



Centro Federal de Educação Tecnológica de Minas Gerais

Departamento de Mecatrônica

Laboratório de Automação e Sistemas Embarcados

HADLER HENRIQUE TEMPESTA

**DESENVOLVIMENTO DE UM MÓDULO *BALL AND BEAM*
MICROCONTROLADO PARA INTRODUÇÃO AO CONTROLE DE
SISTEMAS DE *LOOP* FECHADO INSTÁVEIS**

**DESENVOLVIMENTO DE UM MÓDULO *BALL AND BEAM*
MICROCONTROLADO PARA INTRODUÇÃO AO CONTROLE DE
SISTEMAS DE *LOOP* FECHADO INSTÁVEIS**

HADLER HENRIQUE TEMPESTA

Projeto de iniciação científica no curso
técnico em mecatrônica pelo Centro Federal
de Educação Tecnológica de Minas Gerais –
CEFET-MG.

ORIENTADOR: Prof. Dr. Juliano Coelho Miranda

CO-ORIENTADOR: Prof. Ms. Daniel Soares de Alcântara

RESUMO

A teoria do controle é uma área de grande importância na engenharia, presente em diversos processos automatizados. No entanto, o ensino de controle na educação tecnológica enfrenta desafios na relação ensino-aprendizagem como a falta de sustentação matemática e poucos recursos que auxiliam na abstração dos algoritmos. Nesse contexto, com o intuito de aproximar teoria e prática no ensino da teoria do controle, este projeto teve como objetivo desenvolver o Módulo Ball and Beam LASE II. Esse tipo de sistema traz um processo de grande dinamicidade e que pode ter uma descrição matemática simplificada, tornando-se uma ótima abordagem para o ensino de controle a nível técnico. O módulo foi elaborado com partes rearranjáveis, de forma que sua geometria pode ser alterada gerando variações mecânicas e consequentemente diferentes processos para serem controlados. As partes são facilmente replicáveis pois foram feitas por uma impressora 3D, sendo compostas de plástico ABS são baratas, leves e resistentes a impactos. O sistema todo é controlado por um Arduino UNO equipado com o microcontrolador ATMEGA 328P, um embarcado de fácil aquisição e utilização. Como atuador foi escolhido um servo motor MG-5010 Tower Pro, capaz de realizar ações com bastante precisão. Para o sensoramento da posição da bola na calha, foi escolhido o sensor de distância a laser VL53L0x, que possui grande precisão e exatidão na leitura assim como imunidade a interferências da luz solar. Foi elaborado um código base em linguagem C++ com todas as configurações para leitura, atuação e funcionamento geral do controlador; foi também implementado o algoritmo Proporcional-Integral-Derivativo (PID) no módulo, dessa forma o usuário se preocupa apenas no estudo e entendimento da sintonização do controle. O objetivo do projeto foi alcançado e o módulo pode ser utilizado para o ensino de controle em sistemas instáveis de malha fechada.

Palavras chave: Módulo *Ball and Beam*; Sistema de Controle; PID; Microcontrolador.

LISTA DE FIGURAS

Figura 1. Malha típica de controle por realimentação.....	9
Figura 2. Modelo genérico de um módulo <i>ball and beam</i>	10
Figura 3. Interface gráfica do <i>ball and beam</i> no laboratório virtual.....	13
Figura 4. Calha do módulo <i>ball and beam</i> elaborada pelo autor COSTA, de M. F.....	14
Figura 5. Módulo <i>ball and beam</i> do autor KLUG, M.	15
Figura 6. Arduino UNO	17
Figura 7. Módulo embarcado do sensor VL53L0X.....	18
Figura 8. Servo motor TowerPro SG-5010	19
Figura 9. Impressora 3D Ender 3-Max.....	20
Figura 10. Ultimaker Cura.....	21
Figura 11. Geração de suportes pelo Ultimaker Cura	22
Figura 12. Peça base da calha do módulo	23
Figura 13. Perfil da seção transversal da calha	23
Figura 14. Interface positiva da conexão entre os elementos da calha.....	24
Figura 15. Interface negativa da conexão entre os elementos da calha	24
Figura 16. Processo de encaixe de duas peças.....	25
Figura 17. Peça central da calha.....	25
Figura 18. Suporte do módulo.....	26
Figura 19. Suportes e peça central da calha	26
Figura 20. Peças de extensão	27
Figura 21. Extremidade neutra.....	28
Figura 22. Extremidades para sensores	28
Figura 23. Peça de fixação do motor	29
Figura 24. Trilho de posicionamento do atuador.....	29
Figura 25. Conjunto de suporte do atuador	30
Figura 26. Braço articulado	30
Figura 27. Peça de conexão entre o braço e a calha.....	31
Figura 28. Uso da peça de conexão entre braço e calha.....	31
Figura 29. Posição dos furos da base do módulo.....	32
Figura 30. Arranjo de peças do módulo <i>ball and beam</i>	32

Figura 31. Variáveis para o estudo de movimento	33
Figura 32. Gráfico das curvas de $f(\alpha) = \sin(\alpha)$ e $g(\alpha) = \alpha$	35
Figura 33. Ação do controlador proporcional	36
Figura 34. Ação do controlador integrativo	37
Figura 35. Ação do controlador derivativo.....	38
Figura 36. Ação do controlador PID	39
Figura 37. Arduino IDE.....	40
Figura 38. Filtro de Kalman	41
Figura 39. Peças finais impressas	43
Figura 40. Módulo <i>ball and beam</i> LASE II.....	44
Figura 41. Calha padrão.....	47
Figura 42. Calha curta.....	48
Figura 43. Calha extra-curta	48
Figura 44. Calha assimétrica.....	49
Figura 45. Arquivo de configurações iniciais	50
Figura 46. Escolha do sensor	51
Figura 47. Parâmetro orientação do motor	52
Figura 48. Exemplo de uso das funções	55

LISTA DE ABREVIATURAS E SIGLAS

3D	– Três dimensões
ABS	– <i>Acrylonitrile butadiene styrene</i>
bps	– Bits por segundo
CCP	– <i>Capture/Compare/PWM</i>
CNC	– <i>Computer numerical control</i>
DC	– Corrente contínua
EJS	– <i>Easy Java Simulations</i>
IDE	– <i>Integrated Development Environment</i>
M4	– Rosca métrica de 4mm
MDF	– <i>Medium-density Fiber.</i>
NC	– <i>Numerical control</i>
PID	– <i>Proportional-integrative-derivative</i>
PWM	– <i>Pulse-width Modulation</i>
STL	– <i>Standart Triangle Language</i>
USART	– <i>Universal synchronous and asynchronous receiver-transmitter</i>

SUMÁRIO

RESUMO	1
LISTA DE FIGURAS	3
LISTA DE ABREVIATURAS E SIGLAS	5
SUMÁRIO	7
1. INTRODUÇÃO	9
1.1. Objetivos.....	11
1.1.1. Objetivo geral.....	11
1.1.2. Objetivos específicos	11
2. REVISÃO BIBLIOGRÁFICA	13
3. DESENVOLVIMENTO	17
3.1. Escolha dos componentes.....	17
3.1.1. O controlador.....	17
3.1.2. Sensoriamento	18
3.1.3. O atuador	19
3.2. Estrutura mecânica.....	20
3.2.1. Processo de impressão 3D	20
3.2.2. Desenvolvimento das peças do módulo	22
3.2.2.1. Geometria base da calha.....	23
3.2.2.2. Suportes e peça central.....	25
3.2.2.3. Peças de extensão	27
3.2.2.4. Peças de extremidade	27
3.2.2.5. Suporte do atuador	29
3.2.2.6. Conexão entre atuador e calha	30
3.2.2.7. Montagem	31
3.3. Estudo de movimento da calha.....	33
3.4. Algoritmo de controle	36
3.5. Código base e bibliotecas.....	40
4. RESULTADOS	43
4.1. Módulo <i>Ball and Beam</i> LASE II	43
4.2. Código base do módulo.....	44
3.3. Arquivos do projeto.....	45
5. GUIA DE USO DO MÓDULO <i>BALL AND BEAM</i> LASE II	47
5.1. Montagem do módulo	47
5.2. Conexões do <i>hardware</i>	49

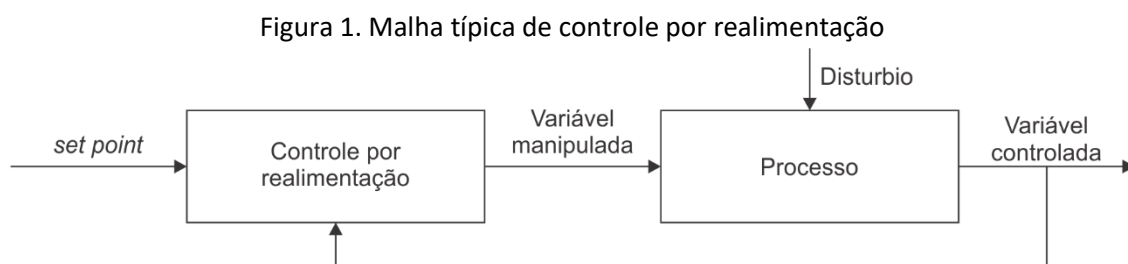
5.3. Configurações iniciais	49
5.3.1. Configurações do PID (<i>PID Settings</i>)	50
5.3.2. <i>Hardware</i>	51
5.3.3. Parâmetros mecânicos (<i>Mechanical</i>).....	52
5.3.4. Firmware	53
5.3.5. Filtro de Kalman	53
5.4. Funções do código.....	53
6. CONCLUSÕES.....	57
7. REFERÊNCIAS.....	59
ANEXO A	61

1. INTRODUÇÃO

Ao longo da história e com o avanço tecnológico, a indústria requisitou processos progressivamente mais velozes, precisos e com alta repetibilidade. Para atender à crescente demanda dessa área, os campos da engenharia e ciência introduziram um conjunto de métodos de análise capazes de analisar uma variável associada a um processo e, por meio de atuadores, realizar as correções necessárias para manter seu funcionamento adequado. Denominada atualmente: teoria do controle clássica, esse campo se tornou essencial nos mais diversos setores da tecnologia, sendo aplicada desde em plantas industriais para o controle de variáveis como pressão, temperatura ou humidade, até sistemas automatizados mais complexos presentes em veículos e na robótica. Diante da importância do controle em sistemas automatizados, é esperado que o profissional da área de mecatrônica tenha conhecimento e familiaridade com teoria e prática do controle automatizado (OGATA, 2010).

O objetivo de um sistema de controle é a estabilização do processo no estado desejado. Esse estado é indicado pela análise da variável controlada. Ela é associada à grandeza no processo que se deseja controlar, e o papel do algoritmo de controle é comparar seu valor a um patamar de referência, denominado set point. A diferença entre a leitura da variável controlada e o valor desejado para ela, indica um erro no processo que deve ser corrigido. Para executar essa correção, o controlador assume uma segunda variável denominada: variável manipulada; à ela está associado um atuador capaz realizar uma ação no processo na tentativa de trazer a grandeza controlada para o valor de referência, que é novamente escrito na variável controlada e o processo se repete até a estabilização. Esse modelo é chamado controle por realimentação, controle em malha fechada ou ainda controle com feedback (GARCIA, 2017).

A Figura 1 mostra a malha típica a um processo de controle em malha fechada.



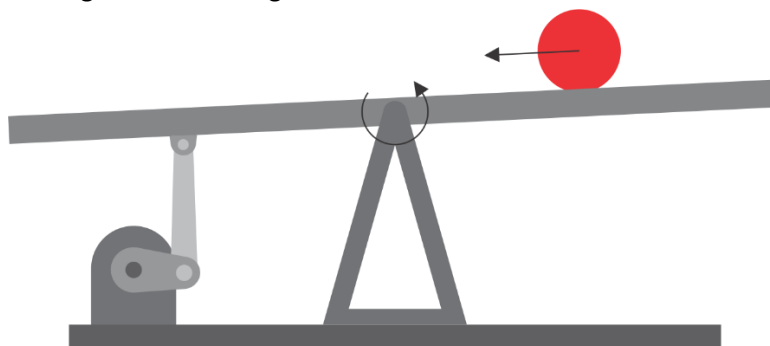
Fonte: (GARCIA, 2017) Modificada pelo autor

O processo de controle com feedback é a técnica de controle mais utilizada na indústria, recebendo muita atenção da comunidade científica com pesquisas, estudos, experimentos e aplicações constantes. A característica que traz essa técnica a tão altos patamares é sua velocidade e capacidade de resposta a estímulos até então desconhecidos. A ação de correção na malha realimentada surge instantes após a oscilação no processo; e com a análise direta da grandeza de interesse a correção acontece não importando a fonte do distúrbio (GARCIA, 2017).

Um processo comumente utilizado pelos profissionais de ensino da área de controle, no primeiro contato entre aluno e disciplina, é o chamado *ball and beam* (HAUSER, SATRY e KOKOTOVIÉ, 1992). Esse sistema – que nome significa: bola e calha – traz um processo com grande dinamicidade mas com descrição matemática que pode ser bastante simplificada. Por essas características, o *ball and beam* se torna uma ótima abordagem para o ensino dos princípios teóricos e práticos de controle, exigindo conceitos de física e matemática do ensino médio com apenas alguns pré-requisitos mínimos de cálculo.

O funcionamento do módulo é simples, uma pequena bola é colocada sobre uma guia de forma que possa rodar livremente, indo de um lado para o outro. A guia por sua vez é conectada a um suporte por um ponto que permite seu livre giro. Por fim são associados um sensor capaz de detectar a posição da bola e um atuador que age na inclinação da calha. O objetivo do algoritmo de controle implementado é manter a bola centralizada na guia. A Figura 2 abaixo mostra um modelo genérico de um módulo *ball and beam*.

Figura 2. Modelo genérico de um módulo *ball and beam*



Fonte: Elaborada pelo autor

Nesse processo, a leitura da posição da bola é a variável controlada, enquanto a inclinação da calha é a variável manipulada. A principal perturbação que a bola sofre é claramente o ganho de velocidade pela ação de seu próprio peso, mas pode também receber

um impulso por alguém ou algo que eventualmente toque-a no intuito de desequilibrar o sistema, ou ainda estar numa base instável. Em resumo, existem inúmeras possíveis interferências que podem acontecer, e por isso o sistema em malha fechada é escolhido para o controle, fazendo a correção da variável controlada independentemente da fonte do erro.

Dentre os algoritmos de controle tradicionais, o controlador Proporcional-Integrativo-Derivativo (PID) é o mais utilizado nos processos automatizados nas indústrias. O controlador PID trabalha calculando o a discrepância entre a variável controlada e o *set point*, o valor encontrado é chamado erro. Esse valor é então processado por três diferentes módulos: o proporcional, integral e derivado; cada um deles é sensível a um tipo de variação do valor analisado e, quando somados, produzem um sinal de saída capaz de amenizar ou eliminar o desvio lido inicialmente de forma rápida e eficiente (CAMPOS e TEIXEIRA, 2010).

1.1. Objetivos

1.1.1. Objetivo geral

Desenvolver um módulo *ball and beam* para auxílio do ensino de controle em sistemas de *loop* fechados aos alunos do ensino técnico profissionalizante.

