ETEC JÚLIO DE MESQUITA

Técnico em Mecatrônica Integrado ao Ensino Médio

Arthur Menon Sposito

Arthur Santana Pinheiro

Felipe dos Santos Sousa

Gabriel Silva Araujo

Henrique Bezerra Costa

Henzel Enrico de Lorena Ridolfi

ALMOXARIFADO INTELIGENTE COM FORMATO 3D

Santo André

2022

Arthur Menon Sposito

Arthur Santana Pinheiro

Felipe dos Santos Sousa

Gabriel Silva Araujo

Henrique Bezerra Costa

Henzel Enrico de Lorena Ridolfi

ALMOXARIFADO INTELIGENTE COM FORMATO 3D

Trabalho de Conclusão de Curso apresentado ao Curso Técnico em Mecatrônica Integrado ao Ensino Médio da Etec Júlio de Mesquita, orientado pelos Professores Edivaldo Nonato Santos Júnior e Janaina Cristina da Silva, como requisito parcial para obtenção do título de Técnico em Mecatrônica.

Santo André

2022

“Acho que esse é o melhor conselho de todos: pense constantemente sobre como você pode fazer as coisas melhor e questione a si mesmo.”

ELON MUSK

RESUMO

O presente trabalho tem por objetivo automatizar o processo de organização de caixas de forma inteligente evitando desperdícios, tendo em vista que existem problemas primeiro com a organização destes almoxarifados, posteriormente os acidentes que acontecem como os trabalhadores daquele ambiente, que tanto podem ser por acidentes no manuseio de objetos no local, quanto em decorrência de doença ocupacional. Foi utilizado principalmente sites para a pesquisa pela forma de acesso remota, mas muito dos materiais originais que foram usados eram jornais, revistas e livros impressos que não estão mais sob circulação ou não foram encontrados de forma impressa. Foi concluído com êxito a compra e a montagem de todo o protótipo, adequado a isso uma programação complexa foi colocada para que todo o mecanismo operasse de forma correta sem trancos. As maiores dificuldades foram adequar a programação ao sistema mecânico, pois os erros são difíceis de serem achados sempre tendo que haver uma revisão para saber se o problema não é mecânico, eletrônico ou se está na forma como foi feito as linhas de código. A substituição das estantes de retirada e entrega futuramente podem ser substituídas por esteiras, para facilitar o transporte e agilidade no processo em que ela seja necessária.

Palavras-chave: Segurança. Doença Ocupacional. Automatização Inteligente.

ABSTRACT

This work objective of automating the process of organization of boxes, considering that there are problems first with the organization of these warehouses, later the accidents that happen with the workers of that environment, which can be due to accidents in the handling of objects in the place, or because of occupational disease. Websites were used for research by way of remote access, but much of the original materials that were used were newspapers, magazines and printed books that are no longer in circulation or could not be found in print. The purchase and assembly of the entire prototype was successfully completed, suitable for this a complex programming was placed so that the entire mechanism operated correctly without jerks. The biggest difficulties were adapting the programming to the mechanical system, because errors are difficult to find, always having to be reviewed to see if the problem is not mechanical, electronic or if it is in the way the lines of code were made. The replacement of pick-up and delivery racks in the future can be replaced by mats, to facilitate transport and speed up the process when it is needed.

Keywords: Security. Occupational Disease. Smart Automation.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O manuseio de cargas superdimensionadas sem equipamentos e técnicas adequadas para o corpo humano traz diversos malefícios à saúde do trabalhador como, por exemplo, problemas posturais, musculares e neurais. Por conta disso, em 1980 foi desenvolvido nos Estados Unidos da América o *National Institute for Occupational Safety and Health* (NIOSH), que tem a finalidade de criar regulamentações trabalhistas relacionadas a quantidade de peso que um trabalhador consegue manusear sem afetar a estrutura corporal dados os limites angulares e as dimensões do operador. Quando descumprido por repetidas vezes, o corpo do operário se desgasta, ocasionando em lesões às quais recaem financeiramente na empresa por meio de indenizações e o descarte do funcionário. Sabendo que a concentração de problemas relacionados ao manuseio de cargas está no setor almoxarifado, enunciado pelo *"Brazilian journal of Development",* a automação dessa sessão é uma solução. Nesse contexto surge o almoxarifado automático que através da implementação de mecanismos mecatrônicos, intenta a automatização do estoque, dispensando qualquer interação humana no processo de manipulação de carga, dessa forma, evitando possíveis acidentes e doenças ocupacionais crônicas.

1.1. Descrição do Contexto

No setor almoxarifado o transporte de carga auxiliado por empilhadeira demanda muito espaço, as empilhadeiras de porte normal podem operar em corredores de 1,8 metros de largura, enquanto outros modelos esse número pode chegar a até 4,3 metros. Esse espaço de manuseio acaba por limitar o espaço de estoque que, por sua vez, poderia acomodar um número maior de produtos.

1.2. Definição do Problema

Observam-se os grandes riscos observados pelos materiais lidos a respeito de acidentes que acontecem dentro deste recinto e os mapas de riscos analisados pelos integrantes do local, o manuseio e transporte de cargas por mãos humanas,

agravadas por conta do ambiente de operação estreito de um almoxarifado que limita o movimento do operador naquele ambiente levando-o a se machucar. Uma vez lesionado, diversos processos podem acontecer, tais como o afastamento do operário, causando uma queda na produtividade e custos extras com novas contratações e, em alguns casos, o pagamento de indenização por parte da empresa. Quando não realizado pelo homem, máquinas de grande porte são utilizadas, diminuindo o espaço útil que poderia ser utilizado para o armazenamento de mais produtos, além de não mitigar possíveis acidentes.

1.3. Objetivo Geral

Este trabalho tem por objetivo geral a prototipação de um processo de armazenamento, visando o aumento da organização e do espaço útil do estoque, dispensando assim qualquer interação humana no processo de manuseio do produto.

1.4. Objetivo Específico

Para a realização da prototipação do processo de armazenamento foram utilizados os seguintes meios:

- Programação em linguagem “C++” para Arduino;
- Utilização de componentes eletroeletrônicos leves;
- Utilização de componentes mecânicos leves;
- Trabalhos de marcenaria;
- Confecção de peças em impressora de 3 dimensões;
- Confecção de projetos no software Inventor;

2. DESENVOLVIMENTO

Com base em problemas decorrentes pelo manuseio de cargas, como caixas, ficou evidenciado que o armazém é a parte mais perigosa. Em espaço confinado onde se armazena produtos inflamáveis, pesados ou voláteis, existem perigos relacionados

a: Incêndios, batidas, quedas e cortes, se descontar o efeito do erro humano. Ao ser adicionado o fator humano nas atividades, ocorre outros acidentes maiores que podem envolver os fatores psicológicos ou emocionais do trabalhador. O cansaço é tido como um dos principais causadores dos acidentes por empilhadeiras, pois esse estado causa letargia e sono no operador, como o que ocorreu abaixo:

“...Um jovem trabalhador, recém-admitido, foi a reorganizar caixas armazenadas em prateleiras em um almoxarifado. O trabalhador estava usando um tipo de empilhadeira preparada para elevar pessoas (tipo plataforma) a uma altura de 3,9 metros. Ele não estava usando nenhum equipamento de proteção contra queda. Para acessar as prateleiras, o trabalhador provavelmente saiu da plataforma do operador para uma plataforma de trabalho que foi montada com um pallet nos garfos da empilhadeira. Ele caiu da plataforma de trabalho para o piso de concreto, uma lesão grave....” (MENDES, 2013)

Erros como esse são comuns dentro do ambiente de trabalho no dia a dia de uma empresa, muitas vezes pressionados pelo excesso de demanda, um mau treinamento dentro da empresa que é obrigatório e deve ser feito de forma seguindo os padrões estipulados, porém estes movimentos e ações são feitos de maneira errada pelos trabalhadores, existindo assim um grande problema, a doença ocupacional, que pode causar diversos problemas tanto para o trabalhador como para a empresa, como explicado abaixo pela Fetropar:

“... Movimentação de pesos inadequada pode causar danos irreparáveis aos trabalhadores. As atividades laborais que exigem esforços por conta da necessidade de movimentar cargas pesadas podem afetar a saúde do trabalhador quando realizadas de maneira incorreta. A movimentação de pesos pode afetar principalmente a coluna do empregado. Para que ela seja protegida, o trabalhador deve seguir algumas medidas de segurança e maneiras corretas de transporte de cargas. A má condução de pesos pode causar lesões musculoesqueléticas, que afetam músculos, articulações, tendões, ligamentos, nervos e ossos, e doenças localizadas no aparelho circulatório."

Levantamento

Ao levantar a carga, o corpo deve estar o mais próximo possível do material, mantendo a coluna sempre reta e na vertical. O levantamento não pode ser feito de forma brusca, com puxões para fazer força e movimentar a carga...” (Fetropar, 2022).

Dentro das NR (Normas Reguladoras) estão especificados todos os cuidados a se ter:

“...17.3.1 A organização deve realizar a avaliação ergonômica preliminar das situações de trabalho que, em decorrência da natureza e conteúdo das atividades requeridas, demandam adaptação às características psicofisiológicas dos trabalhadores, a fim de subsidiar a implementação das medidas de prevenção e adequações necessárias previstas nesta NR...” (Portaria MTP, 2021)
Pág. 2

“...17.4.2 Nas atividades que exijam sobrecarga muscular estática ou dinâmica do tronco, do pescoço, da cabeça, dos membros superiores e dos membros inferiores, devem ser adotadas medidas técnicas de engenharia, organizacionais e/ou administrativas, com o objetivo de eliminar ou reduzir essas sobrecargas, a partir da avaliação ergonômica preliminar ou da AET...”
Pág. 3

“...17.5.1 Não deverá ser exigido nem admitido o transporte manual de cargas por um trabalhador cujo peso seja suscetível de comprometer sua saúde ou sua segurança. 17.5.1.1 A carga suportada deve ser reduzida quando se tratar de trabalhadora mulher e de trabalhador menor nas atividades permitidas por lei...”
Pág. 5

“...Carregar um saco de cimento de 25 ou 50 kg por dia pode ser um problema. Porém, que tal carregar 50 sacos desses no mesmo dia, por vários dias e anos? Muito pior. Nesse exemplo, mesmo pequenas cargas com 1, 2 ou 5 kg, comuns em caixas de supermercado, tornam-se um problema devida a exaustiva repetição. A repetição é relacionada ao peso como um agravante. Assim, mesmo pesos leves podem causar grandes problemas em curto ou longo prazo....” (BARTH, 2020).

