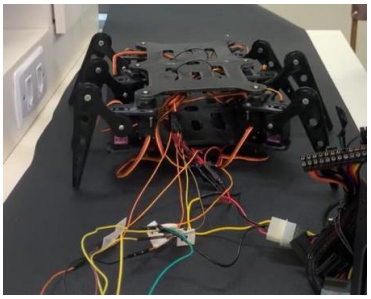
(a) Posição inicial



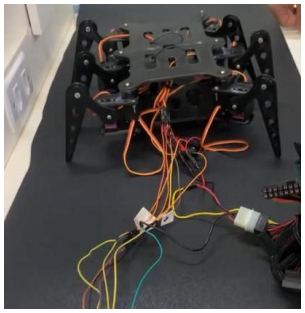
(b) Momento um do movimento de avanço



(c) Momento dois do movimento de avanço



(d) Momento três do movimento de avanço



Fonte: Autoria Própria, 2022.

Ao final do teste, percebeu-se a fragilidade do material utilizado para impressão das peças do robô, onde serão necessárias adaptações para que a estrutura se torne mais rígida e consiga suportar a movimentação plena. Na fixação dos parafusos, ocorreu um desbalanceamento que levou o robô hexápode a desbalancear na parte traseira, afetando o desempenho dos movimentos. Para trabalhos futuros propõe-se o aprimoramento da robostez da estrutura física do robô hexápode.

5 CONCLUSÕES E PERSPECTIVAS

Como pretendido, o protótipo funcional do hexápode foi construído, o algoritmo foi desenvolvido de forma a ser explorada a capacidade de reconfiguração da FPGA, a modelagem da estrutura 3D do robô hexápode foi projetada no *software SolidWorks* e elaborada com as dimensões do servomotor MG996R facilitando assim no encaixe das peças e a montagem da estrutura física.

O planejamento da execução dos movimentos ocorreu de forma satisfatória, foram necessários ajustes na bancada de testes para correção da amplitude de alguns dos ângulos dos servomotores para movimentação adequada do robô, porém os ajustes ocorreram de forma eficaz considerando a adaptabilidade do algoritmo e organização do mesmo em módulos independentes que se comunicam.

A execução dos movimentos, em relação a amplitude dos mesmos, ocorreu conforme esperado após os ajustes e considerando os objetivos. O código de programação desenvolvido no *software Quartus Prime Lite* que se mostrou intuitivo tanto no estudo da linguagem de programação *SystemVerilog* quanto nas ferramentas de simulação e comunicação.

Durante a execução final, notou-se a necessidade de reforçar a parte estrutural para assim obter um movimento mais robusto e preciso, além de afetar na execução dos movimentos do segundo grau de liberdade. Porém o projeto foi suficiente para validar e analisar os movimentos, assim como para estudar a linguagem de programação a nível de *hardware* e a implementação em FPGAs.

O algoritmo de controle foi elaborado e implementado no dispositivo FPGA, especificamente contido na Placa DE2-115, onde a comunicação entre o controle e o protótipo mostraram-se satisfatórios e de alto nível aos requisitos deste projeto, principalmente em razão da velocidade da execução dos comandos que é proporcionado. O projeto criado possui uma função relativamente simples, podendo ser facilmente expansível para que seja aplicado a diferentes soluções através de um dispositivo de lógica reprogramável FPGA, não necessáriamente da fabricante Altera.

O estudo da programação em nível de *hardware* se mostrou uma ferramenta perspicaz para ampla aplicação na área da engenharia. O algoritmo foi projetado de forma a permitir amplas modificações, podendo ser facilmente alterado o número de saídas PWM desejadas e logo, os graus de liberdade que o projeto pode possuir, além dos *duty cycles* e instruções serem inseridos em módulos separados, e escritos de forma clara no algoritmo, assim como os ajustes de *delay* e outras variáveis que possam afetar o projeto.

Um exemplo do código de programação a ser utilizado para um outro projeto é o braço robótico, através da alteração do número de saídas, os valores de *duty cycles*, frequência de saída, instruções dos movimentos, o *delay* para execução e outras variáveis desejadas. Outra adaptação facilmente a ser implementada é a alteração da chave *switch* para o botão *push-botton*, onde seria incluso um *flip-flop* ao módulo lógico da entrada.

Como o projeto possui arquivos em formatos do tipo *SystemVerilog* (SV) é possível ser aberto em outros *softwares*, como por exemplo o Vivado, que possui biblioteca integrada com os dispositivos FPGA produzidos pela empresa Xilinx, de forma que o programa funcione não só com dispositivos FPGA diferentes, mas também como servomotores de outros fabricantes e até estrutura física com outro modelo de chassi e material construtivo.