1.1.2. Objetivos específicos

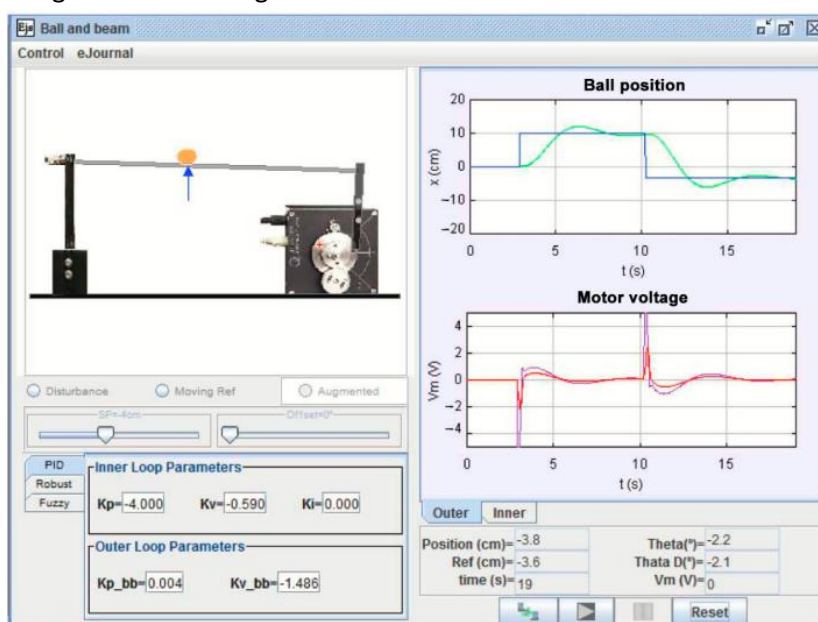
- Escolher o sensoriamento, controlador e atuador adequados.
- Criar uma estrutura mecânica com partes rearranjáveis.
- Estudar o movimento dos elementos do módulo.
- Implementar um algoritmo de controle.
- Elaborar um código base que facilite o uso do módulo.
- Escrever um guia de uso do módulo.

2. REVISÃO BIBLIOGRÁFICA

Nesse tópico são mostradas as abordagens de outros autores sobre o módulo *ball and beam* e suas aplicações práticas ou para o ensino de controle.

Laboratórios virtuais são ferramentas de suma importância na educação a distância e também no complemento ao estudo presencial. Nessa temática pesquisadores elaboraram, por meio da ferramenta EJS (*Easy Java Simulations*) laboratórios virtuais integrados ao sistema Moodle para o auxílio ao ensino de controle. Primeiramente, no laboratório, é montado um módulo convencional de *ball and beam* e uma câmera mirada a ele, ambos são conectados a um computador. O laboratório virtual gera uma interface que mostra a imagem em tempo real gerada pela câmera, plota gráficos de variáveis do sistema e permite que o usuário injete diferentes algoritmos no controlador do sistema de forma remota, através de uma conexão de internet mediada pelo sistema Moodle. Os alunos podem fazer grupos, criando um ambiente interativo com transmissão de voz e vídeo, para testar algoritmos no módulo, verificando a eficiência de diferentes métodos, como o controle PID ou a lógica *Fuzzy*. Uma análise mostrou que o projeto foi bastante efetivo, concluindo que o laboratórios virtuais colaboraram no ensino (DORMIDO et. al, 2015). A Figura 3 mostra a interface elaborada para o laboratório virtual e uso do *ball and beam*.

Figura 3. Interface gráfica do *ball and beam* no laboratório virtual



Fonte: (DORMIDO et. al, 2015)

Um módulo *ball and beam* foi desenvolvido na Universidade Federal de Ouro Preto (UFOP), e teve como objetivo uma análise nos componentes e técnicas necessárias para a montagem de desse equipamento, enfatizando os desafios para a montagem de uma estrutura mecânica e detecção da posição da bola. A solução para montagem mecânica encontrada pelo autor foi a utilização de uma calha em “V” para a movimentação da bola, sendo que nela foram equipadas uma chapa de alumínio e uma resistência de chuveiro; a bola escolhida era composta de ferro e capaz de fechar um contato entre os condutores de modo que um nível de tensão pôde ser injetado num circuito amplificador e lido pelo controlador, revelando um dado proporcional à posição da esfera na calha (COSTA, 2018). A Figura 4 mostra a calha elaborada pelo autor desse projeto.

Figura 4. Calha do módulo ball and beam elaborada pelo autor COSTA, de M. F



Fonte: (COSTA, 2018)

Nesse projeto foram utilizados para o controle do ângulo da calha um motor de corrente contínua (DC) acionado por uma ponte H – circuito eletrônico capaz de permutar o sentido de rotação do motor em questão – controlada pelo sistema embarcado Arduino UNO. O trabalho completou seus objetivos, contudo enfatizou dificuldades na montagem da estrutura por falta de peças adequadas logo no primeiro protótipo (COSTA, 2018).

Um modelo diferente do módulo foi feito na Universidade de Santa Catarina (UFSC), a estrutura base do dispositivo foi feita em MDF (*Medium-density Fiberboard*), enquanto a calha para o deslizamento da bola em acrílico. O atuador do sistema é um servo motor modelo FUTABA S3003, enquanto o sensoramento da posição da esfera é feito por um servo infravermelho modelo SHARP GP2Y0A21YK0F, esse é equipado com uma sávida analógica e faixa de leitura entre, cerca de, 10 a 80 cm (KLUG, 2018). A Figura 5 mostra o módulo elaborado pelo autor do projeto na UFSC.

Figura 5. Módulo ball and beam do autor KLUG, M.



Fonte: (KLUG, 2018)

Nesse trabalho o autor utilizou técnicas refinadas de controle e o algoritmo elaborado foi executado por um computador através do ambiente Matlab, recebendo os dados de leitura do sensor a partir de um Arduino UNO conectado ao sistema (KLUG, 2018).

3. DESENVOLVIMENTO

3.1. Escolha dos componentes

O processo de desenvolvimento se iniciou com a escolha dos três principais componentes do módulo: o controlador, o sensoriamento e o atuador. Pois foi em torno da determinação desses itens que a estrutura mecânica pôde ser projetada.

3.1.1. O controlador

Para o controlador do sistema, procurou-se um dispositivo de fácil aquisição e uso; diante de suas vantagens foi utilizado o sistema embarcado Arduino UNO.

A Figura 6 mostra a vista superior de um Arduino UNO.

Figura 6. Arduino UNO



Fonte: (ARDUINO UNO REV3, 2020)

Esse embarcado é equipado com o microcontrolador ATMEGA 328P e amplamente utilizado pela comunidade *Maker* – composta por projetistas e entusiastas da área de tecnologia – existindo portanto grande suporte a possíveis problemas, muitas bibliotecas disponíveis para uso e um grande leque de módulos compatíveis com ele como: sensores, atuadores, conversores, expansores, etc. O dispositivo conta ainda com uma interface que facilita sua programação: a Arduino IDE¹; esse software se encarrega da configuração dos registradores internos do controlador, permitindo que o projetista possa desenvolver seus algoritmos em alto nível por meio da linguagem de programação C++.

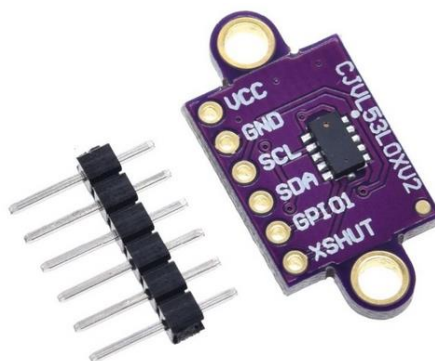
¹ **Arduino IDE:** Disponível em: <<https://www.arduino.cc/en/software>>. Acesso em: 31 dez. 2020.

3.1.2. Sensoriamento

Durantes as pesquisas e testes de métodos de detecção da bola na calha, verificou-se que devido a sua geometria esférica e pequeno tamanho, essa tarefa poderia ser complicada. Por esse motivo foi procurado um sensor de maior precisão, capaz de realizar a leitura de forma e constante e que sofre-se mínima interferência por oscilações em sua grandeza de medição. Após pesquisa foi escolhido o módulo embarcado do sensor a laser VL53L0x.

A Figura 7 mostra a visão superior de um módulo embarcado VL53L0X.

Figura 7. Módulo embarcado do sensor VL53L0X



Fonte: (SENSOR DE DISTÂNCIA VL53L0X, 2020)

Esse módulo vem equipado o circuito integrado VL53L0x, um sensor da nova geração *Time-of-Flight* (ToF) a laser, acoplado no menor encapsulamento presente hoje no mercado e capaz de fazer precisas leituras independentemente da refletância da superfície lida. O laser emitido é invisível aos olhos humanos e tem alta imunidade a iluminação ambiente (VL53L0X, 2018).

O módulo é compatível com o microcontrolador escolhido estabelecendo uma comunicação serial com protocolo de comunicação I2C (*Inter-Integrated Circuit*), com auxílio da biblioteca nativa à Arduino IDE chamada: "Wire.h". Enquanto a geração dos pacotes de requisição de dados assim como a interpretação dos pacotes vindos do sensor foi feita pela biblioteca oficial disponibilizada pela empresa desenvolvedora: © 2001-2020 Pololu Corporation.

3.1.3. O atuador

Para o atuador do sistema, foi procurado um dispositivo que pudesse ser controlado de forma simples pelo Arduino UNO e que, simultaneamente, oferecesse boa precisão e repetibilidade. Após pesquisas a escolha foi por um atuador do tipo servo motor e o modelo optado foi o Servo Motor TowerPro SG-5010.

A Figura 8 mostra um servo motor modelo TowerPro SG-5010.

Figura 8. Servo motor TowerPro SG-5010



Fonte: (TOWERPRO SG-5010, 2020)

O servo é composto internamente por engrenagens de fibra de carbono tornando-o mais leve que modelo com engrenagens metálicas. Se trata de um servo motor digital, controlado por um sinal PWM (*Pulse-width modulation*) equipado com um eixo rotativo com grau de liberdade de 180°. Os circuitos internos garantem um dispositivo com bom torque rápidas respostas a forças externas (TOWERPRO SG-5010, 2020).

O servo pôde ser facilmente controlado pelo Arduino UNO por meio de um de seus módulos CCP (*Capture/Compare/PWM*), ativado com auxílio da biblioteca nativa: "Servo.h". Para sua devida utilização foi também necessária uma pequena fonte DC 5V para alimentá-lo, já que o circuito de alimentação do Arduino UNO não é capaz de fornecer, sozinho, suficiente potência para o motor.

3.2. Estrutura mecânica

Após pesquisas e análise dos possíveis materiais para a montagem do módulo, foi concluído que a melhor opção seria uma manufatura total de todos os elementos mecânicos. O processo escolhido para criação dessas peças foi o processo de impressão 3D por deposição de camadas, utilizando para isso uma impressora 3D CNC. A escolha desse método de fabricação influenciou completamente no planejamento e, principalmente, moldagem das peças do módulo pra que fossem não somente compatíveis mas também otimizadas pra o processo de impressão 3D.

3.2.1. Processo de impressão 3D

No processo de impressão 3D escolhido, deposição de camadas, é utilizada uma máquina CNC (*Computer numerical control*) denominada impressora 3D. A impressora 3D Ender 3-Max é mostrada na Figura 9 a nível de exemplificação.

Figura 9. Impressora 3D Ender 3-Max



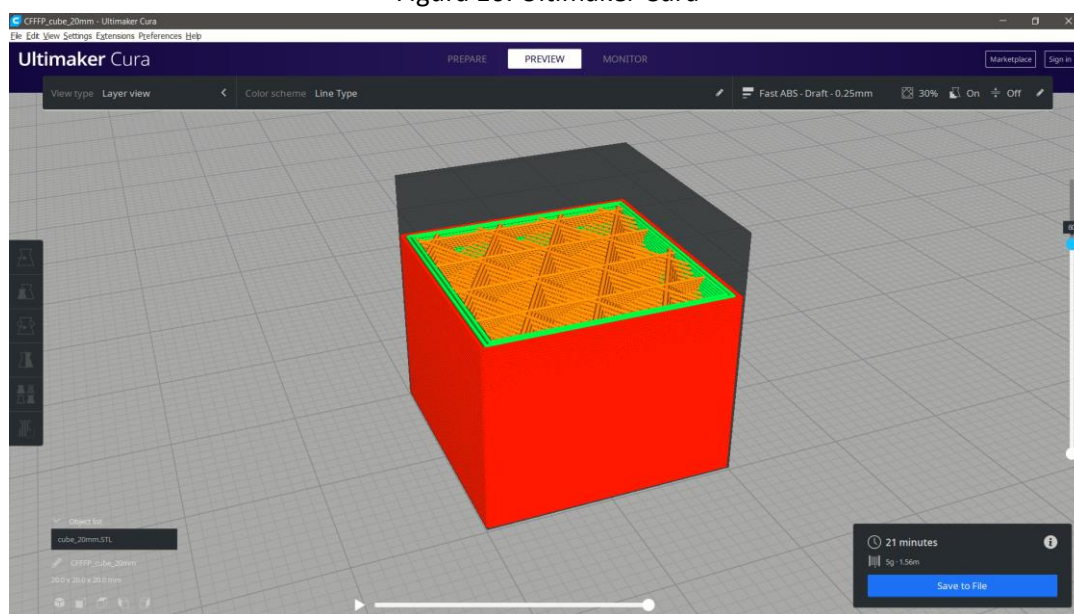
Fonte: (ENDER-3 MAX 3D PRINTER, 2020)

Resumidamente, no processo de deposição de camadas, a peça desejada é interpretada por um tipo de software chamado *slicer*, o papel dele é subdividir a peça em diversas camadas muito finas, na ordem de décimos de milímetros – um valor muito usual é de 0,2mm de espessura. Após geradas as camadas, elas são convertidas em instruções no formato NC (*Numerical control*) formando um código em linguagem GCode, que pode ser lido pela impressora. O material escolhido é então fundido pela máquina e depositado nas posições necessárias – camada a camada – até que a peça seja impressa por completo.

Para o projeto, o *slicer* utilizado foi o Ultimaker Cura 4.2.0². Populamente conhecido como “Cura” esse *software* é gratuito e amplamente utilizado pela comunidade de impressão 3D por sua vasta disponibilidade de recursos e alta tecnologia na geração do GCode, trazendo resultados muito satisfatórios em qualquer impressora 3D.

A Figura 10 mostra a interface do Cura e a pré-visualização das camadas geradas para a impressão de um cubo de 8cm³ de volume. Na imagem, são mostradas as primeiras 60 camadas de 80 presentes no processo. Cada cor representa um diferente processo: vermelho – paredes externas, verde – paredes internas, e laranja – preenchimento.

Figura 10. Ultimaker Cura



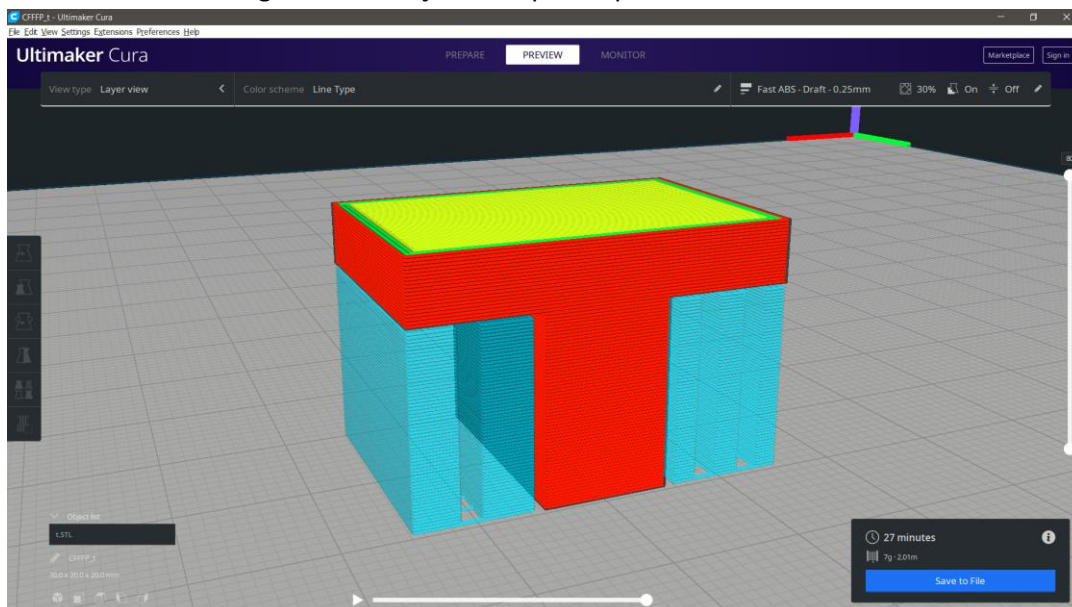
Fonte: Elaborada pelo autor

Com a deposição de material camada a camada, é ideal, embora não estritamente essencial, que as paredes externas não tenham nenhum ângulo menor que 45° em relação à base, pois isso causa uma falta de suporte para deposição do material. Caso isso aconteça o software gerará suportes que serão impressos juntamente com a peça, embora resolvam o problema, esses suportes aumentam o tempo de impressão e perda de material.