Dentro disso veio a solução, que logicamente seria retirar o fator humano de todo manuseio dessa carga quando fosse armazenada e se caso precisa-se do contato de uma pessoa fosse minimizado ao máximo, onde todos os ângulos de

liberdades e de força do corpo humano fossem minimamente exigidos. Com isso veio o desenvolvimento e o processo de adequação a todas as normas de segurança para que assim o trabalhador não se acidentasse processo. O conceito de automatização sem de ter a observação humana faz parte da mecatrônica e foi algo a ser inserido através dos inúmeros cálculos e sensores que farão o processo de cruzamento de informações para realizar as tarefas.

2.1. Princípio de Funcionamento

Tem-se por objetivo a retirada completa do ser humano no processo de manuseio de carga. Portanto, desenvolveu-se uma estrutura capaz de manusear o volume automaticamente ao seu local predeterminado dispensando qualquer ação humana no procedimento.

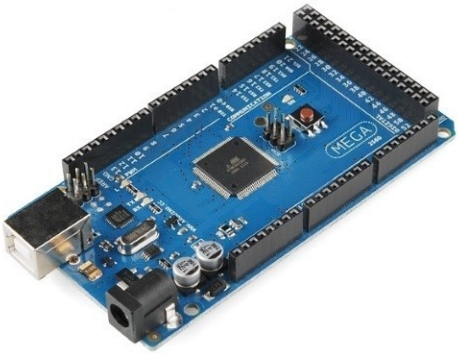
Move-se o carro de transporte através de dois fusos movidos através de motores de passo, um posicionado horizontalmente, para o deslocamento horizontal, e outro posicionado verticalmente, responsável pelos movimentos verticais ascendente e descendente. Quando posicionado no nicho, o carro vertical avança o garfo de manuseio e realiza o procedimento de entrega ou coleta dependendo a instrução do operador. Instruções essas realizadas através de um painel contendo quatro botões que quando pressionados interagem com o painel de visualização que expõe um menu de interação e o acompanhamento do procedimento em tempo real.

2.2. Peças Utilizadas

2.2.1. Arduino Mega

O Arduino Mega é uma placa de prototipagem que atende os usuários com projetos que exigem mais memória e/ou maior número de pinos quando comparados a outras placas do fabricante. A placa Mega é baseada no processador ATmega2560, um microcontrolador de 8 bits. Juntamente de 54 pinos digitais e 16 pinos analógicos. Uma excelente opção para projetos exige numerosos sensores ou atuadores.

Figura 1 - Arduino Mega 2560



Fonte: (Filipe Flop, 2022)

2.2.1.1. Pinos Digitais

Utiliza-se pinos digitais em operações onde a leitura ou acionamento do componente conectado a ele exija a leitura apenas de dois níveis, o alto, correspondendo a 5 V e o baixo de 0 V.

2.2.1.2. Pinos Analógicos

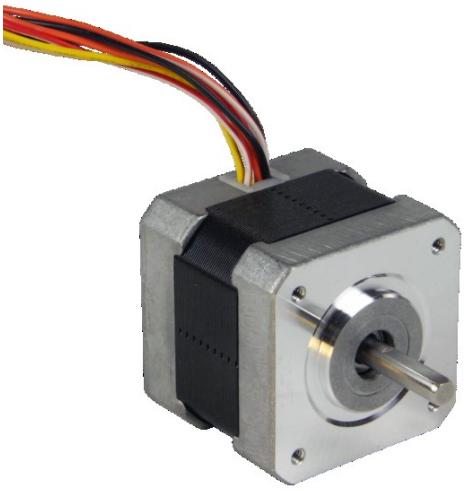
Define-se pinos analógicos como os pinos que possuem a capacidade de ler e enviar dados analógicos, ou seja, a tensão consegue assumir múltiplos valores desde que ela esteja no intervalo de 0 V a 5 V.

2.2.2. Motor Nema 17

Aplica-se os motores de passo em projetos cuja movimentação com precisão é necessária. Uma vez que já que eles são controlados através de sinais digitais que permitem a rotação fracionada do seu eixo, garantindo assim um sofisticado controle de torque, velocidade e posição.

Aproveita-se a robustez combinada com o elevado torque do motor de passo NEMA 17 para a movimentação horizontal e vertical do carro de transporte, além do avanço e retração da garra de coleta.

Figura 2 - Motor Nema 17



Fonte: (Applied Motion, 2022)

O Motor NEMA 17 possui as seguintes especificações conforme Tabela 1:

Tabela 1 - Especificações do Nema 17

Tensão por fase	Corrente por fase	Torque	Ângulo do Passo	Diâmetro do eixo	Comprimento
2.6 V DC	1.7	4 kg.cm	1.8°	Ø5 mm	40 mm

Fonte: Autores (2022)

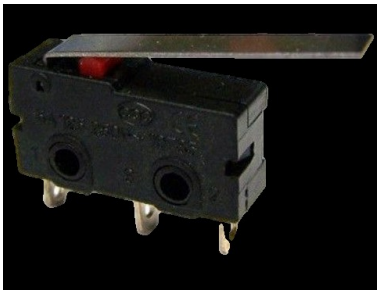
2.2.3. Fuso de Transmissão

O sistema de transmissão por fuso é amplamente utilizado em aplicações que necessitem de uma transmissão de potência de maneira eficiente e altamente precisa. Em razão disso, usufrui-se dela para a realização do movimento horizontal e vertical do projeto através de dois fusos posicionados horizontalmente e verticalmente, respectivamente. Ambos são M8 x 700 mm, número de entradas igual a 4 e avanço igual a 8 mm.

2.2.4.Chave Micro Switch

Popularmente conhecida de chave fim de curso, a chave *micro switch* se trata de um dispositivo eletromecânico, de baixo custo e alta durabilidade, que tem finalidade objetivo indicar quando algo o pressionar.

Figura 3 - Chave Micro switch

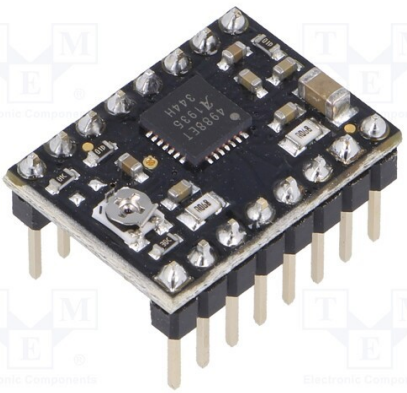


Fonte: (Baú da Eletronica, 2022)

2.2.5. Driver de Motor de Passo A4988

Realiza-se a controle do motor de passo por intermédio de um módulo de controle alcunhado de *driver* de motor de passo cujo modelo escolhido para o projeto é A4988. Por padrão, a cada pulso elétrico enviado a ele, o motor interligado a ele movimenta-se 1,8º, desse modo, a cada 200 pulsos enviados, realiza-se uma revolução.

Figura 4 - Módulo driver do motor de passo A4988



Fonte: (TME, 2022)

2.2.6. Acoplamento Flexível

Une-se os eixos dos motores aos fusos através de um dispositivo mecânico nomeado acoplamento flexível. Um dispositivo mecânico cuja utilidade é unir eixos em sistemas rotativos de transmissão mecânica. Sobretudo para transmissão de torque e rotação, absorção de choques mecânicos e redução de pequenas vibrações e ruídos.

Figura 5 - Acoplamento flexível 8mm

Fonte: (MP MULTIPEÇAS, 2022)

2.2.7. Sensor de Reconhecimento de Cor

Verifica-se a cor de uma superfície por intermédio de um sensor de cor. Mede-se a luz refletida que atinge o sensor, calcula-se e retorna-se a cor.

Atribui-se ao projeto o módulo sensor TCS230, que é composto por 64 fotodiodos. Desses 64 fotodiodos, 16 tem filtros para cor vermelha, 16 para cor verde, 16 para cor azul e 16 não tem filtro. Distribuídos uniformemente sobre o sensor, esses sensores captam a luminosidade, filtrando as cores, e geram na saída um sinal de onda quadrada com as informações sobre a intensidade das cores.

Figura 6 - Sensor de reconhecimento de cor

Fonte: (Eletrônica Faria, 2022)

2.2.8. Barra Retificada

Utiliza-se barras retificadas para garantir o suporte durante a locomoção dos carros de movimento horizontal e vertical por conta da vantagem em relação as barras comuns. O processo de retificação propicia vantagens em relação às barras comuns: maior precisão dimensional, além de terem menor rugosidade, ou seja, o acabamento superficial é mais liso, resultando em menos atrito.

Figura 7 - Barra retificada

Fonte: (Aço Rag, 2022)

2.2.9. Flange

Acopla-se a flange ao fuso de modo a converter o movimento circular do eixo em movimento linear, assim permitindo o deslocamento linear das partes movidas do projeto. Realiza-se a interligação os dispositivos mecânicos de modo que a flange possua o mesmo passo e diâmetro do fuso.

Figura 8 - Fuso retificado com de 8mm com flange

Fonte: (Microwat, 2022)

2.2.10. Rolamento Linear LM8UU

Diminui-se a fricção entre as superfícies movidas através de um rolamento linear permitindo assim movimentos mais livres e controlados. Tal ação é realizada através da utilização de pequenas esferas metálicas internas as quais possuem um sistema de lubrificação interna, mesmo esquema instalado em impressoras 3D, máquinas CNCs, e projetos robóticos em geral.

Figura 9 - Rolamento linear 8mm

Fonte: (Acelera 3D, 2022)

2.2.11. Filamento de Impressora 3D ABS

Faz-se a impressão 3D de algumas das estruturas presentes no projeto. Estruturas essas feitas de ABS, plástico acrilonitrila butadieno estireno, composto muito útil para a produção de peças complexas e de alta resistência e, simultaneamente, possuindo um excelente custo-benefício.

Figura 10 - Filamento de impressora 3D ABS

Fonte: (Filipe Flop, 2022)

2.2.12. Mancal KP08

Deseja-se a fixação do eixo de modo que não haja escorregamento e diminuição do atrito. Para isso, recorre-se ao mancal KP08, um conjunto de peças que embute um rolamento de 8 mm de diâmetro fixado em um suporte aparafusável.

Figura 11 - Mancal KP08

Fonte: (Loja da Robótica, 2022)

2.2.13. Mancal KFL08

Igualmente ao mancal KP08, tira-se proveito do mancal KFL08 para não haver escorregamentos e diminuir o atrito. A única diferença entre eles é a estrutura, sendo esse utilizado para fixação do fuso vertical.

Figura 12 - Mancal KFL08

Fonte: (Maganize Luiza, 2022)

2.3. Impressão 3D

A impressão 3D é um processo de manufatura que atua em diversos segmentos, vindo desde a confecção de peças decorativas até peças móveis e funcionais. Esse processo permite uma flexibilidade enorme em relação à criação de peças totalmente únicas e que seriam, de certa forma, complicadas de serem feitas, por exemplo em equipamentos e usinagem e máquinas de controle numérico (CNC).