A estrutura do robô se mostrou um tanto vulneravél quanto a execução dos movimentos, mais especificamente nas peças das patas, que demonstraram certa fragilidade em suportar o peso da estrutura conectada aos servomotores, neste ponto é necessário um ajuste no material da estrutura, ou a realização de reforços estruturais nas peças denominadas coxa e panturrilha.

Para futuros trabalhos, outro ponto interessante a relatar seriam quais as melhorias possíveis de ser implementadas para um maior aproveitamento do projeto, tanto em questão de custo quanto em questão de autonomia, além de ser um incentivo didático. A síntese FPGA também é um fator a ser aprimorado em caso de mudanças na aplicação do projeto para uma de maior complexidade.

Como sugestões diante as inúmeras possibilidades fornecidas pelo uso do FPGA, pode ser apontada a adaptação de mais um grau de liberdade em cada junta, totalizando dezoito atuadores. Outra possibilidade é adicionar um sensor de obstáculos para desvio de objetos. Ainda abordando as sugestões para melhorias em futuros trabalhos, os Kits FPGAs em sua maioria, comporta diferentes tipos de comunicação, entre elas o protocolo RS-232, Ethernet, e portas USB que permite a comunicação via bluetooth, sendo uma opção o controle através de interface em aplicativo android. A placa ainda possui saída de vídeo VGA e áudio, tornando as possibilidades de melhorias e aplicações maiores.

REFERÊNCIAS

ALI, N. A. *et al.* **PWM controller design of a hexapod robot using FPGA**. In: . IEEE International Conference on Control System, Computing and Engineering, 2013. p. 310–314. ISBN 978-1-4799-1508-8. Disponível em: https://ieeexplore.ieee.org/document/6719980. Acesso em: 25 fev. 2022.

ALL DATASHEET. **MG996R Datasheet**. [*S.l.*], 2022. Disponível em: https://html.alldatasheet.com/html-pdf/1131873/ETC2/MG996R/110/1/MG996R.html. Acesso em: 25 ago. 2022.

BANJANOVIC-MEHMEDOVIC, L. *et al.* **Hexapod Robot Navigation Using FPGA Based Controller**. **Lecture Notes in Networks and Systems**, IEEE Xplore, v. 1, p. 1–6, dez 2019. ISSN 2643-1858. Disponível em: https://ieeexplore.ieee.org/document/8939030. Acesso em: 21 set. 2020.

BEKKER, M. G. **"Is the Wheel the Last Word in Land Locomotion?"**. New Scientist nº248, 1961. 64 p. ISSN 0262-4079. Disponível em: https://books.google.com.br/books?id=BJ5eFU5aRecC&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false. Acesso em: 28 set. 2020.

BIANCHI, R. A. C. **Introdução à Robótica**. 2011. Centro Universitário da FEI. Disponível em: https://fei.edu.br/~rbianchi/robotica/. Acesso em: 28 set. 2020.

BÖTTCHER, S. **Principles of robot locomotion**. 2006. Semantic Scholar. Disponível em: https://www2.cs.siu.edu/~hexmoor/classes/CS404-S09/RobotLocomotion.pdf. Acesso em: 28 out. 2020.

DEMASI, D. **Modelagem dinâmica e de controle de um mecanismo de três graus de liberdade para aplicação em um robô hexápode**. 2012. 126 f. Dissertação (Dissertação (Mestrado)) — Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Rio de Janeiro, RJ, 2012.

ERDEN, M. S. **Six legged walking machine: the Robot EA308**. 2006. 157 p. Tese (Doutorado) — Middle East Techinical University, 2006. Disponível em: http://etd.lib.metu.edu.tr/upload/12607356/index. Acesso em: 25 set. 2020.

FERREIRA, E. de P. **Robótica Básica: Modelagem de robôs**. Campinas: R. VIEIRA, S1991, 1991.

FONTOURA, F. M.; NASCIMENTO, L. P. d.; GIOPPO, L. L. **Robô Hexápode Controlado por FPGA**. 2013. 161 p. Dissertação (Trabalho de Conclusão de Curso (Engenharia de Computação)) — Universidade Tecnológica Federal do Paraná, Curitiba, 2013.

FOSTER, H. D. **Trends in functional verification: a 2014 industry study**. 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), p. 13, 2015. Disponível em: https://dvcon-proceedings.org/wp-content/uploads/trends-in-functional-verification-a-2016-industry-study.pdf. Acesso em: 21 set. 2022.