A Figura 11 mostra a geração de suportes – em azul – numa peça em formato de “T” pelo Cura quando não existe apoio, pelas camadas inferiores, para a impressão de uma nova camada.

² **Ultimaker Cura.** Disponível em: <<https://ultimaker.com/software>>. Acesso em: 31 dez. 2020.

Figura 11. Geração de suportes pelo Ultimaker Cura



Fonte: Elaborada pelo autor

Baseado nisso, a modelagem das peças para o módulo procurou evitar geometrias que necessitariam a geração de suportes durante a impressão, embora em alguns casos isso não tenha sido possível e suportes tiveram que ser utilizados.

O último passo foi a escolha do material que seria utilizado na confecção das peças. Após considerações foi escolhido o plástico ABS (*Acrylonitrile butadiene styrene*), especificamente o ABS Premium+ da empresa 3D Fila. Esse material tem como principais características a resistência térmica e a impactos, assim como durabilidade, ele foi especialmente desenvolvido para o uso em impressoras 3D na confecção de peças funcionais e complexas (FILAMENTO ABS PREMIUM+, 2020).

3.2.2. Desenvolvimento das peças do módulo

Uma vez que foram escolhidos os componentes do módulo e o processo a ser utilizado para a criação de suas partes, foi iniciado o processo de desenvolvimento da estrutura mecânica em si. Para melhor compatibilidade com o processo de impressão 3D, foi optado a criação das peças pelo método de modelagem 3D. Para isso foi utilizado o *software* SOLIDWORKS³, ele possui uma grande quantidade de recursos para a criação de peças em 3D;

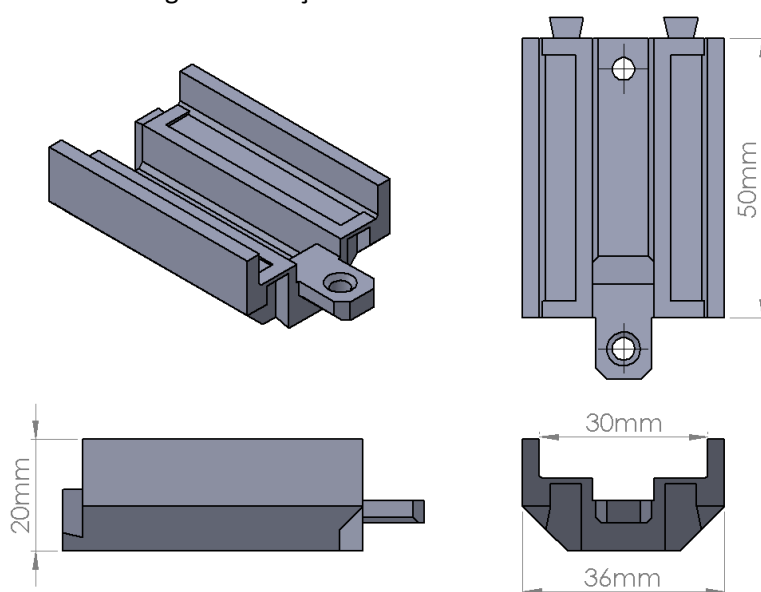
³ **SOLIDWORKS**: Disponível em: <<https://www.solidworks.com/pt-br>>. Acesso em: 31 dez. 2020

além de permitir que sejam exportadas para o formato STL (*Standard Triangle Language*), que pode ser lido pelo Cura para geração do GCode.

3.2.2.1. Geometria base da calha

O objetivo da estrutura mecânica foi a criação de peças modulares rearranjáveis que permitissem ao usuário montar diferentes processos. A modelagem se iniciou com a criação de um modelo base para a calha e uma interface de encaixe entre as peças. O resultado foi uma pequena peça que serviu de base para as outras que seriam criadas. Ela é mostrada na Figura 12.

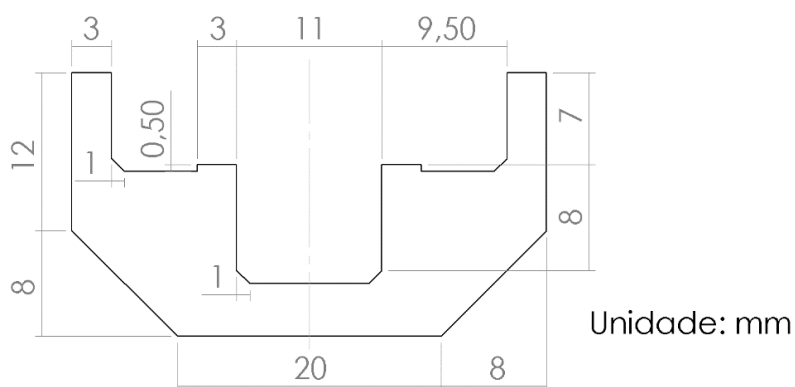
Figura 12. Peça base da calha do módulo



Fonte: Elaborada pelo autor

A Figura 13 mostra com detalhes as dimensões do perfil desenhado para a calha do módulo.

Figura 13. Perfil da seção transversal da calha

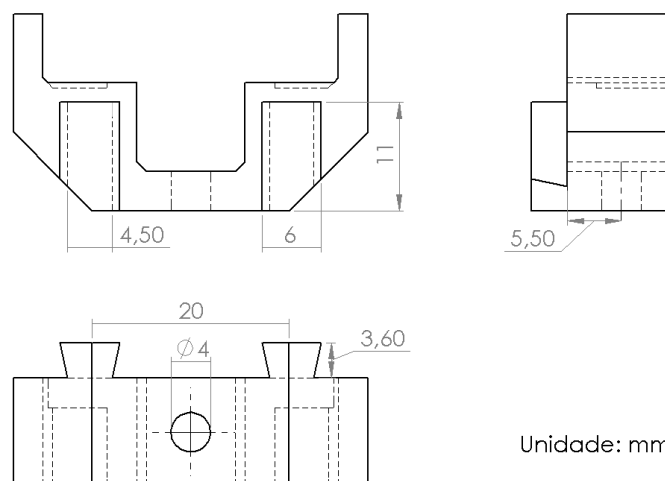


Fonte: Elaborada pelo autor

Na peça foram criadas duas interfaces de conexão, onde as outras partes poderiam ser conectadas futuramente. A interface de conexão consistiu num encaixe do tipo “rabo de andorinha” para posicionamento e estabilização, e um pino com furo para parafuso, o objetivo dessa segunda é fixação final das peças.

Cada extremidade recebeu uma das duas partes dessa interface de conexão, que foram apelidadas como interface positiva e interface negativa. A Figura 14 mostra em detalhes a interface positiva.

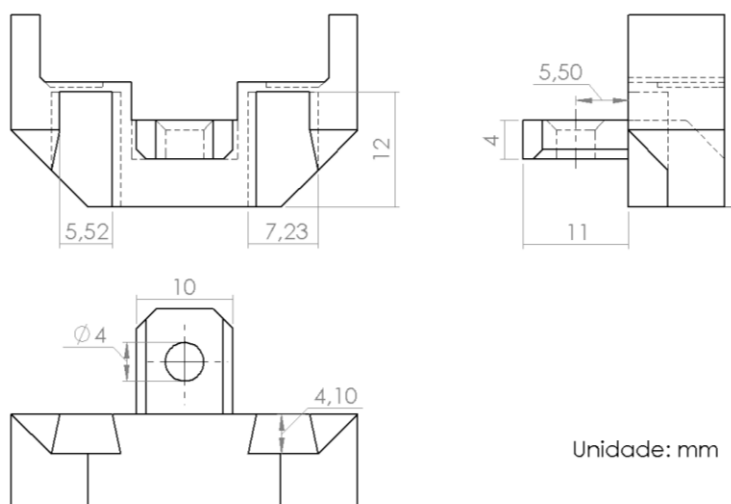
Figura 14. Interface positiva da conexão entre os elementos da calha



Fonte: Elaborada pelo autor

Já a Figura 15 mostra os detalhes e principais medidas da interface negativa de conexão.

Figura 15. Interface negativa da conexão entre os elementos da calha

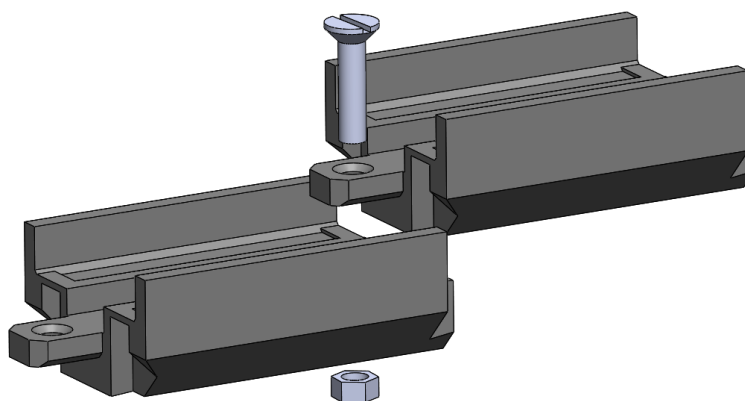


Fonte: Elaborada pelo autor

O furo para parafuso possui um diâmetro de 4mm, com a intenção de utilização de parafusos métricos de rosca M4, embora também sejam compatíveis os parafusos imperiais com rosca de 5/32”.

Para realizar o encaixe entre duas peças, basta deslizar uma interface negativa sobre uma positiva e inserir um parafuso adequado pelo furo, apertando uma porca na parte inferior. O processo é sugerido na Figura 16.

Figura 16. Processo de encaixe de duas peças

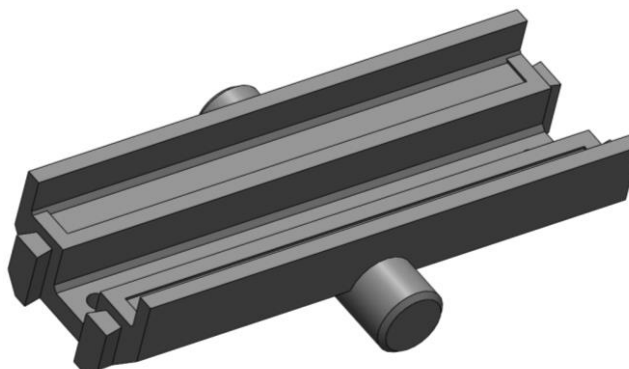


Fonte: Elaborada pelo autor

3.2.2.2. Suportes e peça central

Com um modelo para a calha pronto e uma interface de encaixe funcional, o próximo passo foi a confecção do suporte da calha e sua peça central. A peça central segue o modelo de perfil criado para a calha, foi feita com uma extensão de 100mm e recebeu, nas duas extremidades, interfaces de encaixe positivas. Além de duas extrusões cilíndricas na laterais cujo objetivo é encaixe nos suportes. A Figura 17 mostra a peça central.

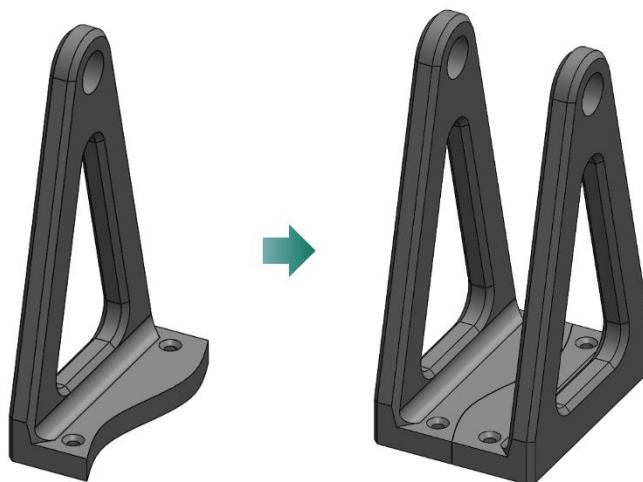
Figura 17. Peça central da calha



Fonte: Elaborada pelo autor

O suporte desenvolvido é composto por duas peças iguais num formato triangular, suas bases tem um formato que garante o perfeito encaixe entre as partes, a peça desenvolvida, assim como o encaixe entre duas delas, é mostrado abaixo na Figura 18.

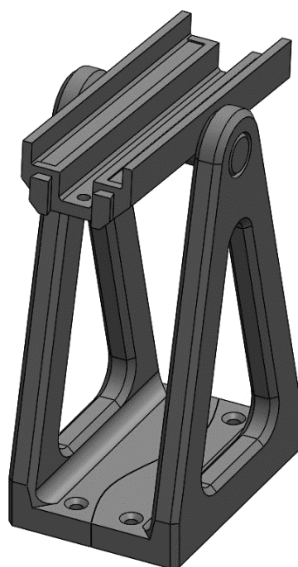
Figura 18. Suporte do módulo



Fonte: Elaborada pelo autor

A montagem da peça central nos suportes é mostrada na Figura 19, basta encaixar os pinos laterais nos furos dos suportes e a estrutura ficará apta ao giro.

Figura 19. Suportes e peça central da calha

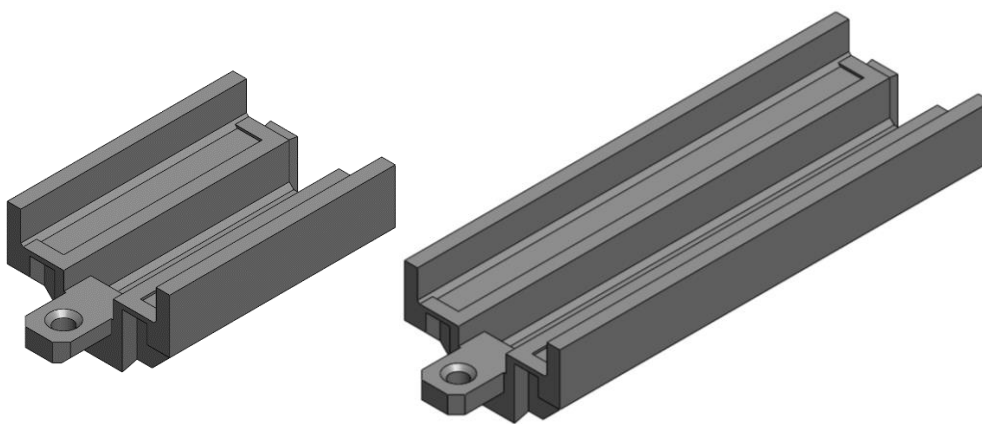


Fonte: Elaborada pelo autor

3.2.2.3. Peças de extensão

As peças de extensão foram projetadas com o objetivo de permitir a montagem de diferentes calhas. Elas são peças simples, com o perfil padrão da calha e compostas por uma interface de conexão positiva e uma negativa; elas foram feitas em duas dimensões, uma com 50mm e outra com 100mm de comprimento. No projeto foi optado pela manufatura de duas de cada uma, equipando o módulo com 4 peças neutras para aumento do comprimento da calha. As extensões são mostradas na Figura 20.

Figura 20. Peças de extensão



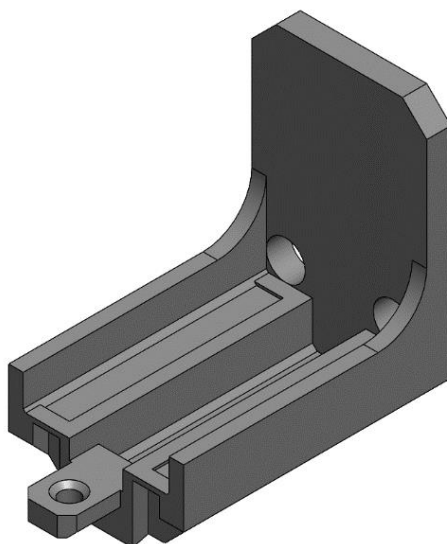
Fonte: Elaborada pelo autor

3.2.2.4. Peças de extremidade

As peças da extremidade da calha tem funções importantes; além de bloquear a bola de escapar pelas pontas, essas peças também servem de apoio para o sensor do módulo. Por isso, a geometria delas é um pouco diferente, mas não muito distante das demais. Foram feitas 4 variantes, todas equipadas com uma interface de conexão negativa, comprimento de 50mm, uma retenção para a bola e, em 3 delas, um suporte para sensor.