Tal flexibilidade foi imprescindível para sua escolha neste projeto, uma vez que a necessidade de se utilizar peças totalmente criadas do zero e dimensionadas para uma função em específico, não seria possível assim a utilização dos equipamentos da oficina escolar além de que, a posse e uso de impressoras 3D por membros do grupo já é feito desde antes do início do projeto, dando ao grupo uma maior capacidade de escolha para a confecção das partes do projeto.

2.3.1. Software de Fatiamento

Uma impressora 3D trabalha com arquivos GCODE que são interpretados de modo a criar camadas de material empilhadas gerando o modelo 3D físico. Os softwares de fatiamento codificam o arquivo a ser impresso e através dos padrões preestabelecidos na configuração no software, sendo esses: paredes, preenchimento, construção de suporte, altura de camada, dentre outras configurações indispensáveis para impressão 3D.

Compreende-se camadas como a parte em que o material será colocado e dará assim a forma da impressão calculadas através do software de fatiamento, o qual possui esse nome justamente por “cortar” o objeto 3D em milhares de camadas no qual será impresso o objeto físico. A altura da camada trabalha para dar a forma do objeto determinando a altura que o material terá durante a impressão sendo ela variável de acordo com a parte da impressão. Camadas maiores possuam um acabamento inferior a camadas menores. Determina-se paredes como a área externa na qual o formato final do objeto será visível, podendo se usar mais de uma camada de paredes (indicado o uso de 3 paredes). Define-se preenchimento como o volume interno e que dará ao objeto a resistência necessária para seu fim específico, quanto maior a porcentagem do preenchimento, mais material se utiliza e mais resistente o objeto se torna.

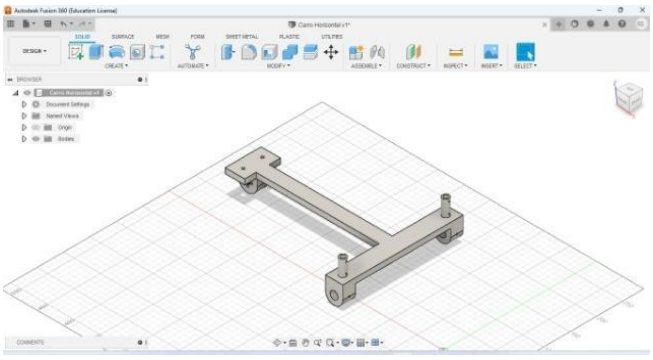
Em casos em que não há apoio para o material de impressão utiliza-se suportes. Assim é possível realizar a impressão sem danificá-la ou alterá-la. Tem-se a base como a parte principal da impressão. Ela é a única que possuirá o preenchimento de 100%, visto que sua boa qualidade é de suma importância para a continuação bem-sucedida da impressão. Chama-se a área útil da impressora de área de impressão. Em um software de fatiamento a parte em se é possível colocar uma impressão, é configurada pelo operador da impressora, que define no software todos os parâmetros da máquina.

Outro ponto essencial na impressão 3D é a temperatura durante a impressão, cada tipo de material (PLA, ABS etc.) possui uma temperatura específica para seu uso, com essa temperatura alterando-se também de fabricante para fabricante, criando assim obstáculos para uma boa impressão, mas que podem ser evitados por testes de temperatura e configurações pré-definidas incluídas nos softwares de fatiamento.

2.3.2. Cortes e Encaixes

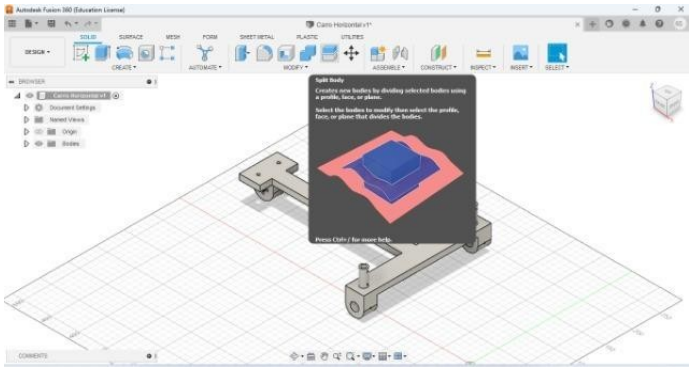
Dadas as grandes dimensões das peças projetadas, fez-se necessário o corte delas e subsequentemente a criação de encaixes através do software de modelagem 3D Autodesk Fusion 360. A
Figura 13, a Figura 14 e a Figura 15 descreve, o procedimento de abertura do 3D e a realização dos corte e os encaixes. Nota-se que é necessário selecionar a opção “Split” e o plano em que será cortado.

Figura 13 - Recorte do carro horizontal para impressão



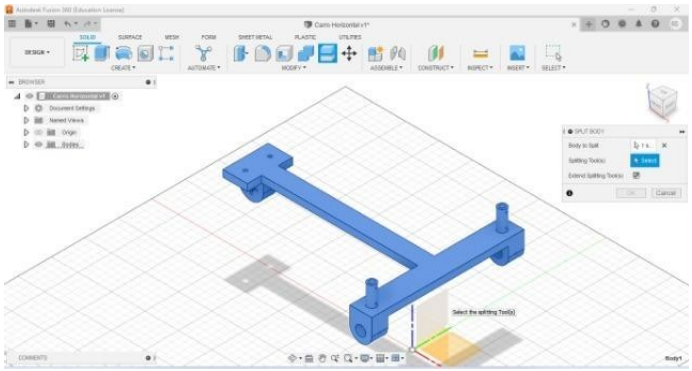
Fonte: Autores (2022)

Figura 14 - Função de corte de peça



Fonte: Autores (2022)

Figura 15 - Seleção de peça

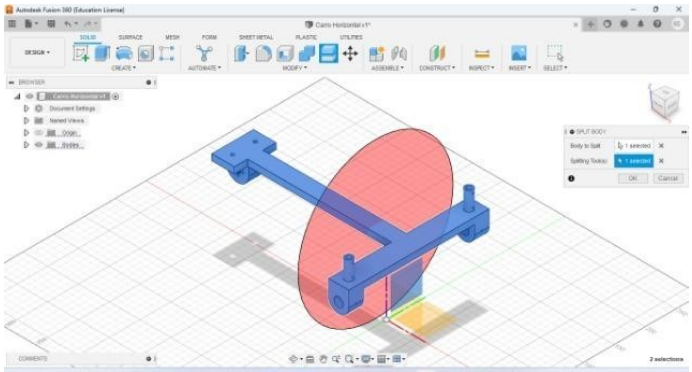


Fonte: Autores (2022)

A coletânea iniciada na

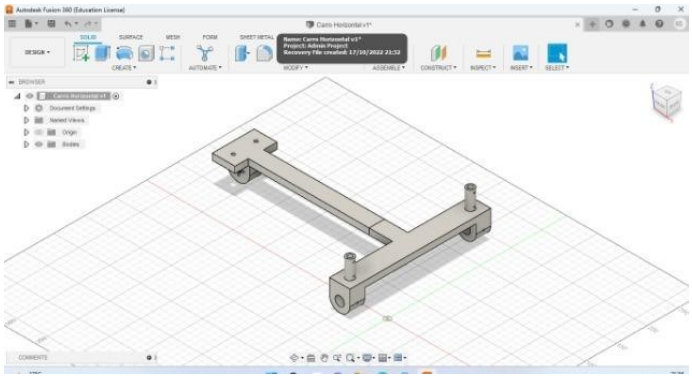
Figura 16 e terminada na Figura 18 mostra o plano selecionado, a área onde será aplicado o corte, e o corte realizada na peça.

Figura 16 - Escolha da área de corte



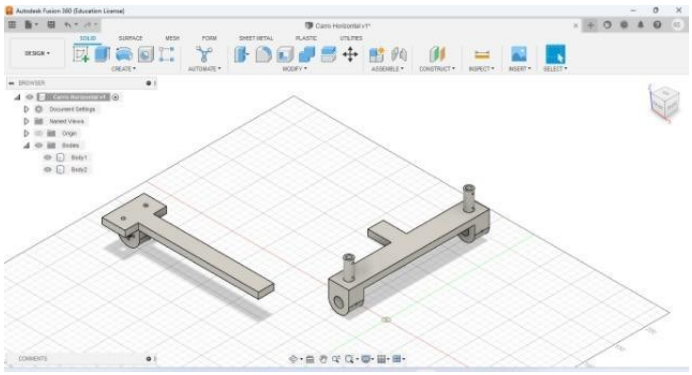
Fonte: Autores (2022)

Figura 17 - Inserção de pino de encaixe



Fonte: Autores (2022)

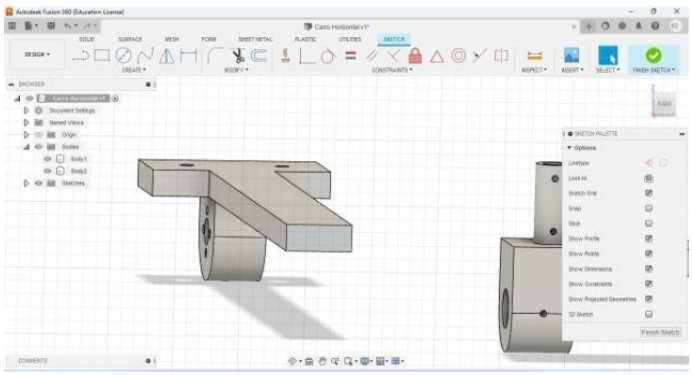
Figura 18 - Separação das seções da peça



Fonte: Autores (2022)

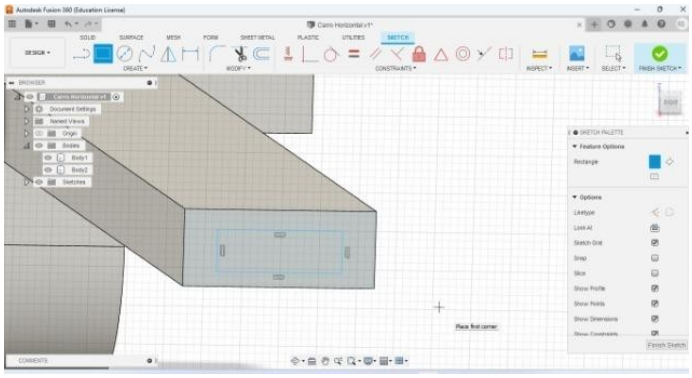
Já as sequências de figuras entre a **Erro! Auto-referência de indicador não válida.** e a Figura 22 mostram o processo de criação do encaixe. Primeiramente “esboço” na área cortada. Feito isso, cria-se um retângulo nesta área que será usado como encaixe, logo após, a função “Extrude” é utilizada para dar o volume externo do encaixe.