MARCHI, J. **Navegação de Robôs Móveis Autônomos: Estudo e Implementação de Aborgadens**. Universidade Federal de Santa Catarina, 2001. Disponível em: https://repositorio.ufsc.br/xmlui/handle/123456789/81441. Acesso em: 28 set. 2020.

MARTINS, I. de T. E. **Kit FPGA - EE03**. [*S.l.*], 2022. Disponível em: http://www.professoremersonmartins.com.br/site/products/KIT-FPGA-\%252d-EE03.html. Acesso em: 25 out. 2020.

MATARIC, M. *et al.* **Introdução À Robótica**. São Paulo: Editora Unesp/Blucher, 2014. 368 p. ISBN 9788539304905.

PIERI, E. R. de. **Curso de Robótica Móvel**. Florianópolis, 2002. 141 p. Disponível em: https://www.adororobotica.com/CURSO_DE_ROBOTICA_MOVEL.pdf. Acesso em: 21 set. 2020.

RABELO, H. P. **Desenvolvimento de um robô hexápode**. 2015. 140 f. Dissertação (Trabalho de Conclusão de Curso (Tecnologia em Mecatrônica Industrial)) — Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Fortaleza, 2015.

REICHENBACH, M. *et al.* **A New Virtual Hardware Laboratory for Remote FPGA Experiments on Real Hardware**. p. 7, abril 2011. Disponível em: http://worldcomp-proceedings.com/proc/p2011/EEE3354.pdf. Acesso em: 25 set. 2020.

ROBOTICSTODAY. **SILO Series: SILO 6**. Robotics Today, 2018. Disponível em: https://www.roboticstoday.com/robots/silo-6-description. Acesso em: 28 set. 2020.

SATELLITE. **PRO-460**. [*S.l.*], 2021. Disponível em: http://www.satellite-computer.com/content/?990.html. Acesso em: 25 jul. 2022.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots**. [*S.l.*]: 2. ed. Massachusetts: Massachusetts Institute of Technology, 2011.

SILVA, K. R. G. d. **Uma metodologia de Verificacão Funcional para circuitos digitais**. 2007. 121 p. Dissertação (Tese de Doutorado em Engenharia Elétrica) — Universidade Federal de Campina Grande, Campina Grande, 2007.

SILVA, V. B. d. **Ferramenta Web Semiautomática para Geração de Ambientes de Verificação UVM com SystemVerilog**. 2018. 82 p. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal do Pampa, Alegrete, 2018.

SUTHERLAND, S.; MILLS, D. **HDVL += (HDL & HVL) SystemVerilog 3.1 The Hardware Description AND Verification Language**. SNUG San Jose, p. 21, mar 2003. Disponível em: https://sutherland-hdl.com/papers/2003-SNUG-paper_SystemVerilog.pdf. Acesso em: 25 jul. 2022.

TERASIC. **DE2-115 User Manual**. [*S.l.*], 2017. 121 p. Disponível em: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=502&PartNo=4#contents. Acesso em: 25 ago. 2022.

TERASIC. **DE10-Pro SX User Manual**. [*S.l.*], 2020. 209 p. Disponível em: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=248&No=1144&PartNo=4#contents. Acesso em: 25 ago. 2022.

APÊNDICE A – Custos envolvidos no projeto

Acerca dos custos envolvidos no projeto, é necessário um certo adendo, devido ao aumento do preço dos produtos utilizados para a realização do projeto. A Placa utilizada neste projeto é a DE2-115 fabricada pela Terasic e cedida pelo Departamendo de Ciências da Computação deste campus da Universidade, porém outros modelos de Kits se adequeam ao projeto. Há a necessidade de aquisição de servomotores, neste caso, o modelo MG996R fabricado pela Tower Pro, assim como obter uma fonte de alimentação para suprir os servomotores. A fonte de alimentação utiliza teve um custo de R$ 200,00 aproximadamente. Ademais, os servomotores custam em média R$ 50,00 a unidade, sendo necessário doze unidades e mais algumas para substituição em caso de danos que possam vir a ocorrer e comprometer algum dos servomotores. A Tabela 1 apresenta o custo médio de um projeto similar, onde é necessário a aquisição dos equipamentos.

Tabela 1 – Custos envolvidos no projeto.

Descrição	Quantidade	Valor	Total
Kit de Placa de Desenvolvimento FPGA IV EP4CE	1	R$364,46	R$364,46
Servo Motor MG996R Metal Gear Tower Pro	12	R$43,99	R$527,88
Fonte ATX Satellite 400W Pro-460	1	R$200,00	R$200,00
Jumpers, Proboard e Equipamentos	1	R$45,00	R$45,00
Confecção das peças	1	R$150,00	R$150,00
Total			R$1287,34

Fonte: Autoria própria, 2022.