A única peça que não contém um suporte para sensor foi chamada de extremidade neutra, ela foi desenvolvida simplesmente para restringir a bola de sair pela ponta da calha oposta ao sensor, pensada para ser utilizada em conjunto com outra extremidade. Essa peça é mostrada na Figura 21.

Figura 21. Extremidade neutra

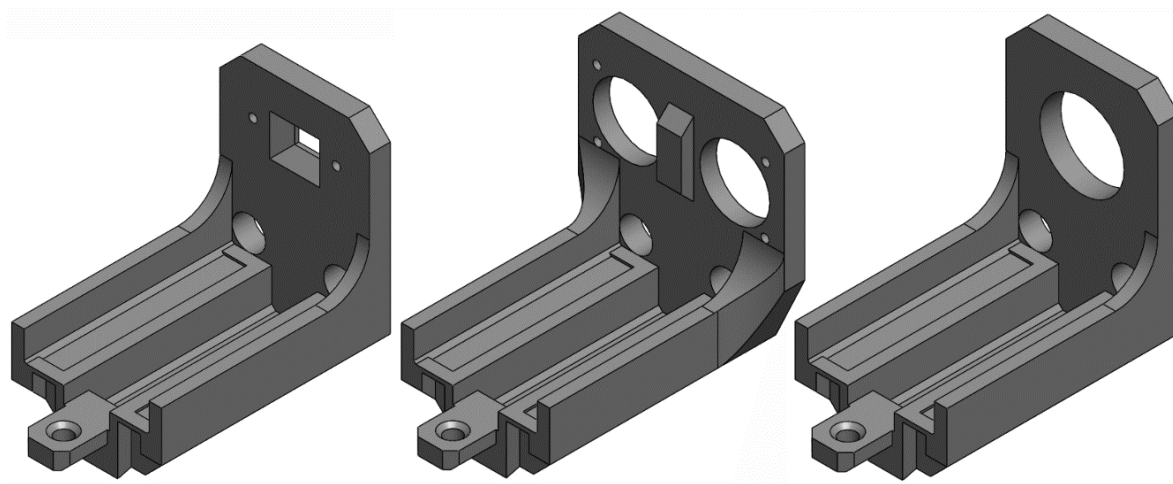


Fonte: Elaborada pelo autor

A segunda extremidade foi elaborada para suportar o sensor escolhido no projeto, o sensor infravermelho VL53L0X. Uma terceira extremidade foi desenvolvida para acomodar um sensor do tipo ultrassônico modelo HC-SR04, embora esse sensor não seja alvo de estudo no projeto é um módulo muito utilizado em projetos de robótica e conhecido pela comunidade, concluindo-se aditiva sua compatibilidade com o módulo. Na temática de possíveis expansões, a quarta extremidade recebeu apenas um furo circular genérico, compatível com o encapsulamento de muitos sensores infravermelhos comerciais, prevendo o possível uso de outros tipos de sensores.

A extremidades para o sensor VL53L0X, para o sensor HC-SR04 e genérica são mostradas, respectivamente, da esquerda para a direita na Figura 22.

Figura 22. Extremidades para sensores



Fonte: Elaborada pelo autor

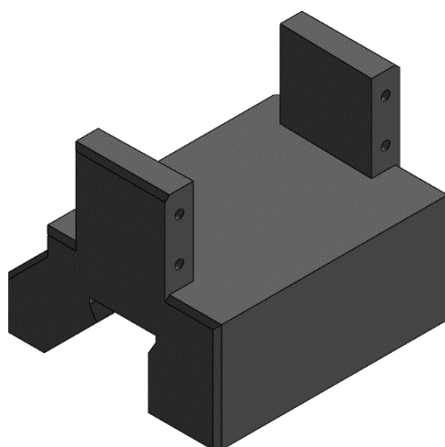
Nota-se que todas as extremidades possuem dois pequenos furos na parte inferior. O objetivo desses furos é a inserção de barras metálicas de 8mm de diâmetro, as barras podem ser colocadas por toda a extensão da calha criando um trilho uniforme e com mínimo atrito para o trânsito da bola.

3.2.2.5. Suporte do atuador

Foram feitas duas peças de suporte para o atuador, em uma delas o motor pode ser aparafusado e a outra é um espécie de trilho onde a primeira pode correr. Esse modelo permite que o motor possa ser movimentado de um lado para o outro alterando seu impacto no sistema.

A peça de fixação do motor é mostrada abaixo na Figura 23.

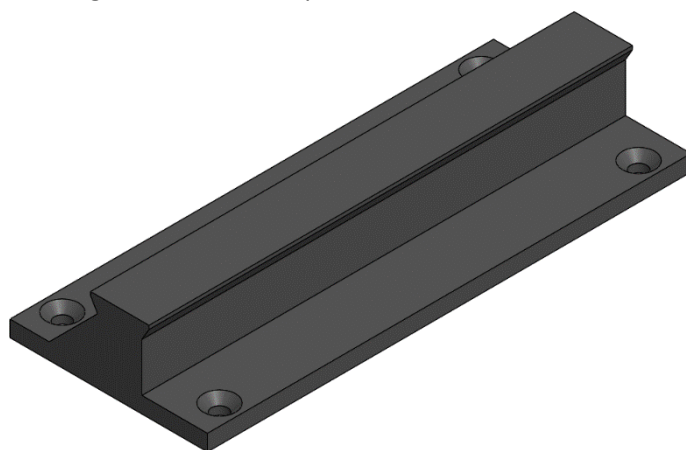
Figura 23. Peça de fixação do motor



Fonte: Elaborada pelo autor

A segunda peça é o trilho para o deslizamento do conjunto do motor, mostrada na Figura 24.

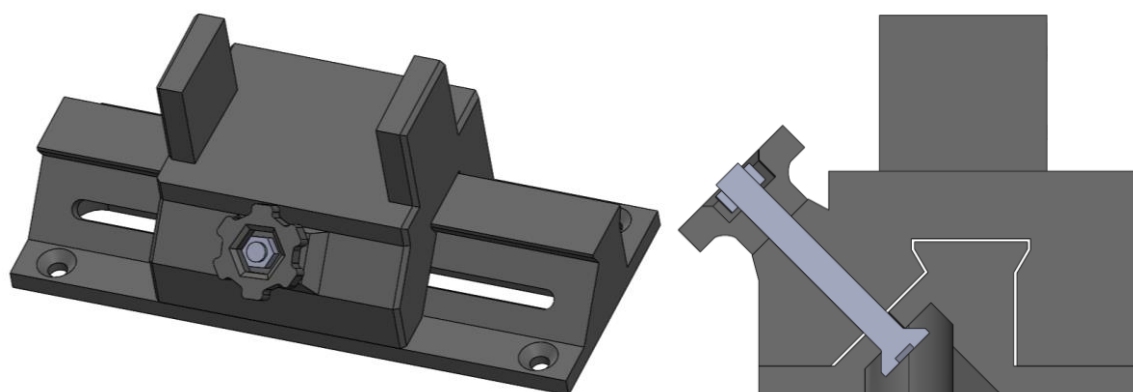
Figura 24. Trilho de posicionamento do atuador



Fonte: Elaborada pelo autor

Para a montagem é colocado um parafuso M4 ou 5/32", com cerca de 35mm de comprimento, passando da base até o suporte do motor, na outra extremidade é colocada uma porca que pode ser associada à pequena peça desenvolvida, ela permite o aperto sem a necessidade de uma chave específica. Uma vez que a posição do motor foi escolhida basta apertar a porca e o sistema ficará estático. A Figura 25 mostra a vista traseira do conjunto montado e uma vista em corte de como posicionar o parafuso;

Figura 25. Conjunto de suporte do atuador

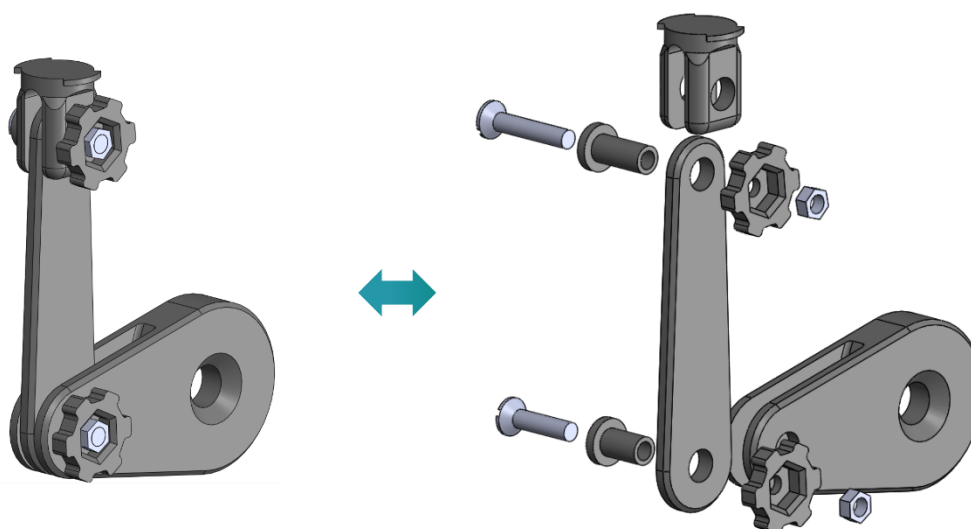


Fonte: Elaborada pelo autor

3.2.2.6. Conexão entre atuador e calha

A conexão entre o atuador e calha foi composta por diversas pequenas peças que juntas formaram um pequeno braço articulado. A Figura 26 mostra a montagem do braço e uma visão explodida com todas as peças, incluindo dois parafusos e duas porcas M4.

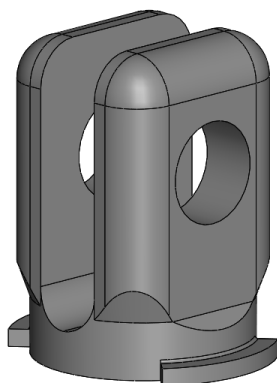
Figura 26. Braço articulado



Fonte: Elaborada pelo autor

A peça que faz conexão com a calha em si é a mostrada na Figura 27.

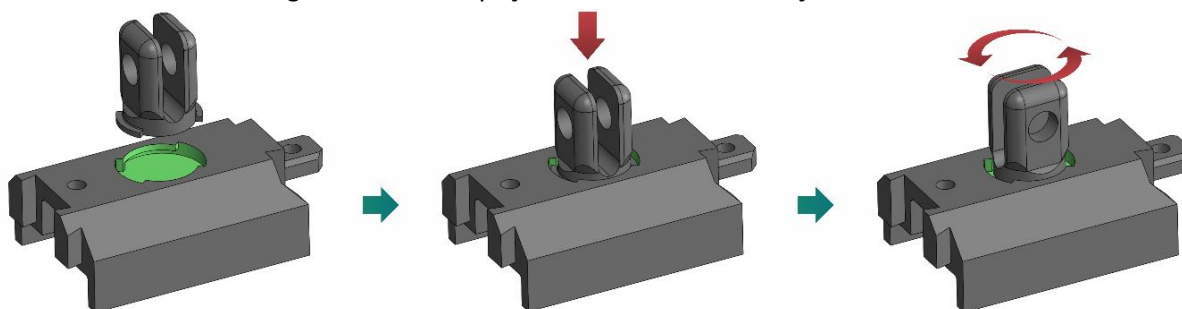
Figura 27. Peça de conexão entre o braço e a calha



Fonte: Elaborada pelo autor

Em cada parte já mostrada que compõe a calha – peça central, extensões e extremidades – foi colocado na parte inferior um encaixe que suporta essa peça de conexão. A Figura 28 mostra o encaixe – em verde – na peça de extensão de 50mm; e também como colocar a conexão no encaixe: bastando fazer o alinhamento com o furo, inserir a peça e girá-la em 90°.

Figura 28. Uso da peça de conexão entre braço e calha

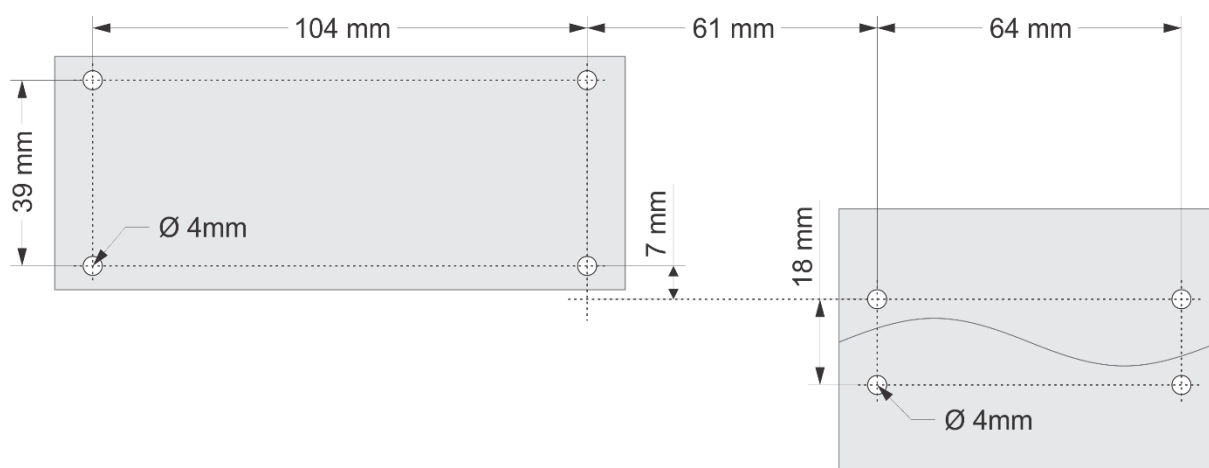


Fonte: Elaborada pelo autor

3.2.2.7. Montagem

Com todas as peças feitas, é finalmente projetada uma base que suporte todos os elementos, a base pode ser composta de qualquer material, bastando ter a furação mostrada na Figura 29, que garante o perfeito posicionamento dos suportes e trilho do motor de forma que o restante das peças se encaixem sem problemas.

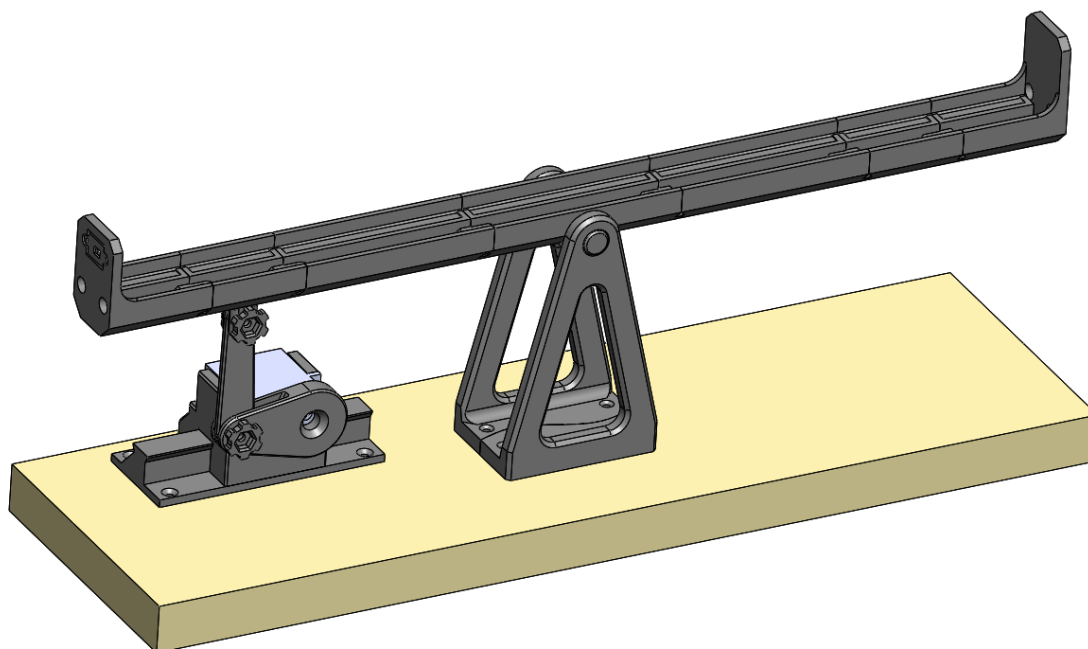
Figura 29. Posição dos furos da base do módulo



Fonte: Elaborada pelo autor

Com a base adequadamente perfurada, uma possibilidade de montagem do módulo seria a mostrada na Figura 30 abaixo.