Figura 19 - Seleção da peça com pino



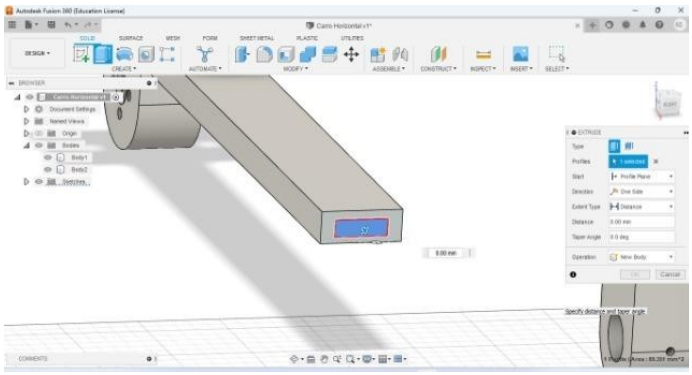
Fonte: Autores (2022)

Figura 20 - Inserção do pino



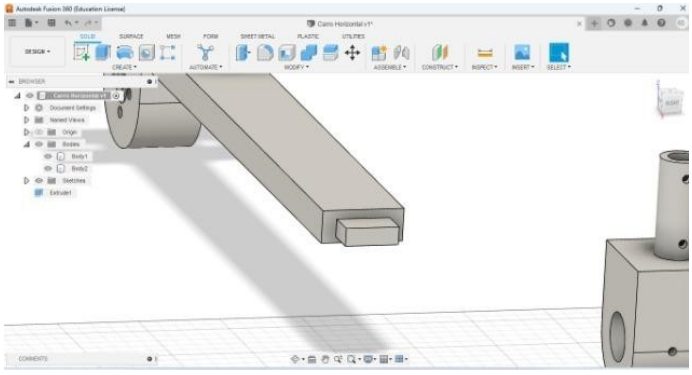
Fonte: Autores (2022)

Figura 21 - Escolha da profundidade do pino



Fonte: Autores (2022)

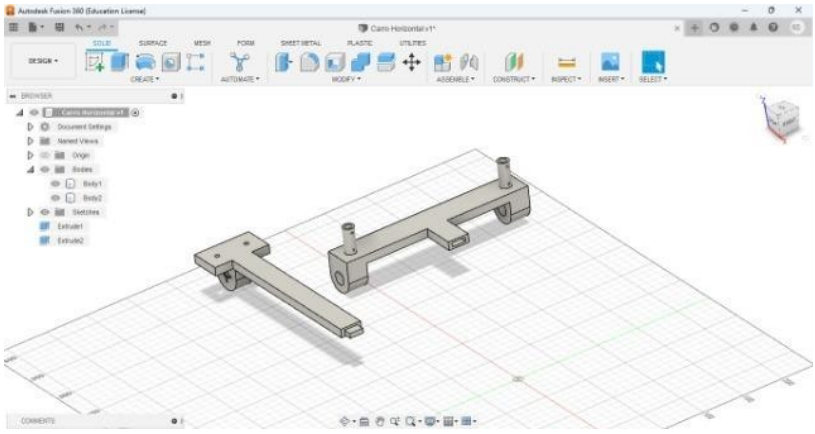
Figura 22 - Resultado



Fonte: Autores (2022)

Por sua vez, a Figura 23 demostra o mesmo procedimento na outra metade da peça, para criar o “vazio” no qual a parte anterior irá se encaixar.

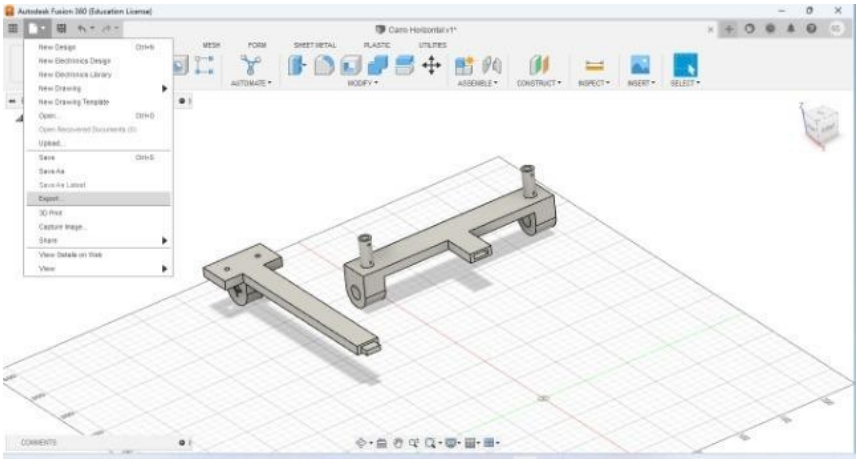
Figura 23 - Rebaixo realizado para o encaixe das peças



Fonte: Autores (2022)

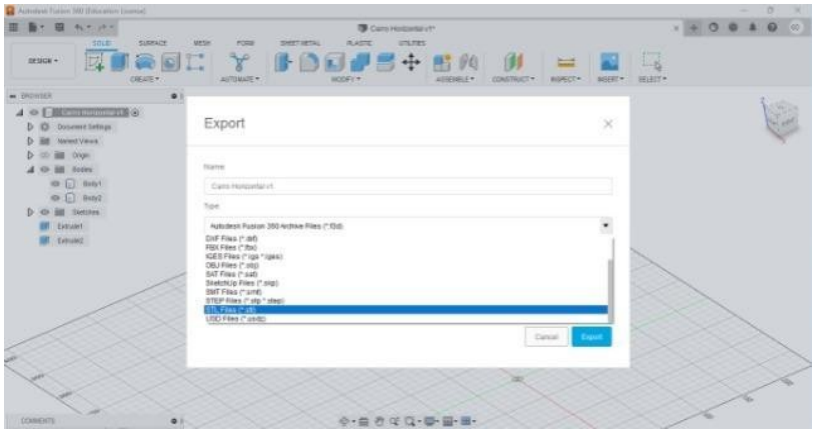
Com os encaixes prontos, a Figura 24, Figura 25 e Figura 26 apresentam o processo necessário para que ambas as partes sejam salvas separadamente e na extensão.stl. Para isso utiliza-se a opção “*Export*” com o objetivo mudar o tipo de extensão no qual o arquivo foi salvo. Para a exportação correta para impressão 3D, seleciona-se “.stl” e em qual pasta será salvo o arquivo, após isso clica-se em "*Export*" no canto direito da tela.

Figura 24 - Separação das peças em arquivos diferentes



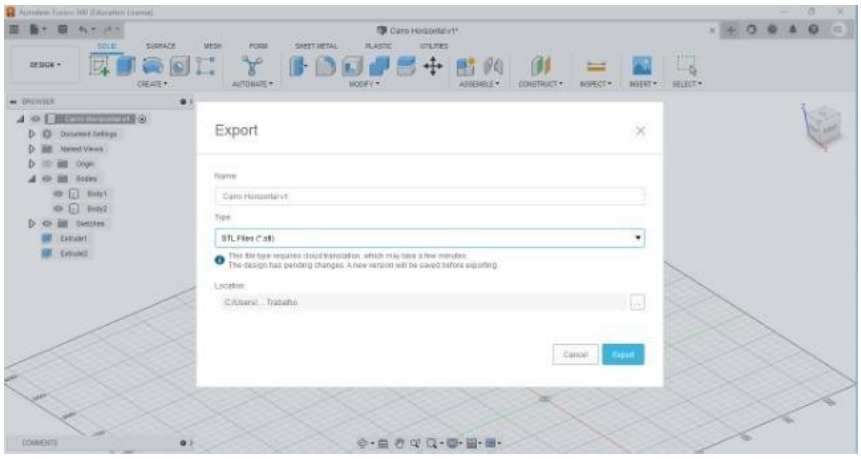
Fonte: Autores (2022)

Figura 25 - Separação dos arquivos



Fonte: Autores (2022)

Figura 26 - Tela de confirmação

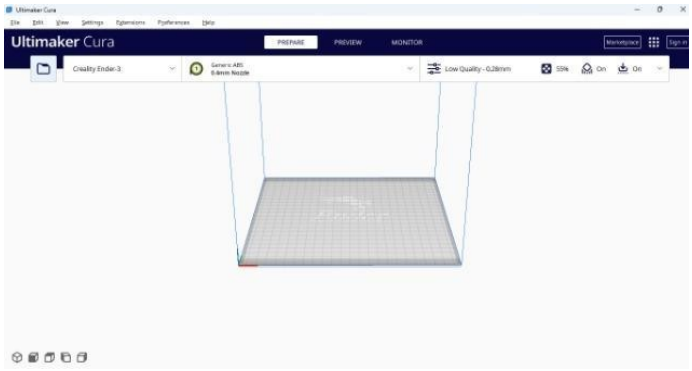


Fonte: Autores (2022)

2.3.3. Fatiamento das Peças

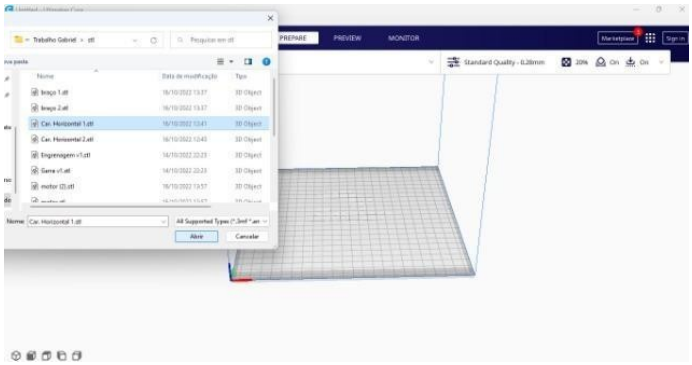
Faz-se necessário a geração de um código GCODE através de um arquivo STL, sendo essa o tipo de arquivo que um software de fatiamento aceita. Apresenta-se da Figura 27 e Figura 28 como adicionar o arquivo a ser utilizado. Clica-se no botão “Importar Arquivo” na margem superior esquerda da tela e após isso seleciona-se qual arquivo será fatiado.