Figura 30. Arranjo de peças do módulo *ball and beam*

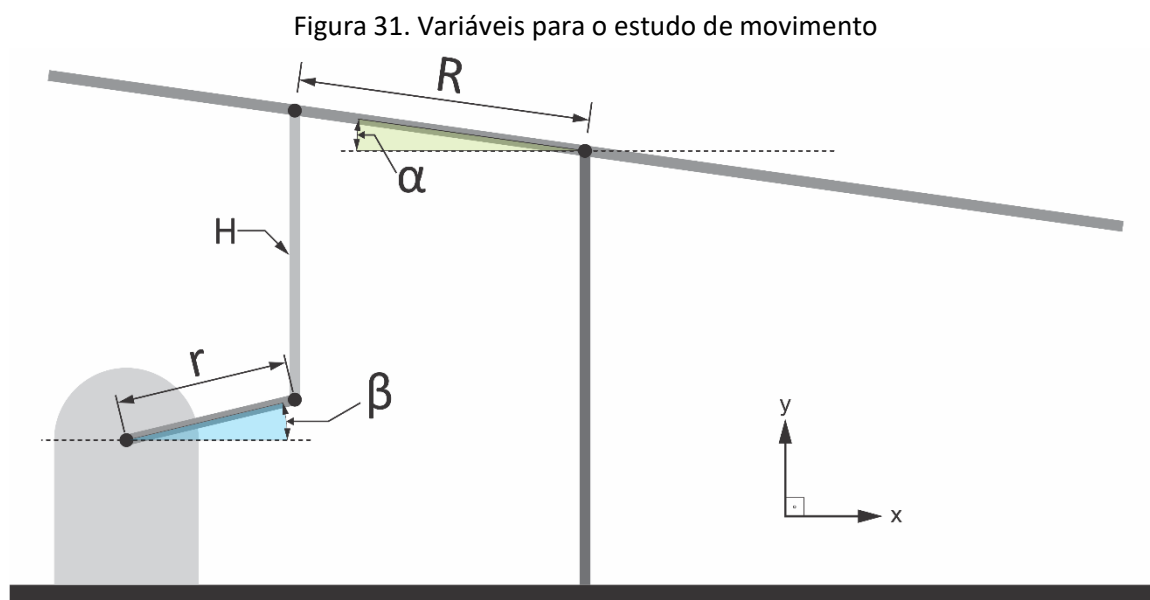


Fonte: Elaborada pelo autor

3.3. Estudo de movimento da calha

O estudo da movimentação da calha teve como objetivo a modelagem de uma equação capaz de descrever, continuamente, a ação do atuador do sistema em função da saída desejada na variável manipulada: a inclinação da calha.

O desenvolvimento dessa relação partiu da análise das variáveis mostradas na Figura 31, que traz um modelo simplificado do conjunto atuador-calha no módulo *ball and beam*.



Fonte: Elaborada pelo autor

Onde: (r) – “Raio do motor”. Raio do extensor acoplado ao eixo do motor;

(R) – “Raio da calha”. Distância entre o ponto de rotação e conexão ao braço articulado;

(α) – Ângulo da calha em relação à direção x;

(β) – Ângulo do motor em relação à direção x;

(H) – Conexão entre atuador e calha.

O primeiro passo do processo de análise foi observar a amplitude de oscilação possível para α . Após observações no *software* de modelagem SOLIDWORKS, foi visto que a própria estrutura mecânica limitaria a oscilação num intervalo de cerca de 15° em ambas as direções, matematicamente definido por: $Dom(\alpha) = \left[-\frac{\pi}{12} rad, \frac{\pi}{12} rad\right]$. Portanto: $\{\alpha \in \mathbb{R}; |\alpha| \leq \frac{\pi}{12}\}$.

Na amplitude encontrada, foi possível assumir que o movimento da peça H seria resumido à translações na direção y. Pois, após observações do movimento, verificou-se que ela permanecia praticamente vertical para a amplitude no domínio de α . Com essa

aproximação foi possível escrever que as projeções ortogonais de r e de R na direção y eram iguais em módulo. A relação vetorial é sugerida na equação 1.1.

$$\left| \text{proj}_{\vec{y}} r \right| = \left| \text{proj}_{\vec{y}} R \right| \quad (1.1)$$

A relação acima pôde ser desenvolvida em função dos ângulos assumidos, gerando a relação algébrica mostrada na equação 1.2.

$$r \cdot \text{sen}(\beta) = R \cdot \text{sen}(\alpha) \quad (1.2)$$

A ação restante foi uma manipulação algébrica afim de formar uma função f de α em função de β , matematicamente definida por: $(f: \alpha \rightarrow \beta)$. O resultado é mostrado abaixo na equação 1.3.

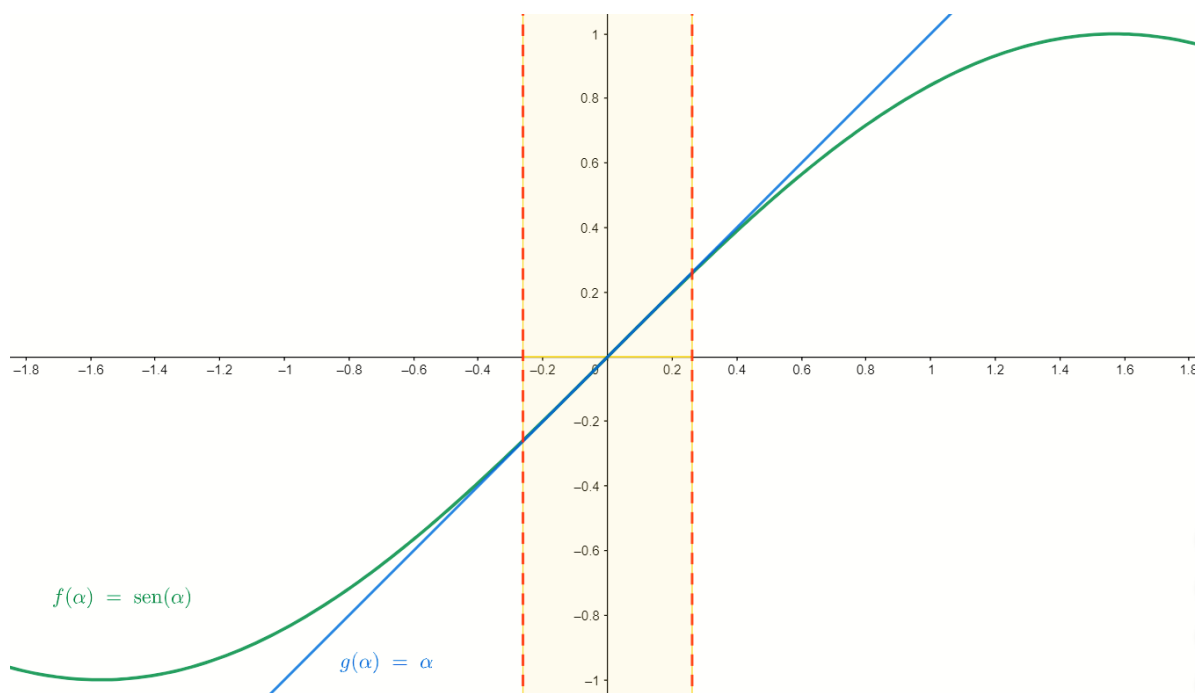
$$\beta = \arcsen\left(\frac{R}{r} \cdot \text{sen} \alpha\right) \quad (1.3)$$

Devido às funções trigonométricas na função, a entrada de α deverá ser feita em radianos e β será retornado também em radianos.

No entanto, em microcontroladores é preferível a utilização de funções mais simples e com menor número possível de conversões de unidade, isso se deve à seu processador de núcleo único e memória limitada, a otimização do código traz benefícios como diminuição no tempo de resposta. Por esse motivo, a função gerada foi estudada no domínio inicialmente definido para α ($Dom(\alpha)$).

Primeiramente foi analisada a função trigonométrica que aparece mais internamente na equação – $\text{sen}(\alpha)$. Quando plotadas as funções $f(\alpha) = \text{sen}(\alpha)$ e $g(\alpha) = \alpha$, no domínio dos reais (\mathbb{R}), foi notada uma grande semelhança no intervalo inicialmente definido como domínio de α . As curvas são mostrados no gráfico da Figura 32.

Figura 32. Gráfico das curvas de $f(\alpha) = \sin(\alpha)$ e $g(\alpha) = \alpha$



Fonte: Elaborada pelo autor

Essa condição tornou válida a aproximação: $(\sin(\alpha) = \alpha; |\alpha| \leq \frac{\pi}{12})$ e simplificou a função inicial para o modelo mostrado na equação 1.4.

$$\beta = \arcsen\left(\frac{R}{r} \cdot \alpha\right) \quad (1.4)$$

Um processo de aproximação foi tentado para a função $f(\alpha) = \arcsen\left(\frac{R}{r} \alpha\right)$, mas, devido ao coeficiente angular no argumento da função trigonométrica (R/r) , houveram muitas discrepâncias e a função original foi mantida.

Por fim foi adicionado um coeficiente linear θ à função, prevendo que a posição zero do motor não seja na direção de x. O valor de θ deverá ser dimensionado de forma que $\beta(0) = \theta$, ou seja, θ é o valor de β no qual a calha fica com inclinação 0.

A função final é mostrada na equação 1.5. É válido lembrar que as variáveis R e r, devem ser inseridas na mesma unidade de comprimento (m, cm, mm, etc.) e as demais variáveis deverão ser utilizadas em radianos (rad).

$$\beta(\alpha) = \arcsen\left(\frac{R}{r} \cdot \alpha\right) + \theta \quad (1.5)$$

Caso o usuário prefira utilizar a equação com as todas as variáveis de ângulo em graus (°), deverá ser utilizada a variação mostrada abaixo da equação (1.6).

$$\beta(\alpha) = \frac{180}{\pi} \cdot \arcsen\left(\frac{R}{r} \cdot \frac{\pi}{180} \cdot \alpha\right) + \theta \quad (1.6)$$

3.4. Algoritmo de controle

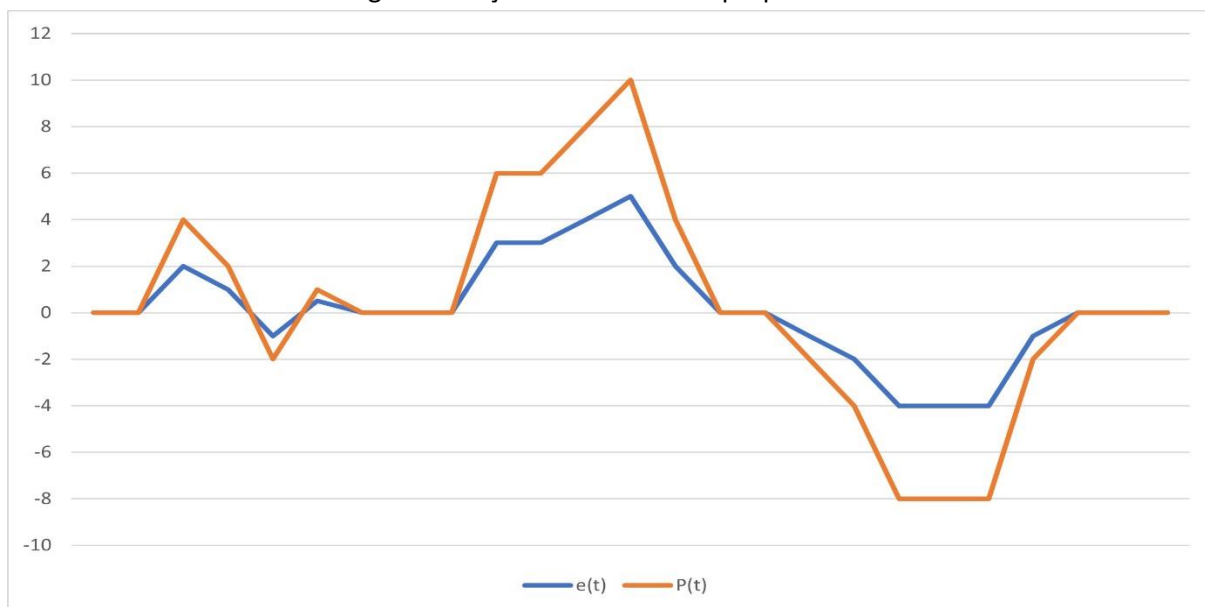
Para o algoritmo de controle foi escolhida a abordagem mais tradicional: o controle PID. Cada letra da sigla representa um dos três módulos de controle, sendo eles: proporcional, integral e derivado. O controlador PID trabalha em malha fechada, baseando-se no erro em função do tempo ($e(t)$), isso é, a diferença entre a variável controlada e o valor de referência esperado para ela, denominado *set point*.

O controlador proporcional produz uma saída, como o nome sugere, proporcional ao erro detectado, geralmente é associada a essa saída um fator multiplicativo chamado k_p . A equação geral da função erro proporcional ($P(t)$) é mostrada abaixo no item 2.0.

$$P(t) = k_p \cdot e(t) \quad (2.0)$$

A Figura 33 mostra uma curva de erro de uma série de dados com *set point* = 0, em azul, e a correção feita pela ação proporcional, em laranja, com um ganho $k_p = 2,0$.

Figura 33. Ação do controlador proporcional



Fonte: Elaborada pelo autor

O controlador integrativo remete a operação do cálculo chamada integração. Seu objetivo é a memorização de erros passados, dessa forma o sistema responderá mais rapidamente a oscilações parecidas, pois a saída tende a convergir para o valor de estabilização. Esse armazenamento de dados é feito pela integração da curva de erro do momento inicial até o atual. Esse valor normalmente recebe um fator multiplicativo chamado k_i . A equação geral do erro integral $I(t)$ é mostrada no item 2.1.

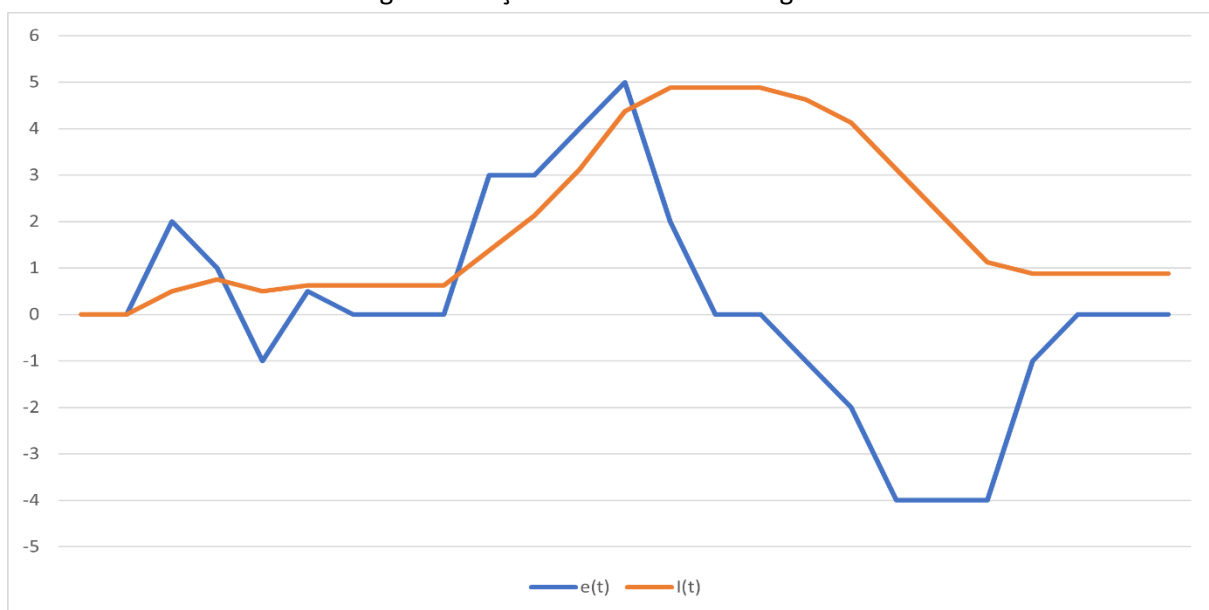
$$I(t) = k_i \int_0^t e(u) du \quad (2.1)$$

No entanto, para o uso da operação integrativa é necessário conhecimento da função da função do erro, o que não acontece nos sistemas instáveis. Além disso, o cálculo integral não é parte do conteúdo do ensino médio. Por esses motivos, ela foi substituída pela operação de somatório; essa substituição é válida pois mantém o princípio de funcionamento do erro integral. A nova função é mostrada abaixo no item 2.2.

$$I(t) = k_i \sum_{n=0}^t e(n) \quad (2.2)$$

A Figura 34 abaixo mostra uma curva de erro da mesma série de dados anterior – em azul – e a correção feita pela ação integral – em laranja – com um ganho $k_i = 0,25$.