Figura 27 - Abertura do *software Ultimaker Cura*



Fonte: Autores (2022)

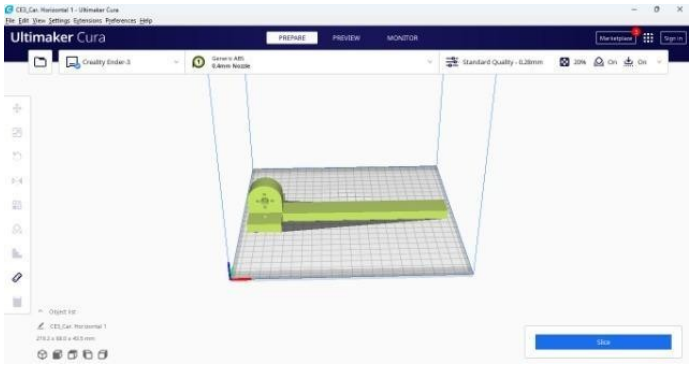
Figura 28 - Seleção do arquivo



Fonte: Autores (2022)

A Figura 29 apresenta o arquivo escolhido, a forma em 3D será gerada na área de trabalho, desse modo, move-se a peça de forma a encaixá-la na parte clara da mesa. As bordas escuras da mesa representam a área da mesa em que impressão não pode ocorrer, visto seu uso para outras funções.

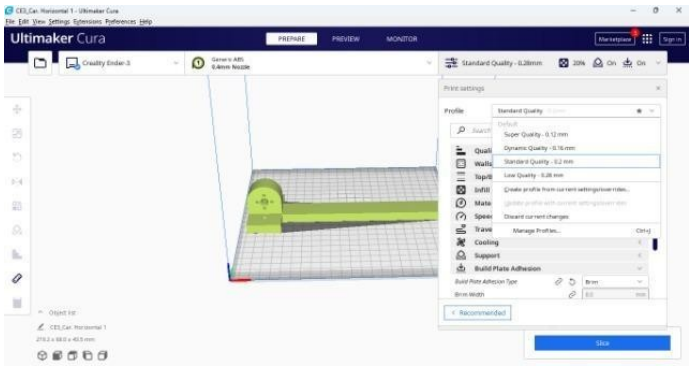
Figura 29 - Inserção da peça na área de trabalho da Impressora 3D



Fonte: Autores (2022)

Já a Figura 30 apresenta as configurações de impressão como temperatura, preenchimento etc. Tais configurações são feitas de acordo com o material utilizado (PLA e ABS geralmente), com cada um tendo sua temperatura específica, tendo ela leves mudanças de acordo com o fabricante, e parâmetros da própria impressora também são colocados nas configurações.

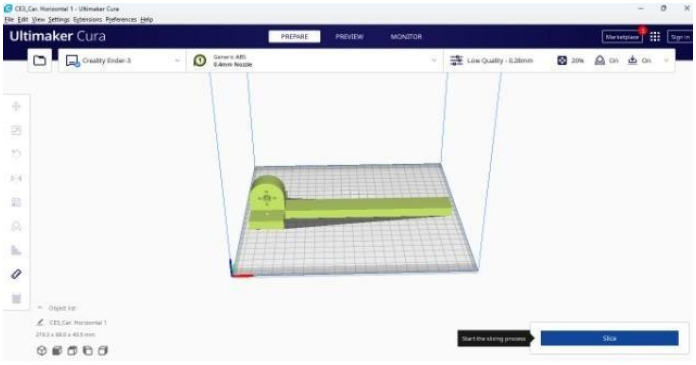
Figura 30 - Configuração da impressora



Fonte: Autores (2022)

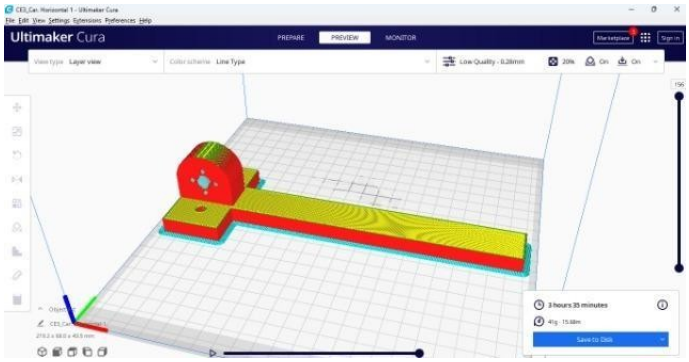
A Figura 31 e Figura 32 nos apresentam a peça pronta para ser fatiada, clicando-se no botão “Slicer” no canto inferior direito da tela, a peça será fatiada.

Figura 31 - Início da exportação do arquivo para a impressora



Fonte: Autores (2022)

Figura 32 - Vista 3D das camadas

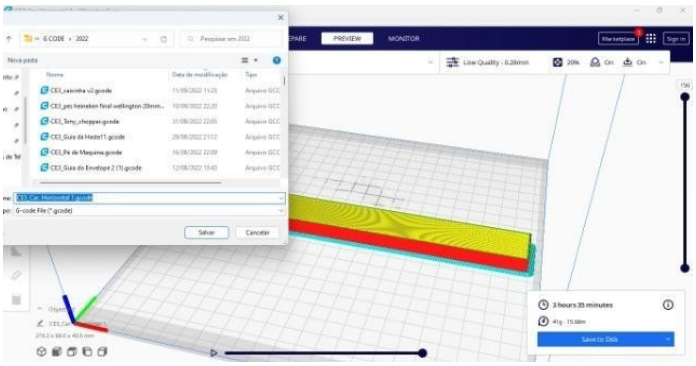


Fonte: Autores (2022)

Com a peça fatiada, a Figura 33 mostrará a peça na forma de camadas, podendo ser visto a evolução da impressão através da barra lateral azul. Outros pontos importantes também são mostrados após o fatiamento como quantidade de material usado e tempo de impressão.

Para salvar o GCODE usa-se o botão “Salvar no Disco” no canto inferior direito da tela, com isso seleciona-se qual pasta será armazenado o arquivo na extensão GCODE.

Figura 33 - Salvamento do arquivo em um USB



Fonte: Autores (2022)

2.3.4. Softwares Utilizados

2.3.4.1. Ultimaker Cura

O *Ultimaker Cura* é atualmente um dos softwares de fatiamento mais utilizados no mundo, sua interface muito bem trabalhada nos traz uma grande facilidade e rapidez para um bom fatiamento de peças 3D além de possuir uma enorme gama de configurações de impressão.

Figura 34 - Logo *Ultimaker Cura*

Fonte: (Loja 3D, 2022)

Dentro deste software pode-se editar cada tipo de impressão, desde impressões rápidas e com menos acabamento até impressões com melhor acabamento, dados a velocidade de impressão à temperatura da mesa e da extrusora. Todos os tipos de materiais estão disponíveis, como PLA, ABS e até mesmo material reciclado, os mesmos podem ser editados de acordo com a temperatura especificada pelo fabricante.

2.3.4.2. Autodesk Fusion 360

O software de projeção Fusion 360 tem por objetivo desenvolver e recriar peças em 3D, sendo muito utilizado na confecção de peças para impressão 3D. Durante o desdobramento deste projeto este software foi, em geral, usado para as projeções básicas das partes em 3D do trabalho e para o corte e criação de encaixes nas peças finais em 3D.

A capacidade do software de se fazer uma projeção rápida mostrou-se essencial para o início do projeto, onde a criação da forma inicial das peças necessitava de uma maleabilidade e rapidez. As peças finais do projeto foram projetadas para um tamanho específico, tendo sido necessário o uso de cortes e a criação de encaixes nas mesmas para a realização de sua impressão.

Figura 35 - Logo Autodesk Fusion

Fonte: (Autodesk, 2022)

2.3.5. Impressora

Para a impressão das peças projetadas, utilizou-se a impressora *Ender* 3, esta impressora é muito utilizada para impressões 3D por ser um equipamento de altíssima qualidade, com uma alta calibração e preço relativamente acessíveis.

Figura 36 - Impressora 3D *Ender* 3

Fonte: (Filipe Flop, 2022)

2.4. Aceleração e Desaceleração

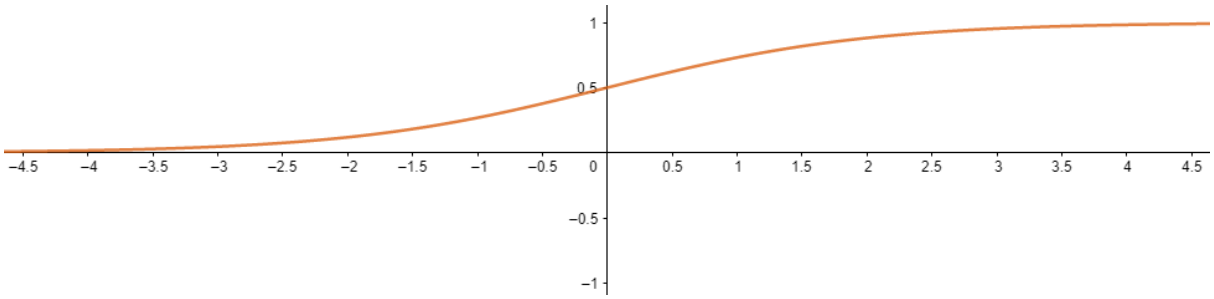
2.4.1. Função Sigmoide

A função sigmoide é uma função matemática que possui grande aplicação em ramos da economia e computação. Define-se ela como:

$$f(x) = \frac{1}{1 + e^{-x}}$$

A Figura 37 apresenta o esboço do gráfico da função sobre o plano cartesiano. Por efeito da subida suave, porém acentuada característica da sigmoide, ela é utilizada para ditar a velocidade dos motores entre o estado de inércia e a velocidade nominal definida, garantindo assim maior tempo de vida aos componentes.

Figura 37 - Traçado da função sigmoide.



Fonte: Autores (2022).

2.4.2. Adequação da Função Sigmoide

Visa-se a adaptação da função sigmoide com finalidade de adequá-la aos parâmetros do projeto, portanto, adiciona-se quatro novas variáveis a função, sendo elas:

a) Amplitude;
b) Abscissa do ponto médio;
c) Coeficiente inclinação da função;
d) Velocidade Inicial.