Figura 34. Ação do controlador integrativo



Fonte: Elaborada pelo autor

Já o controlador derivado, remete a operação do cálculo chamada derivada. Ao observar a taxa de variação do erro num determinado instante, o controlador derivado tenta prever qual será a alteração no valor do erro em ciclos imediatamente futuros, dessa forma aplicando a correção antes mesmo de erro acontecer, evitando a desestabilização. O controlador recebe um fator multiplicativo chamado k_d . A equação geral do erro derivado $D(t)$ é mostrada no item 2.3.

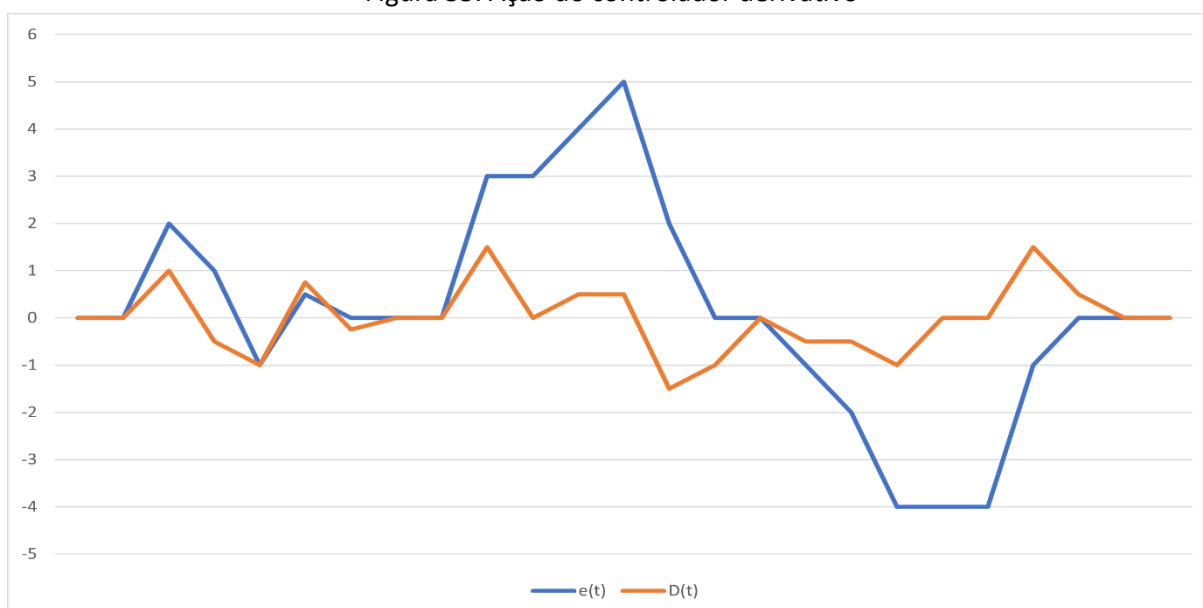
$$D(t) = k_d \cdot e(t)dt \quad (2.3)$$

Assim, como a operação integral, a derivada também não é vista por alunos do ensino médio e requer o conhecimento da função para seu cálculo. Por esses motivos é feita uma aproximação; ela calcula a variação entre dois pontos próximos da curva – nos momentos t e $t - \Delta t$, onde Δt assume um valor relativamente pequeno. Mantendo assim o princípio de funcionamento do controle derivado, isso é, a sensibilidade a variações do sinal de entrada. A nova expressão é mostrada no item 2.4.

$$D(t) = k_d \cdot [e(t) - e(t - \Delta t)] \quad (2.4)$$

A Figura 35 mostra uma curva de erro ainda da mesma série de dados – em azul – e a correção feita pela ação derivada – em laranja – com um ganho $k_d = 0,5$.

Figura 35. Ação do controlador derivativo



Fonte: Elaborada pelo autor

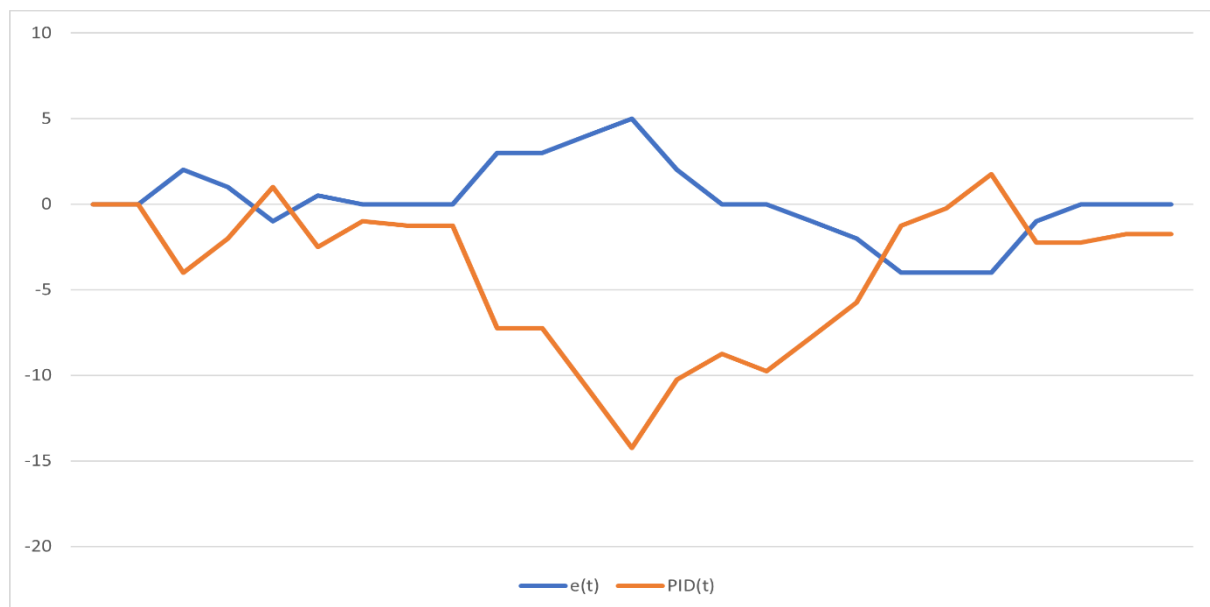
O controlador PID é o resultado do efeito dos três processos mostrados acima. A equação do controlador PID é dada por um valor de referência (S_0), ou saída padrão, ela representa a saída esperada quando a variável controlada está em perfeito equilíbrio, com desvio nulo em relação ao *set point*; e o resultado dos três módulos de controle. É importante lembrar que a ação do controle PID terá que ser negativa, dessa forma ela pode se opor ao erro; caso contrário, e o resultado seja somado à saída, o erro na realidade aumentará.

No item 2.5 é mostrada a equação final para o controle PID.

$$PID(t) = S_0 - [P(t) + I(t) + D(t)] \quad (2.5)$$

Na Figura 36 é mostrada a ação de controle PID, com os multiplicadores de cada controlador: $k_p = 1$, $k_i = 0,5$ e $k_d = 0,5$, e saída padrão $S_0 = 0$. A série de dados usada é a mesma dos três exemplos anteriores.

Figura 36. Ação do controlador PID



Fonte: Elaborada pelo autor

Existem diversas técnicas pra sintonização do multiplicadores do controle PID, mas como nesse projeto o processo não era de alta complexidade, a sintonização foi feita com testes pelo método de tentativa e erro.

3.5. Código base e bibliotecas

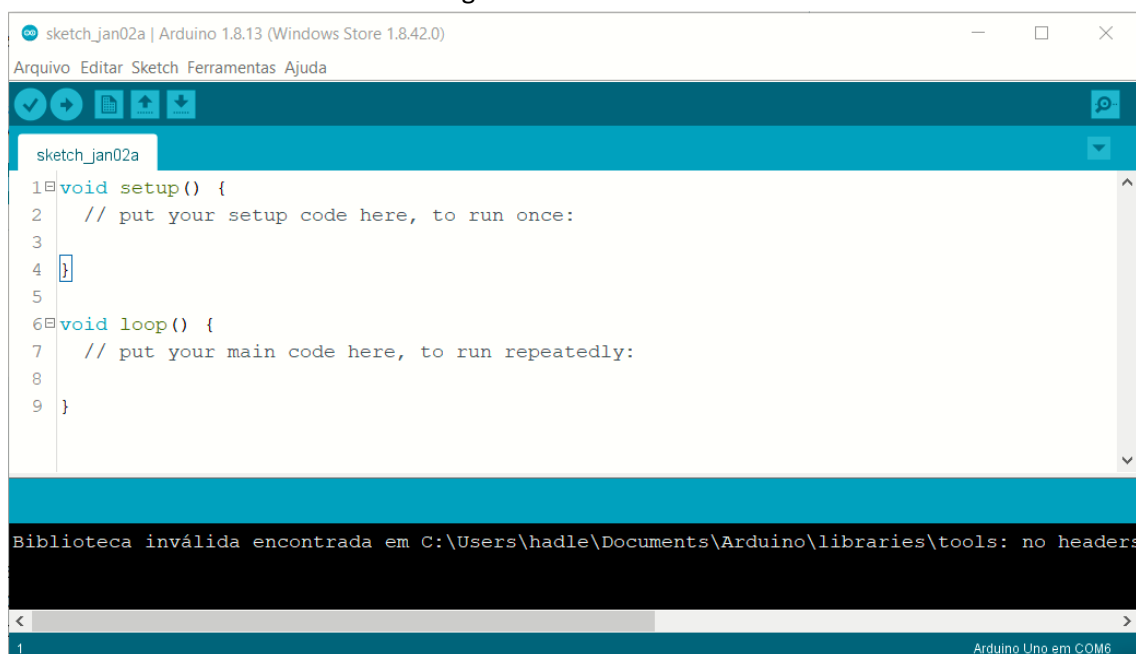
Após o desenvolvimento da estrutura mecânica, estudo de movimento e escolha do algoritmo, foi desenvolvido um código base em linguagem C++ pela interface de programação Arduino IDE. O objetivo desse código foi implementar o algoritmo de controle escolhido, além de realizar todas as configurações bases do controlador, como comunicação com sensores e atuador.

O código desenvolvido oferece ao usuário a possibilidade de configuração de todos os parâmetros do módulo, observação das variáveis em tempo real por meio do plotador de curvas da interface e interação com o processo, permitindo a injeção de distúrbios aleatórios para observar a resposta do controle.

A Arduino IDE pode ser baixada pelo site oficial disponível em: <https://www.arduino.cc/en/software>.

A Figura 37 abaixo mostra a interface inicial da Arduino IDE.

Figura 37. Arduino IDE



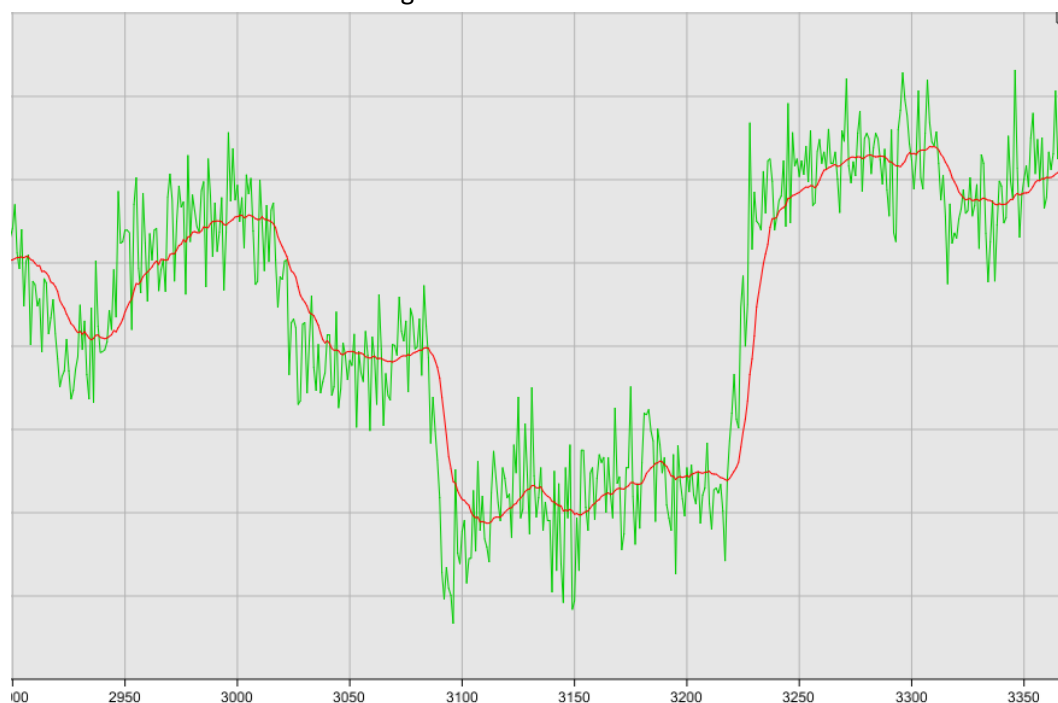
Fonte: Elaborada pelo autor

Durante a elaboração do código foram utilizadas 6 bibliotecas, dentre elas 4 são nativas à interface: “Arduino.h”, “Math.h”, “Servo.h” e “Wire.h”. As outras duas foram instaladas, são elas “SimpleKalmanFilter.h” e “VL53L0X.h”.

A biblioteca “SimpleKalmanFilter.h” foi criada pelo autor Denys Sene, e disponibilizada para acesso, podendo ser acessada pelo gerenciador de bibliotecas nativo da Arduino IDE. Seu algoritmo tem como objetivo a implementação do filtro de Kalman, esse é um algoritmo capaz de reduzir o ruído num conjunto de dados e foi utilizada para a suavização da curva de leitura de dados do sensor de posição.

A ação do filtro de Kalman numa amostra de dados é mostrada na Figura 38.

Figura 38. Filtro de Kalman



Fonte: (SENE, 2020)

Por razões de otimização o algoritmo criado pelo autor Denys Sene foi inserido diretamente no código base desse projeto, deixando explicitamente, na forma de comentários, os créditos ao seu criador. A biblioteca original pode ser encontrada em: (SENE, 2020).

Já a biblioteca “VL53L0X.h”, criada pela empresa Pololu, estabelece a comunicação entre o controlador e o sensor VL53L0X, por meio da interpretação e geração dos pacotes de comunicação. Ela foi baixada diretamente do repositório oficial no GitHub da empresa, disponível em: (VL53L0X-ARDUINO, 2021).

4. RESULTADOS

4.1. Módulo *Ball and Beam* LASE II

Após o desenvolvimento das peças do módulo, elas foram submetidas ao processo de impressão 3D, o resultado é mostrado na Figura 39.