Atribuindo-as, a função sigmoide transforma-se na função s(x), descrito por pela equação (1):

$$s(t) = v_{min} + \frac{L}{1 + e^{k(t - x_m)}} \qquad (1)$$

Onde:

$$s(t) = velocidade\ no\ instante\ t\ \left[\frac{m}{s}\right]$$

$$v_{min} = velocidade\ mínima\ \left[\frac{m}{s}\right]$$

$L = amplitude\ de\ velocidade\ \left[\frac{m}{s}\right]$

$k = coeficiente\ de\ inclinação\ \left[\frac{1}{s}\right]$

$t = instante\ de\ tempo\ [s]$

$x_m = abcissa\ do\ ponto\ médio\ [s]$

2.4.2.1. Amplitude

A amplitude diz respeito à variação entre o valor máximo e o valor mínimo de determinado intervalo. Assim, a amplitude da velocidade é igual a diferença entre a velocidade máxima e a velocidade mínima:

$$L = v_{max} - v_{min}$$

Dessa forma, relaciona-se a expressão com a equação (1):

$$s(t) = v_{min} + \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}} \qquad (2)$$

2.4.2.2. Ponto Médio

A derivada da velocidade em função do tempo resulta na aceleração. Portanto, a derivada da função s(x), equação (2), resulta em uma função chamada p(x), sendo essa a função que tem como imagem a aceleração instantânea do instante x sobre s(x).

$$p(t) = \frac{d[s(t)]}{dt} \Longrightarrow p(t) = \frac{d\left[v_{min} + \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}\right]}{dt} \Rightarrow p(t) = \frac{d(v_{min})}{dt} + \frac{d\left[\frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}\right]}{dt}$$

$$\Rightarrow$$

$$p(t) = \frac{d\left[\frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}\right]}{dt} \Rightarrow$$

$$p(t) = \frac{\frac{d(v_{max} - v_{min})}{dt}\left[1 + e^{k(t-x_m)}\right] - (v_{max} - v_{min})\frac{d\left[1 + e^{k(t-x_m)}\right]}{dt}}{[1 + e^{k(t-x_m)}]^2} \Rightarrow$$

$$p(t) = \frac{-(v_{max} - v_{min})ke^{(t-x_m)}}{[1 + e^{k(t-x_m)}]^2} \qquad (3)$$

Os pontos máximos e mínimos de uma função é definida como a solução da igualdade entre a derivada da função e 0, relação ao ponto é igual a 0. Em função disso, o ponto de aceleração máxima e o ponto de desaceleração máxima é tido como a igualdade entre a derivada da aceleração e 0.

$$\frac{d[p(x)]}{dt} = 0 \Rightarrow \frac{d\left\{\frac{-(v_{max} - v_{min})ke^{(t-x_m)}}{[1+e^{k(t-x_m)}]^2}\right\}}{dt} = 0 \Rightarrow$$

$$\frac{-k(v_{max} - v_{min})d\left\{\frac{e^{k(t-x_m)}}{[1+e^{k(t-x_m)}]^2}\right\}}{dt} = 0 \Rightarrow$$

$$\frac{d\left\{\frac{e^{k(t-x_m)}}{[1+e^{k(t-x_m)}]^2}\right\}}{dt} = 0$$

Aplica-se a regra da cadeia, desse modo:

$$\frac{\frac{d\left[e^{k(t-x_m)}\right]}{dt}\left[1+e^{k(t-x_m)}\right]^2 - \frac{d\left\{\left[1+e^{k(t-x_m)}\right]\right\}}{dt}e^{k(t-x_m)}}{\{[1+e^{k(t-x_m)}]^2\}^2} = 0 \Rightarrow$$

$$\frac{d\left[e^{k(t-x_m)}\right]}{dt}\left[1+e^{k(t-x_m)}\right]^2 - \frac{d\left\{\left[1+e^{k(t-x_m)}\right]\right\}}{dt}e^{k(t-x_m)} = 0 \Rightarrow$$

$$ke^{k(t-x_m)}[1+e^{k(t-x_m)}]^2 - 2\left[\left[1+e^{k(t-x_m)}\right]ke^{k(t-x_m)}e^{k(t-x_m)}\right] = 0 \Rightarrow$$

$$ke^{k(t-x_m)}\left\{\left[1+e^{k(t-x_m)}\right]^2 - 2\left[1+e^{k(t-x_m)}\right]e^{k(t-x_m)}\right\} = 0 \Rightarrow$$

$$\left[1+e^{k(t-x_m)}\right]^2 - 2e^{k(t-x_m)}\left[1+e^{k(t-x_m)}\right] = 0$$

Assume-se $y = e^{k(t-xm)}$:

$$(1+y)^2 - 2y(1+y) = 0 \Rightarrow 1 + 2y + y^2 - 2y - 2y^2 = 0 \Rightarrow 1 - y^2 = 0 \Rightarrow -y^2 = -1$$

$$\Rightarrow y^2 = 1$$

$$y = \pm 1$$

Como $y = e^{k(t-xm)}$, y não assume valores negativos, uma vez que o membro da direita se trata de uma função exponencial, portanto, apenas a parcela positiva de y será considerada, assim $y = 1$. Relaciona-se as expressões, assim:

$$1 = e^{k(x-xm)} \Rightarrow \ln 1 = \ln e^{k(x-xm)} \Rightarrow 0 = k(x - xm) \Rightarrow 0 = \mathrm{x} - \mathrm{xm} \Rightarrow \mathrm{x} = x_m$$

Conclui-se, portanto, que o ponto médio (M), definido como $M = [x_m, s(x_m)]$ trata-se do ponto de maior aceleração ou maior desaceleração da equação 2.

2.4.2.3. Aceleração Máxima

Aplica-se x_m na função $p(x)$, equação (3):

$$p(x_m) = \frac{-(v_{max}-v_{min})ke^{(x_m-x_m)}}{[1+e^{k(x_m-x_m)}]^2} \Rightarrow p(x_m) = \frac{-(v_{max}-v_{min})ke^0}{[1+e^0]^2} \Rightarrow p(x_m) = \frac{-(v_{max}-v_{min})k}{[1+1]^2} \Rightarrow$$

$$p(x_m) = \frac{-(v_{max}-v_{min})k}{[2]^2} \Rightarrow p(x_m) = \frac{-(v_{max}-v_{min})k}{4}$$

Sabe-se que $p(x_m)$ é igual a aceleração máxima, assim:

$$a_{max} = \frac{-(v_{max}-v_{min})k}{4} \quad (4)$$

Onde:

$a_{max} = aceleração\ máxima\ \left[\frac{m}{s^2}\right]$

$k = coeficiente\ de\ inclinação\ \left[\frac{1}{s}\right]$

$v_{max} = velocidade\ máxima\ \left[\frac{m}{s}\right]$

$v_{min} = velocidade\ mínima\ \left[\frac{m}{s}\right]$

2.4.2.4. Assíntotas Horizontais

Na medida que uma determinada função cresce ou decresce tendendo a encontrar-se com uma reta horizontal, entretanto nunca chegando a encontrarem-se, alcunha-se a reta de assíntota horizontal.

Traçando o gráfico da função sigmoide, supõe-se que existam duas assíntotas horizontais, definindo-as como $g(x) = a$ e $h(x) = b$. Sabe-se que as assíntotas horizontais de uma função são definidas como o limite de x tendendo ao infinito e o limite de x tendendo a menos infinito de uma função. Relaciona-se a definição com as assíntotas g(x) e h(x), conforme as equações (5) e (6).

$$\lim_{x\to\infty} f(x) = a \quad (5)$$

$$\lim_{x\to-\infty} f(x) = b \quad (6)$$

Primeiramente aplica-se na equação (5) a função definida pela equação (2):

$$\lim_{t\to\infty} s(t) = a \Rightarrow \lim_{t\to\infty}[v_{min} + \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}] = a \Rightarrow v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}} = a$$

$$v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + e^{k(\infty - x_m)}} = a \Rightarrow v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + e^{k\infty}} = a \Rightarrow$$

$$v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + e^{\infty}} = a \Rightarrow v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + \infty} = a \Rightarrow v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{\infty} = a \Rightarrow$$

$$v_{min} + 0 = a \Rightarrow v_{min} = a$$

Em seguida, faz-se o mesmo processo substituindo a equação (5) pela equação (6):

$$\lim_{t\to-\infty} s(t) = b \Rightarrow \lim_{t\to-\infty}[v_{min} + \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}] = b \Rightarrow v_{min} + \lim_{t\to-\infty} \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}} = b$$

$$v_{min} + \lim_{t\to-\infty} \frac{v_{max} - v_{min}}{1 + e^{k(-\infty - x_m)}} = b \Rightarrow v_{min} + \lim_{t\to-\infty} \frac{v_{max} - v_{min}}{1 + e^{k(-\infty)}} = b \Rightarrow$$

$$v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + e^{-\infty}} = b \Rightarrow v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1 + 0} = b \Rightarrow v_{min} + \lim_{t\to\infty} \frac{v_{max} - v_{min}}{1} = b \Rightarrow$$

$$v_{min} + v_{max} - v_{min} = b \Rightarrow v_{max} = b$$

Como exposto, $g(x) = a$, como a variável à é igual à v_{min}. Assim, pode-se afirmar $g(x) = v_{min}$. Como a função representa uma assíntota horizontal de $s(t)$, equação (2), a imagem dessa apenas tenderá à v_{min}, contudo, nunca estará no conjunto. Conclui-se que v_{min} não representa um valor possível da função $s(t)$.

$$g(x) = v_{min} \qquad (7)$$

Seguindo a mesma lógica, $h(x) = v_{max}$, a velocidade máxima não é um valor contido dentro da imagem de $s(t)$, apenas valores que tendem a v_{max} estão dentro desse conjunto.

$$h(x) = v_{max} \qquad (8)$$

2.4.2.5. Deslocamento

Dada uma função que descreve a velocidade em função do tempo, a integral definida dessa é igual ao deslocamento.

$$\int_{t_i}^{t_f} v(t)dt = \Delta s \qquad (9)$$

Assim, aplica-se a substitui-se no membro direito da equação (9) a equação descrita por (2) e desenvolve-se a relação:

$$\int_{t_i}^{t_f} s(t)dt = \Delta s \Rightarrow \int_{t_i}^{t_f} \left[v_{min} + \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}\right] dt = \Delta s \Rightarrow$$

$$\int_{t_i}^{t_f} v_{min}\, dt \; + \; \int_{t_i}^{t_f} \frac{v_{max} - v_{min}}{1 + e^{k(t-x_m)}}\, dt = \Delta s$$

Assume-se $y = e^{k(t-x_m)}$, dessa forma:

$$\frac{dy}{dt} = \frac{d\left[e^{k(t-x_m)}\right]}{dt} \Rightarrow \frac{dy}{dt} = ke^{k(t-x_m)} \Rightarrow \frac{dy}{dt} = ky \Rightarrow dt = \frac{dy}{ky}$$

Portanto:

$$\int_{t_i}^{t_f} v_{min}\, dt \; + \; \int_{t_i}^{t_f} \frac{v_{max} - v_{min}}{k(1+y)y}\, dy = \Delta s \Rightarrow$$

$$\int_{t_i}^{t_f} v_{min}\, dt \; + \; \frac{v_{max} - v_{min}}{k} \int_{t_i}^{t_f} \frac{1}{(1+y)y}\, dy = \Delta s$$

Sabe-se que $\frac{1}{(1+y)y} = \frac{1}{y} - \frac{1}{1+y}$, dessa maneira:

$$\int_{t_i}^{t_f} v_{min}\, dt \; + \; \frac{v_{max} - v_{min}}{k} \left(\int_{t_i}^{t_f} \frac{1}{y}\, dy - \int_{t_i}^{t_f} \frac{1}{y+1}\, dy\right) = \Delta s$$