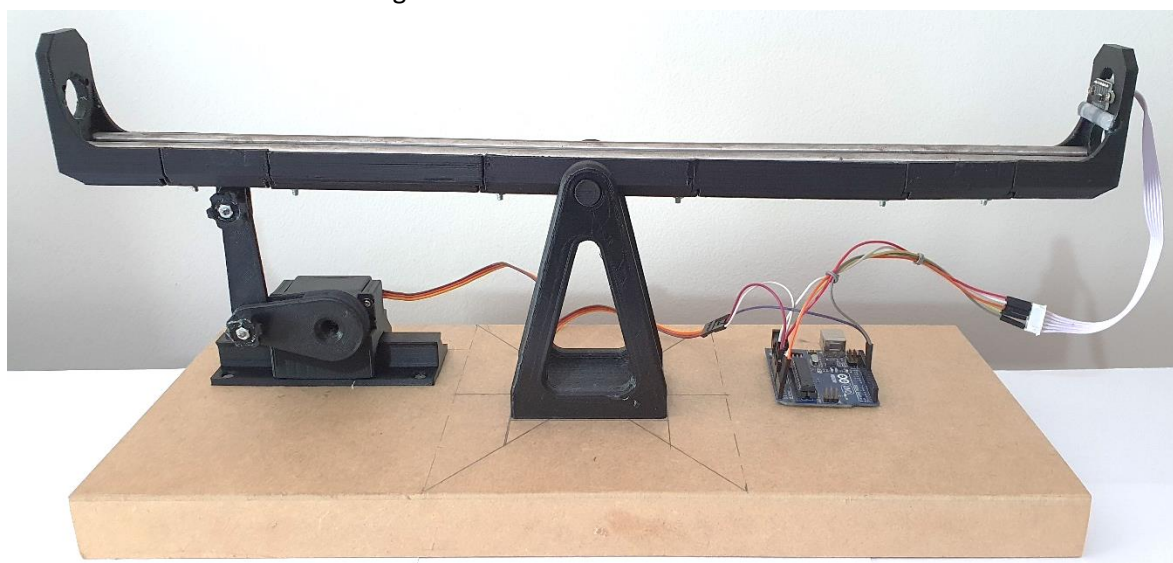
Figura 39. Peças finais impressas



Fonte: Elaborada pelo autor

Com as peças feitas; foi então escolhida uma base de MDF, perfurada conforme instruções no item: 3.2.2.7. Montagem, e realizada a montagem do módulo, o resultado é mostrado na Figura 40.

Figura 40. Módulo *ball and beam* LASE II



Fonte: Elaborada pelo autor

Para a bola do módulo foi utilizada uma bolinha emborrachada de cerca de 5cm de diâmetro. Foram associadas também as barras metálicas nos furos apropriados, como descrito no item: 3.2.2.4. Peças de extremidade.

4.2. Código base do módulo

O código final elaborado é formado por 5 arquivos: `BallAndBeam.ino`, `BallAndBeam.h`, `BallAndBeam.cpp`, `Configuration.h` e `ConfigurationCheck.h`. Cada um deles contém uma parte do código e funcionalidade específica.

O arquivo `BallAndBeam.ino` é o arquivo principal, nele são declaradas as funções de *setup* e *loop* da Arduino IDE. Por esse arquivo o usuário pode chamar as funções internas do módulo para serem executadas.

O arquivo `BallAndBeam.h` recebeu a classe `BallAndBeam`. Ela se encarrega da declaração dos protótipos de funções e das variáveis globais do código, definindo também quais desses elementos são públicos e privados, isso é, quais funções e variáveis podem ser

usadas pelo usuário e quais são de execução exclusiva pelo algoritmo. Nesse arquivo também são importadas as bibliotecas utilizadas no projeto.

No arquivo `BallAndBeam.cpp` são implementadas todas as funções declaradas na classe `BallAndBeam`, dentre elas funções de sistema, como: o algoritmo de controle PID, funções de inicialização e leitura do sensor, filtro de Kalman e escrita do atuador; e outras de interação com o usuário, como a plotagem de curvas das variáveis e possibilidade de geração de distúrbios no sistema.

Já o arquivo `Configuration.h` é composto inteiramente por macros, isso é, diretivas de pré-processamento. Seu objetivo é permitir ao usuário a inserção das condições iniciais do código, como a saída onde está conectada o sensor, e especificações mecânicas, como o comprimento da calha.

No arquivo `ConfigurationCheck.h` existem ainda outras diretivas de pré-processamento, elas checam por possíveis entradas absurdas ou incompatíveis na configurações inseridas pelo usuário, avisando o problema se um erro desse tipo for detectado.

3.3. Arquivos do projeto

Todos os arquivos criados no projeto: modelo 3D das peças, código e este documento; Foram disponibilizadas para acesso no repositório GitHub dedicado ao projeto.

O repositório está disponível no link abaixo, item 3.0:

github.com/HadlerHT/BallAndBeam

(3.0)

5. GUIA DE USO DO MÓDULO *BALL AND BEAM* LASE II

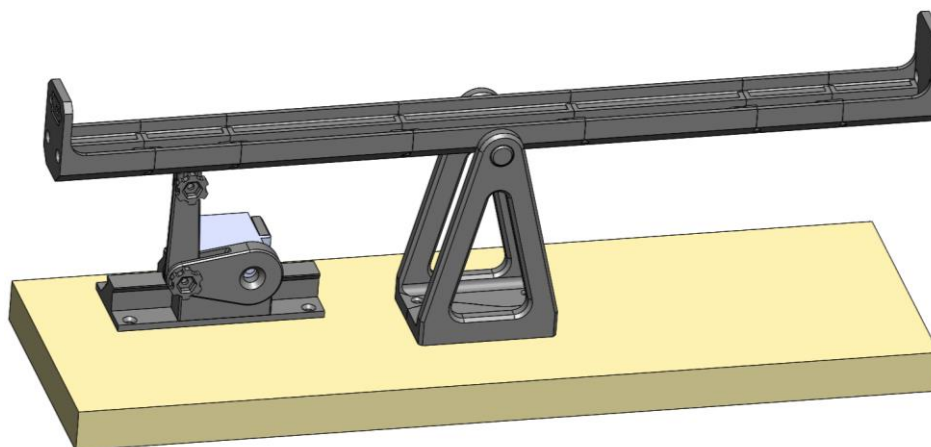
O objetivo desse tópico é a instrução do usuário sobre como utilizar o módulo *ball and beam* LASE II. O tópico enfatiza a possibilidade de escolha do modelo da calha, personalizando comprimento e simetria; montagem do circuito eletroeletrônico, além da explicação de como realizar as configurações do código do módulo.

5.1. Montagem do módulo

As instruções sobre encaixes e funcionalidades das peças foram especificadas no item 3.2. Estrutura mecânica, e nesse tópico são exibidas algumas sugestões de configurações de montagem das peças.

A primeira delas é o modo de montagem padrão. Utilizando todas as peças de extensão a calha atinge cerca de 50cm de comprimento, deixando bastante espaço para oscilações da bola. O modelo é mostrado abaixo na Figura 41.

Figura 41. Calha padrão

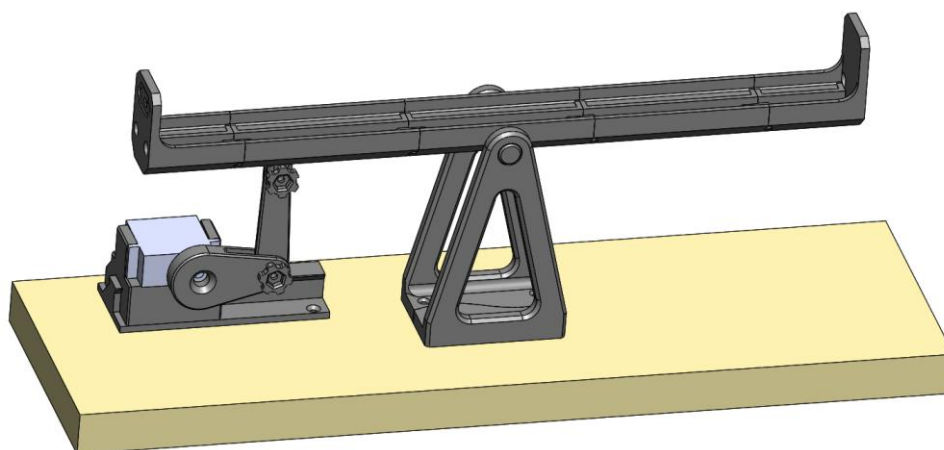


Fonte: Elaborada pelo autor

A utilização das barras metálicas na calha não é obrigatória, mas certamente cria um ambiente mais próximo ao ideal, com mínimo atrito e bordas de contato com a bola bastante lisas; por essas razões seu uso é encorajado.

Se utilizadas apenas as peças de extensão longas é criado o modelo: Calha curta. Com uma extensão de cerca de 40cm, existe menos espaço de oscilação da bola. Ele é mostrado na Figura 42.

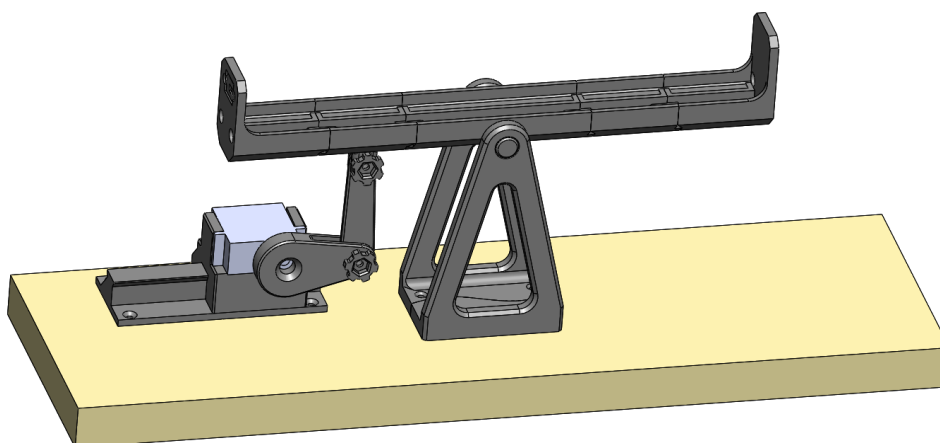
Figura 42. Calha curta



Fonte: Elaborada pelo autor

Com a utilização de apenas duas extensões de 5cm, é montado o modelo: Calha extra-curta. Com apenas 30cm, o controle da bola com tão pouco espaço de oscilação exige uma sintonização precisa do PID. Ela é mostrada na Figura 43.

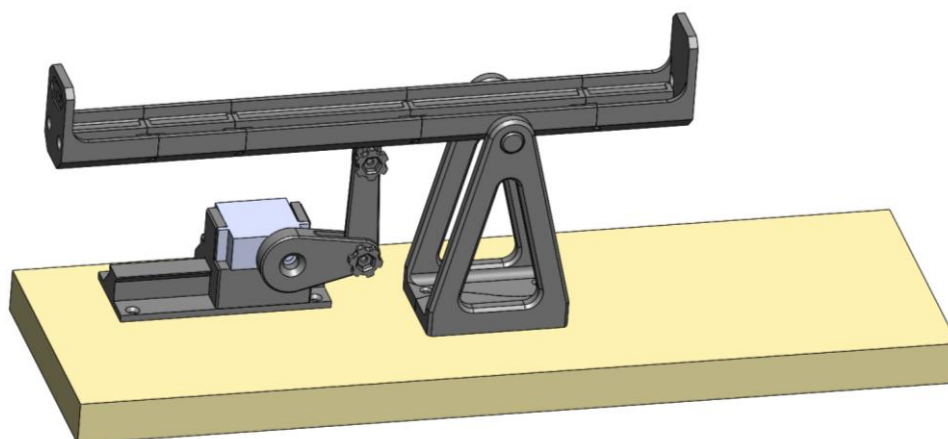
Figura 43. Calha extra-curta



Fonte: Elaborada pelo autor

Com o conceito de peças rearranjáveis, o usuário fica livre para montar a calha da forma que melhor o servir, podendo até mesmo extrapolar com modelos assimétricos como o mostrado na Figura 44.

Figura 44. Calha assimétrica



Fonte: Elaborada pelo autor

5.2. Conexões do *hardware*

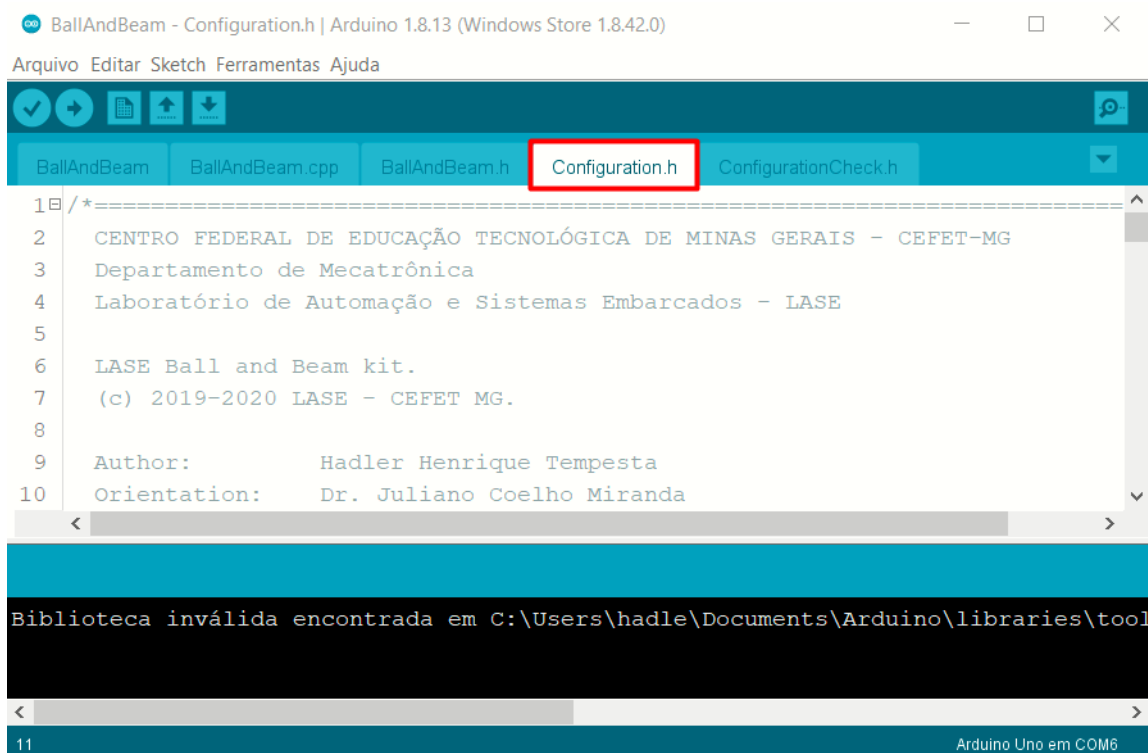
As conexões no módulo são relativamente simples. Primeiramente deve-se realizar a conexão da alimentação em todos os componentes. Então são conectados os dois pinos de comunicação do sensor: SCL e SDA, aos pinos correspondentes do Arduino, eles são, respectivamente: A5 e A4; pino XSHUT pode ser conectado a qualquer saída digital do embarcado. Por último o servo motor tem seu pino de controle (PWM) conectado a uma das saídas do controlador com suporte para essa função, sendo elas os pinos digitais 9 e 10.

Um modelo do circuito é mostrado no Anexo A.

5.3. Configurações iniciais

Como citado no item: 4.2. Código base do módulo, as configurações iniciais são feitas a partir do arquivo: "Configuration.h". Para acessá-lo, abra o arquivo BallAndBeam.ino utilizando a interface Arduino IDE, e localize na parte superior a aba: "Configuration.h". Como mostrado na Figura 45.

Figura 45. Arquivo de configurações iniciais



Fonte: Elaborada pelo autor

Nesse arquivo serão encontradas seções de configuração dos parâmetros do módulo.

5.3.1. Configurações do PID (*PID Settings*)

A primeira seção é referente ao controle PID. Abaixo é descrito a função de cada parâmetro presente:

- **PID_SETPOINT**: Define o valor do *set point*, em mm, para o controle PID;
- **PID_KP**: Valor da constante multiplicativa do controlador proporcional;
- **PID_KI**: Valor da constante multiplicativa do controlador integrativo;
- **PID_KD**: Valor da constante multiplicativa do controlador derivativo;
- **PID_INTEGRATIVE_REFRESH_MS**: Define o intervalo de tempo, em ms, entre cada coleta de dados para ser utilizada na função soma;
- **PID_DERIVATIVE_REFRESH_MS**: Define o intervalo de tempo, em ms, entre as comparações da variação da variável controlada;
- **ERROR_ADMISSION**: Define um coeficiente de tolerância de erro, isso é, o intervalo de dados em torno do *set point* em que o processo é considerado em equilíbrio.