Resolve-se as integrais, assim:

$$v_{min}|_{t_i}^{t_f} + \frac{v_{max} - v_{min}}{k}\left[\ln y\,|_{t_i}^{t_f} - \ln(y+1)\,|_{t_i}^{t_f}\right] = \Delta s$$

Como $y = e^{k(t-x_m)}$, retorne-se y para o membro da esquerda:

$$v_{min}|_{t_i}^{t_f} + \frac{v_{max} - v_{min}}{k}\left[\ln e^{k(t-x_m)}\,|_{t_i}^{t_f} - \ln\left(e^{k(t-x_m)} + 1\right)|_{t_i}^{t_f}\right] = \Delta s \Rightarrow$$

$$v_{min}|_{t_i}^{t_f} + \frac{v_{max} - v_{min}}{k}\left[k(t - x_m)|_{t_i}^{t_f} - \ln\left(e^{k(t-x_m)} + 1\right)|_{t_i}^{t_f}\right] = \Delta s$$

Resolve-se os limites de integração:

$$v_{min}\, x\, t_f - v_{min}\, x\, t_i \; +$$

$$\frac{v_{max} - v_{min}}{k}\left[k\left(t_f - x_m\right) - k(t_i - x_m) - \ln\left(e^{k(t_f - x_m)} + 1\right) - \ln\left(e^{k(t_i - x_m)} + 1\right)\right] = \Delta s \Rightarrow$$

$$v_{min}\left(t_f - t_i\right) + \frac{v_{max} - v_{min}}{k}\left\{k\left[t_f - x_m - (ti - x_m)\right] - \ln\left(\frac{e^{k(t_f - x_m)} + 1}{e^{k(t_i - x_m)} + 1}\right)\right\} = \Delta s$$

Realiza-se a as demais simplificações, resultando na equação que descreve o deslocamento entre o instante inicial e final:

$$\Delta s = v_{min}\left(t_f - t_i\right) + \frac{v_{max} - v_{min}}{k}\left\{k\left(t_f - t_i\right) - \ln\left[\frac{e^{k(t_f - x_m)} + 1}{e^{k(t_i - x_m)} + 1}\right]\right\} \qquad (10)$$

Onde:

$\Delta s = deslocamento\ [m]$

$v_{min} = velocidade\ mínima\ \left[\frac{m}{s}\right]$

$v_{max} = velocidade\ máxima \left[\frac{m}{s}\right]$

$k = coeficiente\ de\ inclinação\ \left[\frac{1}{s}\right]$

$t_i = instante\ inicial\ [s]$

$t_f = instante\ final\ [s]$

$x_m = abcissa\ do\ ponto\ médio\ [s]$

2.4.3. Função Aceleração

Toma-se a equação (2) como base e define-se a velocidade mínima como a velocidade inicial, a velocidade máxima como a velocidade nominal e o coeficiente de inclinação da função como o coeficiente de aclive, de modo obter a seguinte função:

$$a(t) = v_i + \frac{v_{nom} - v_i}{1 + e^{k_a(t - x_m)}} \qquad (11)$$

Onde:

$a(t) = velocidade\ no\ instante\ t\ \left[\frac{m}{s}\right]$

$v_i = velocidade\ inicial\ \left[\frac{m}{s}\right]$

$v_{nom} = velocidade\ nominal\ \left[\frac{m}{s}\right]$

$k_a = coeficiente\ de\ aclive\ \left[\frac{1}{s}\right]$

$t = instante\ de\ tempo\ [s]$

$x_m = abcissa\ do\ ponto\ médio\ [s]$

2.4.3.1. Restrições

Dados os instantes de tempo da aceleração inicial (t_i) e final (t_f), deseja-se encontrar uma função cuja estrutura atenda o modelo proposto pela equação (11) e que possua as seguintes características:

a) Intervalo de tempo da aceleração contido entre os instantes inicial e final.
b) Variação crescente de velocidade entre os instantes inicial e final.
c) Aclive mais acentuado da função contido entre os instantes inicial e final.
d) Velocidade nominal, imagem do instante final, próxima à velocidade nominal definida.

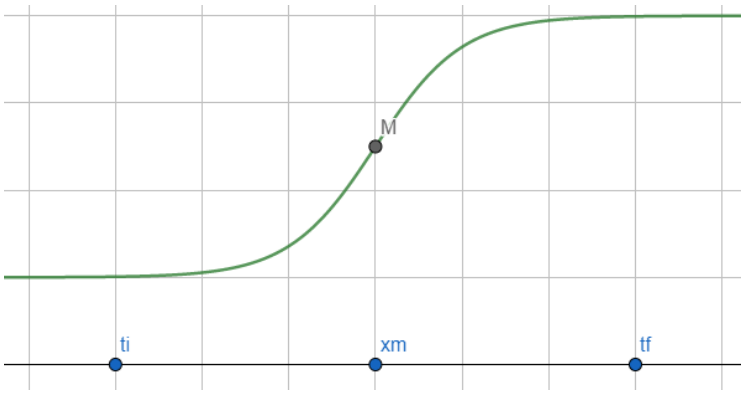
2.4.3.2. Cálculo das Variáveis

2.4.3.2.1. Ponto Médio

Sabe-se que o ponto médio (M), tido como [x_m, $a(x_m)$], é caracterizado como o local de maior aceleração dentro da função $a(t)$. Dessa maneira, como a zona de maior crescimento deve estar contida entre o instante de tempo da aceleração inicial e final, define-se a abscissa do ponto médio como a média aritmética entre esses dois instantes:

$$x_m = \frac{t_f + t_i}{2} \qquad (12)$$

Figura 38 - Ponto médio e os instantes de tempo inicial e final da aceleração

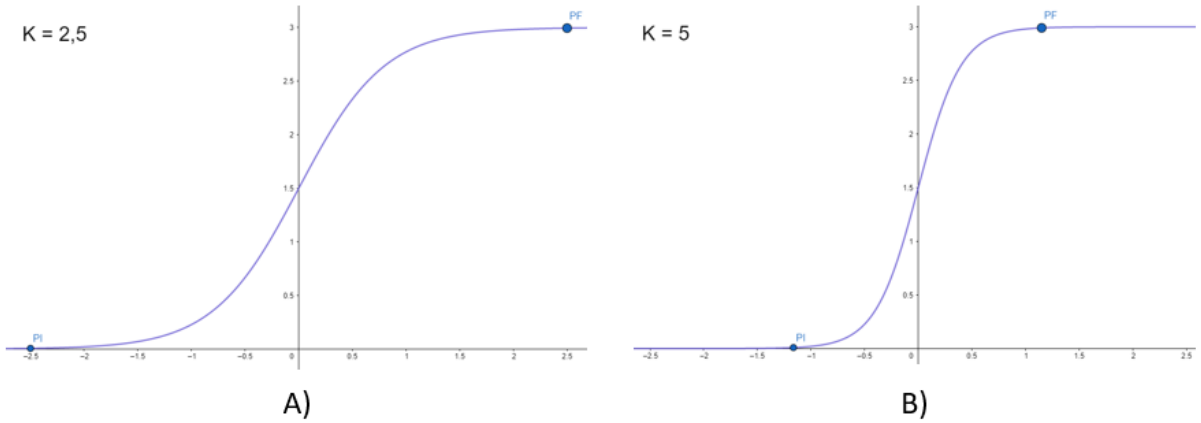


Fonte: Autores (2022).

2.4.3.2.2. Coeficiente de Aclive

Como proposto, a função deverá ser crescente. Portanto, o coeficiente de inclinação da função$s(x)$, equação *(1),* deverá ser um coeficiente de aclive. Observa-se a Figura 39 e constata-se que os únicos valores que variam além do coeficiente de aclive são os instantes iniciais e finais da aceleração. Conclui-se que existe uma relação direta entre os elementos.

Figura 39 - Coeficiente de declive sobre a função s(x).



Fonte: Autores (2022).

Tendo isso em vista, faz-se necessário encontrar o coeficiente de aclive para determinados instantes de aceleração, de modo que a aceleração tenha início no período inicial e termine próxima ao período final, conforme o restringindo.

2.4.3.2.2.1. Cálculo do Coeficiente de Aclive

Como restrito, a imagem da função $a(t)$, equação (11), quando t é igual ao instante de aceleração final é um valor igual ou próximo de velocidade nominal. Contudo, v_{nom} é uma assíntota horizontal, conforme a equação (8), dessa forma não está no conjunto imagem da função, portanto um valor próximo a ele será usado, sendo esse $0{,}99v_{nom}$:

$$a(t_f) = 0{,}99v_{nom} \qquad (13)$$

Sabe-se que x_m corresponde à média aritmética entre os instantes iniciais e finais, equação (12), assim, isolando o instante final de aceleração, tem-se:

$$2x_m = t_f + t_i \Rightarrow t_f = 2x_m - t_i$$

Relaciona-se as expressões, assim:

$$a(2x_m - t_i) = 0{,}99v_{nom} \Rightarrow v_i + \frac{v_{nom} - v_i}{1 + e^{k_a(2x_m - t_i - x_m)}} = 0{,}99v_{nom} \Rightarrow$$

$$v_i + \frac{v_{nom} - v_i}{1 + e^{k_a(x_m - t_i)}} = 0{,}99v_{nom} \Rightarrow \frac{v_{nom} - v_i}{1 + e^{k_a(x_m - t_i)}} = 0{,}99v_{nom} - v_i \Rightarrow$$

$$\frac{1}{1 + e^{k_a(x_m - t_i)}} = \frac{0{,}99v_{nom} - v_i}{v_{nom} - v_i} \Rightarrow 1 + e^{k_a(x_m - t_i)} = \frac{v_{nom} - v_i}{0{,}99v_{nom} - v_i} \Rightarrow$$

$$e^{k_a(x_m - t_i)} = \frac{v_{nom} - v_i}{0{,}99v_{nom} - v_i} - 1 \Rightarrow e^{k_a(x_m - t_i)} = \frac{v_{nom} - v_i}{0{,}99v_{nom} - v_i} - \frac{0{,}99v_{nom} - v_i}{0{,}99v_{nom} - v_i} \Rightarrow$$

$$e^{k_a(x_m - t_i)} = \frac{0{,}01v_{nom}}{0{,}99v_{nom} - v_i} \Rightarrow e^{k_a(x_m - t_i)} = \frac{v_{nom}}{99v_{nom} - 100v_i} \Rightarrow$$

Isola-se o coeficiente de aclive, desse modo:

$$\ln e^{k_a(x_m - t_i)} = \ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right) \Rightarrow k_a(x_m - t_i) = \ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right) \Rightarrow$$

$$k_a = \frac{\ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right)}{x_m - t_i}$$