5.3.2. Hardware

Nessa seção são especificadas as conexões feitas no circuito físico e parâmetros do sensor. Primeiramente relacionadas ao atuador.

- **MOTOR_CONTROL_PIN:** Especifica o pino digital do Arduino em que o servo motor foi conectado.

No seguimento, é oferecido ao usuário a possibilidade de escolha de um sensor dentre o VL53L0X, o HC-SR04 e um analógico genérico. Para selecionar um deles, basta comentar – adicionar duas barras “//” a extrema esquerda da linha – no outros dois. No exemplo da Figura 46 o sensor VL53L0X está selecionado.

Figura 46. Escolha do sensor

```
//=====
//=====  HARDWARE  =====
//=====
#define MOTOR_CONTROL_PIN 9

#define SENSOR_VL53L0X
//#define SENSOR_HSR04
//#define SENSOR_GENERIC_ANALOG
```

Fonte: Elaborada pelo autor

Após a escolha é necessária a configuração dos parâmetros associados ao sensor selecionado, no caso do VL53L0X existem três:

- **SENSOR_RESET_PIN:** Especifica o pino digital do Arduino em que foi conectado o pino XSHUT do sensor;
- **ATTEMPTS_TO_BOOT_SENSOR:** Número de tentativas de inicialização do sensor antes de emitir um alerta de erro crítico;
- **SENSOR_TIMEOUT_MS:** Tempo de espera, em ms, pela resposta do sensor antes de emitir um sinal de perda de comunicação.

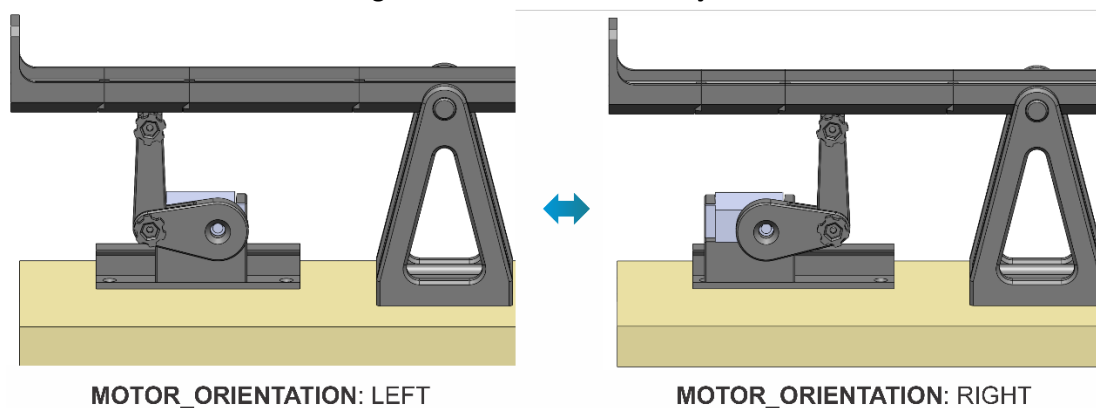
5.3.3. Parâmetros mecânicos (*Mechanical*)

Nessa seção são configuradas as características mecânicas da montagem do módulo.

Sendo elas:

- **BEAM_RADIUS_MM**: Distância, em mm, entre o ponto de rotação da calha e o ponto onde foi conectado o braço articulado. Parâmetro “raio da calha”, especificado no item 3.3. Estudo de movimento da calha;
- **BEAM_LENGTH_MM**: Comprimento total da calha;
- **MOTOR_RADIUS_MM**: Comprimento, em mm, da peça do braço articulado conectada diretamente ao eixo do motor. Parâmetro “raio do motor”, especificado no item 3.3. Estudo de movimento da calha. Com as peças mecânicas desenvolvidas nesse projeto, esse valor é de 40mm;
- **BEAM_HYSTERESIS_DEG**: Histerese, em graus ($^{\circ}$), de oscilação da calha em torno da origem, isso é, quantos graus a calha pode oscilar num sentido e no outro. Parâmetro associado ao $Dom(\alpha)$, tratado no item 3.3. Estudo de movimento da calha;
- **MOTOR_OFFSET_DEG**: Ângulo base, em graus, que deve ser escrito no motor para que a calha tenha inclinação nula. Parâmetro θ tratado no item 3.3. Estudo de movimento da calha;
- **MOTOR_ORIENTATION**: Sentido que a peça associada ao eixo do motor aponta quando a calha está paralela à base e o módulo é visto de frente. Recebe os valores: LEFT ou RIGHT. A Figura 47 exemplifica cada caso.

Figura 47. Parâmetro orientação do motor



Fonte: Elaborada pelo autor

5.3.4. Firmware

Essa seção tem objetivo configurar parâmetros da execução do código, como a interface Arduino IDE já se encarrega de grande parte dessas configurações, apenas um item é mostrado:

- **USART_BAUDRATE:** Taxa de transferência de dados, em bps, do terminal USART do Arduino, ele estabelecerá comunicação entre o computador e o Arduino durante a execução do processo.

5.3.5. Filtro de Kalman

Essa seção oferece acesso a configuração inicial dos parâmetros do filtro de Kalman, os valores foram sintonizados para uso com o processo e não precisam ser alterados. Caso o usuário opte por outros valores os parâmetros disponíveis são:

- **KALMAN_ESTIMATED_ERROR:** Índice de incerteza da estimativa de erro;
- **KALMAN_MEASUREMENT_ERROR:** Índice de incerteza na medida coletada;
- **KALMAN_PROCESS_NOISE:** Índice de ruído do processo.

5.4. Funções do código

O código é dividido entre funções privadas e públicas. As funções privadas executam algoritmos e ações internas ao processo e não podem ser acessadas pelo usuário, elas são chamadas por outras funções ou condições pelo próprio programa. Já as funções públicas podem ser usadas pelo usuário, recebendo parâmetros e/ou retornando valores, as funções públicas presentes no código são descritas abaixo

- **initialize**
 - Protótipo: void initialize (void)
 - Funcionalidade: A função Initialize é chamada uma vez no início do código e realiza a configuração inicial de todo o módulo, como inicialização do sensor, terminal de comunicação, e configuração das saídas e entradas.

- **readSensor**

- Protótipo: float readSensor (bool useKalman)
- Funcionalidade: A função readSensor realiza uma leitura no sensor escolhido, retornando o valor lido; a unidade retornada depende do sensor, no caso do VL53L0X a resposta é em mm. O único argumento é do tipo booleano, ou binário, indicando se deverá ser aplicado ou não o filtro de Kalman na série de valores lidos. O valor “true” ativa o uso do filtro, enquanto o valor “false” o desativa.

- **runPID**

- Protótipo: void runPID (void)
- Funcionalidade: A função run PID executa um ciclo do algoritmo de controle PID, devendo ser chamada constantemente para que o controle seja feito. A função runPID faz por si mesma a aquisição de dados do sensor e escrita da resposta do algoritmo no atuador

- **setPIDParameters**

- Protótipo: void setPIDParameters (float _kp, float _ki, float _kd)
- Funcionalidade: Essa função permite que o usuário altere, durante a execução do código, por meio de alguma condição implementada, os valores das constantes multiplicativas dos módulos do PID. O argumentos da função são, respectivamente, os coeficientes k_p , k_i e k_d .

- **plotPID**

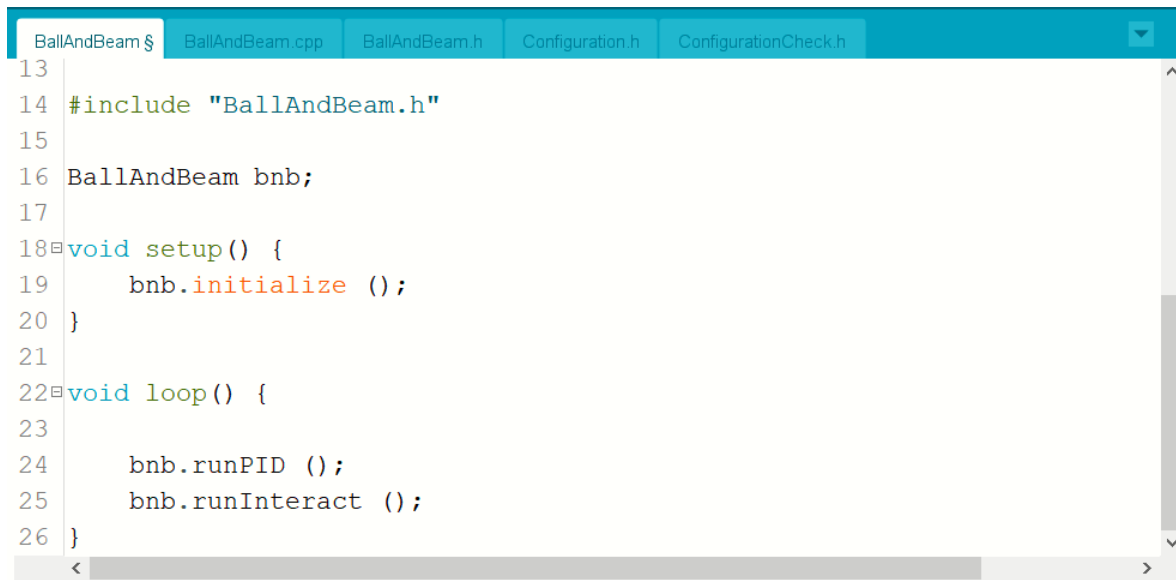
- Protótipo: void plotPID (bool justOutput)
- Funcionalidade: Essa função envia os valores dos controladores do PID (Proporcional, Integral, Derivado e Saída final) pela porta serial do Arduino. Quando lidos pelo *plotter* serial da Arduino IDE (Acessado com Ctrl + Shift + L) é possível observar as curvas de dados do controle. O único argumento do tipo binário oferece a opção de visualização de todos os dados ou apenas da saída final, onde: “true” para ver somente a saída e “false” para ver todos os dados.

- **runInteract**

- Protótipo: `void runInteract (void);`
- Funcionalidade: A função `runInteract`, quando chamada ciclo a ciclo durante a execução do programa garante ao usuário duas interações com o processo por meio do monitor serial da interface Arduino IDE (Acessado pelo atalho Ctrl + Shift + M). Quando enviado um caractere 'R' o sistema gera um ruído aleatório que é aplicado ao sistema no intuito de criar um erro no processo, dessa forma é possível observar a reação do controle a essa intervenção repentina. Quando enviado um caractere 'P', o sistema fica pausado; para retornar basta enviar outro 'P'. Nenhum dos comandos é sensível a capitalização dos caracteres (*case sensitive*).

Para uso das funções no código, basta chamá-las antecipadas do objeto `BallAndBeam` (bnb) como mostrado no exemplo da Figura 48 abaixo; nele são usadas as funções “initialize”, “runPID” e “runInteract”.

Figura 48. Exemplo de uso das funções



```
13
14 #include "BallAndBeam.h"
15
16 BallAndBeam bnb;
17
18 void setup() {
19     bnb.initialize ();
20 }
21
22 void loop() {
23
24     bnb.runPID ();
25     bnb.runInteract ();
26 }
```

Fonte: Elaborada pelo autor

6. CONCLUSÕES

A teoria do controle é um campo extenso de técnicas que garantem o funcionamento correto de mecanismos em todos os setores da tecnologia, assumindo dessa forma grande importância para os profissionais da área de engenharia e da ciência. Nesse contexto, a proposta desse projeto foi desenvolvido o módulo *ball and beam* LASE II, idealizado como uma ferramenta didática para o ensino de mecanismos de controle a nível técnico profissionalizante.

A estrutura mecânica foi modelada e impressa em 3D, garantindo alta replicabilidade. Na calha, as peças foram feitas com encaixes permitindo que o usuário possa montá-la de diferentes maneiras, consequentemente criando novos processos.

O código elaborado, e executado no Arduino UNO, contém pré-configurações para o funcionamento de sensores, atuador, controle PID e parâmetros técnicos do controlador. De modo que o aluno e docente tenham uma experiência de ensino totalmente dedicada à observação e sintonização das ações de controle, garantindo melhor desenvolvimento do assunto.

O resultado final foi um equipamento versátil, de fácil montagem e uso, ele está preparado para o uso por docentes do ensino técnico da área de controle para o ensino de técnicas clássicas como o controle PID.

7. REFERÊNCIAS

CAMPOS, M. C. M. M. de; TEIXEIRA, H. C. G. **Controles típicos de equipamentos e processos industriais**. 2.ed. São Paulo: Blucher, 2010.

COSTA, de M. F. **Desenvolvimento e controle do sistema “ball and beam”**. Monografia (Bacharelado em Engenharia Elétrica). Universidade Federal de Ouro Preto (UFOP), João Monlevade, 2018. [Orientador: Prof. Víctor Costa da Silva Campos]. Disponível em: <<http://monografias.ufop.br/handle/35400000/1144>>. Acesso em: 29 dez. 2020.

DORMIDO, S. et. al. **The ball and beam system: a case study of virtual and remote lab enhancement with Moodle**. IEEE transactions on automatic control, vol.11, no.4. 2015. Disponível em: <<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7120976>>. Acesso em: 29 dez. 2020.

GARCIA, C. Controle de processos industriais: estratégias convencionais. vol.1. São Paulo: Blucher, 2017.

HAUSER, J.; SATRY, S.; KOKOTOVIÉ, P. **Non linear control via approximate input-output linearization: The ball and beam exemple**. IEEE transactions on automatic control, vol.37, no.3. 1992. Disponível em: <<https://people.eecs.berkeley.edu/~sastry/pubs/OldSastryALL/HauserBallandBeam1992.pdf>>. Acesso em: 29 dez. 2020.

KLUG, M. **Construção e controle de uma plataforma experimental ball and beam**. Universidade Federal de Santa Catarina (UFSC), Joinville, 2018. Disponível em: <https://periodicos.ifsc.edu.br/index.php/rtc/article/view/1376/1194>. Acesso em: 30 dez. 2020.

OGATA, K. **Modern control engineering**. 5th ed. Prentice-Hall, 2010.

SENE, D. **SimpleKalmanFilter**. 2020. Disponível em: <<https://www.arduino.cc/reference/en/libraries/simplekalmanfilter/>>. Acesso em: 02 jan. 2021.

TOWERPRO SG-5010. **Motor SG-5010 Datasheet**. 2020. Disponível em: <<https://datasheetspdf.com/datasheet/SG-5010.html>>. Acesso em: 31 dez. 2020.

VL53L0X. **World’s smallest Time-of-Flight ranging and gesture detection sensor**. 2018. Disponível em: <<https://www.st.com/resource/en/datasheet/vl53l0x.pdf>>. Acesso em: 30 dez. 2020.

VL53L0X-ARDUINO. **GitHub ryantm**. 2021. Disponível: <https://github.com/pololu/vl53l0x-arduino>. Acesso em: 02 jan. 2021.

Fontes de imagens:

ARDUINO UNO REV3. **Arduino Store**. 2020. Disponível em: <store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 30 dez. 2020.

ENDER-3 MAX 3D PRINTER. **CREALITY**. 2020. Disponível em: <<https://www.creality.com/goods-detail/creality-ender-3-max-3d-printer>>. Acesso em: 31 dez. 2020.

FILAMENTO ABS PREMIUM+. **3DFila**. 2020. Disponível em: <<https://3dfila.com.br/produto/filamento-abs-premium>>. Acesso em: 31 dez. 2020.

SENSOR DE DISTÂNCIA VL53L0X. **BYTEFLOP componentes eletrônicos**. 2020. Disponível em: <<https://www.byteflop.com.br/sensor-de-distancia-vl53l0x-purple>>. Acesso em: 30 dez. 2020.

ANEXO A

Esquema de conexão dos componentes do módulo