Relaciona-se a igualdade com a equação (12), chegando-se à equação corresponde ao coeficiente de aclive:

$$k_a = \frac{\ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right)}{\frac{t_f + t_i}{2} - t_i} \Rightarrow k_a = \frac{\ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right)}{\frac{t_f + t_i}{2} - \frac{2t_i}{2}} \Rightarrow k_a = \frac{\ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right)}{\frac{t_f - t_i}{2}}$$

$$k_a = \frac{2\ln\left(\frac{v_{nom}}{99v_{nom} - 100v_i}\right)}{t_f - t_i} \qquad (14)$$

Onde:

$k_a = coeficiente\ de\ aclive\ \left[\frac{1}{s}\right]$

$v_{nom} = velocidade\ nominal\ \left[\frac{m}{s}\right]$

$v_i = velocidade\ inicial\ \left[\frac{m}{s}\right]$

$t_i = instante\ inicial\ de\ aceleração\ [s]$

$t_f = instante\ final\ \ de\ aceleração\ [s]$

2.4.4. Função Desaceleração

Novamente toma-se como base a equação (2) e define-se a velocidade mínima como a velocidade final, a velocidade máxima como a velocidade nominal e o

coeficiente de inclinação como o coeficiente de declive, criando a função desaceleração:

$$d(t) = v_f + \frac{v_{nom} - v_f}{1 + e^{k_d(t - x_m)}} \quad (15)$$

Onde:

$d(t) = velocidade\ no\ instante\ t\ \left[\frac{m}{s}\right]$

$v_f = velocidade\ final\ \left[\frac{m}{s}\right]$

$v_{nom} = velocidade\ nominal\ \left[\frac{m}{s}\right]$

$k_d = coeficiente\ de\ declive\ \left[\frac{1}{s}\right]$

$t = instante\ de\ tempo\ [s]$

$x_m = abcissa\ do\ ponto\ médio\ [s]$

2.4.4.1. Restrições

Dados os instantes de tempo de desaceleração inicial (t_i) e final (t_f), deseja-se encontrar uma função cuja estrutura atenda o modelo proposto pela equação (15) e que possua as seguintes características:

a) Intervalo de tempo de desaceleração contido entre os instantes inicial e final.
b) Variação decrescente de velocidade entre os instantes inicial e final.
c) Declive mais acentuado da função contido entre os instantes inicial e final.
d) Velocidade final, imagem do instante final, próxima à velocidade nominal definida.

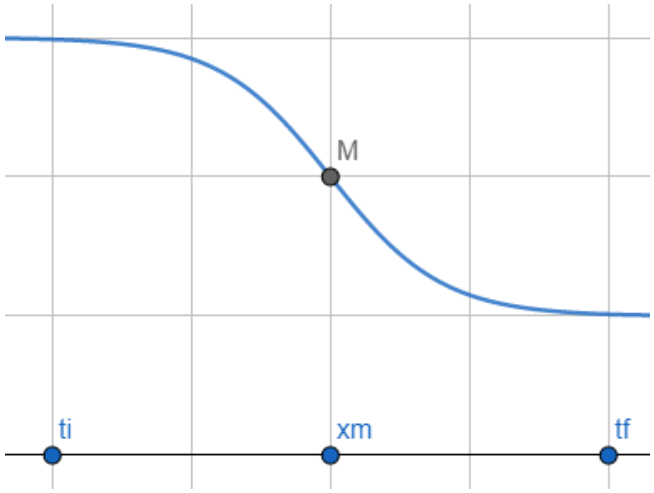
2.4.4.2. Cálculo das Variáveis

2.4.4.2.1. Abscissa do Ponto Médio

Dados os instantes iniciais (t_i) e final (t_f) da desaceleração, pode-se calcular o ponto médio através da média aritmética entre os dois instantes:

$$x_m = \frac{t_f + t_i}{2} \quad (16)$$

Figura 40 - Pontos inicial e final da desaceleração.



Fonte: Autores (2022).

2.4.4.2.2. Coeficiente de Declive

Dada a similaridade entre as restrições, o ponto médio e funções entre a aceleração e desaceleração, realiza-se o mesmo procedimento realizado para o cálculo do coeficiente de aclive, entretanto, define-se $d(t_i) = 0{,}99v_{nom}$.

$$k_d = \frac{2\ln\left(\frac{v_{nom}}{99v_{nom} - 100v_f}\right)}{t_i - t_f} \tag{17}$$

Onde:

$k_d = coeficiente\ de\ declive\ \left[\frac{1}{s}\right]$

$v_{nom} = velocidade\ nominal\ \left[\frac{m}{s}\right]$

$v_f = velocidade\ final\ \left[\frac{m}{s}\right]$

$t_i = instante\ inicial\ de\ desaceleração\ [s]$

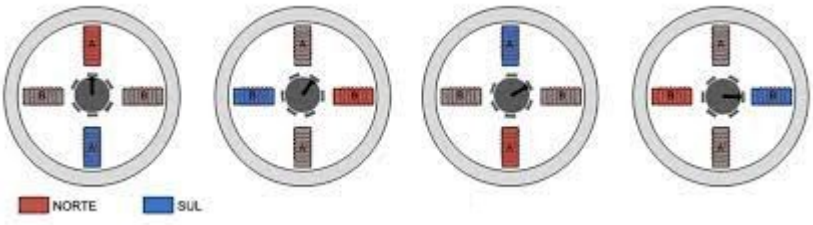
$t_f = instante\ final\ \ de\ desaceleração\ [s]$

2.5. Motor de Passo

Ao contrário dos motores convencionais, os motores de passo caracterizam-se pelo controle extremamente preciso de velocidade, torque e posição. Seu funcionamento se dá através do controle individual de bobinas que atraem os polos do imã fixado

no eixo rotor acoplado a uma caixa de redução interna, garantindo assim o controle extremamente preciso de posição e rotação.

Figura 41 - Princípio de funcionamento do motor de passo de imã permanente



Fonte: (Unesp, 2013)

Entretanto, o controle unitário de cada bobina pode ser deveras complexo, portanto, utiliza-se um sistema embarcado conhecido como módulo de controle. Ao enviar-lhe um pulso elétrico, o motor realiza um deslocamento angular específico, alcunhado de passo. Repete-se o processo novamente, e um novo deslocamento angular igual ao anterior ocorre, assim, após uma quantidade de pulsos por revolução pré-estabelecida, o motor realiza uma volta completa. O valor numérico desse deslocamento é dado por:

$$deslocamento\ angular = \frac{360°}{passos\ por\ revolução}$$

Deseja-se encontrar a expressão que relacione o RPM e o tempo de um único pulso elétrico. Parte-se da relação de que mantendo uma frequência constante de pulsos elétricos de mesma duração, o motor girará a um RPM constante, de modo que a relação é descrita por:

$$RPM = \frac{Pulsos\ por\ Segundo\ x\ 60}{Pulsos\ por\ Revolução}$$

Isola-se os pulsos por segundo:

$$Pulsos\ por\ Segundo = \frac{RPM\ x\ Pulsos\ por\ Revolução}{60}$$

Analisa-se que a cada segundo uma quantidade pulsos que segue a expressão $\frac{RPM\ x\ Pulsos\ por\ Revolução}{60}$ é realizada. Define-se uma variável t a qual dita o tempo em segundos de um único pulso, traçando a seguinte relação:

$$\frac{1\ s}{t} = \frac{\frac{RPM\ x\ Pulsos\ por\ Revolução}{60}}{1}$$

Isola-se t, encontrando:

$$t = \frac{60\ s}{RPM\ x\ Pulsos\ por\ Revolução}$$

Ao multiplicar o membro direito da igualdade por 1 000 000, converte-se o tempo em segundos para os microssegundos, traçando a equação (18) a qual dá o tempo de um único pulso em função do RPM e quantidade de pulsos por revolução:

$$t = \frac{60\ 000\ 000\ \mu s}{RPM\ x\ Pulsos\ por\ Revolução} \quad (18)$$

2.5.1. Velocidade Inicial e Final

A velocidade é uma grandeza vetorial que relaciona o deslocamento em um dado intervalo de tempo.

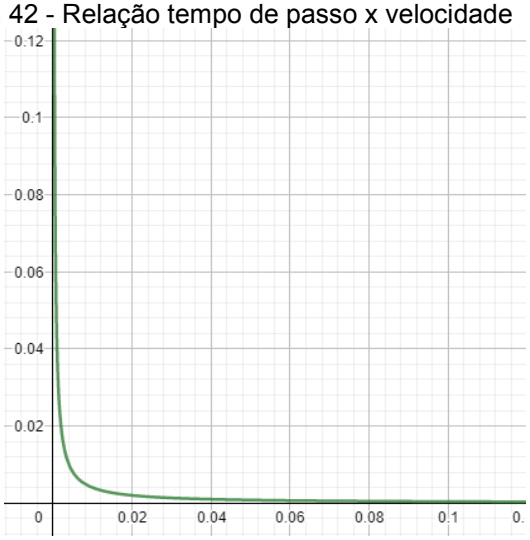
$$\vec{v} = \frac{\Delta s}{\Delta t} \quad (19)$$

A Figura 42 trata-se da representação da relação entre o tempo de passo, equação (18) com algumas com modificações, e a velocidade do motor do projeto. Constata-se que a relação entre o tempo de passo e a velocidade do motor é inversamente proporcional, de modo que quanto mais próxima a velocidade de zero, cada vez mais o tempo de passo tenderá ao infinito, inviabilizando-se a definição da velocidade de inicial e final como zero. Por conta disso, utiliza-se a velocidade inicial do intervalo de aceleração e a velocidade final do intervalo de desaceleração como 0,005 m/s.

$$v_i = 0{,}005\ {}^{m}/_{s} \quad (20)$$

$$v_f = 0{,}005\ {}^{m}/_{s} \quad (21)$$

Figura 42 - Relação tempo de passo x velocidade



Fonte: Autores (2022)

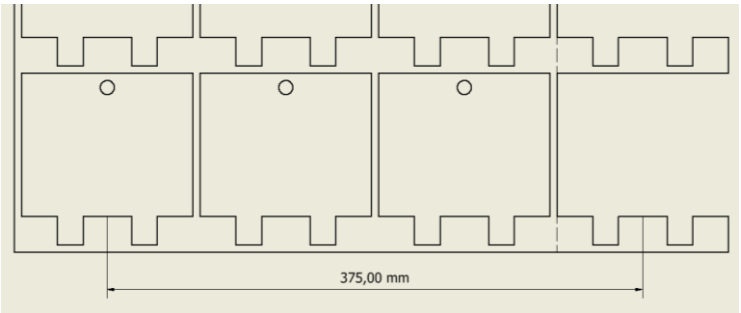
2.5.2. Motor Horizontal

2.4.2.1 Velocidade Nominal

Sabe-se que o deslocamento horizontal útil do projeto é igual a $375\,mm$, Figura 43. Divide-se o valor por 1000, convertendo-o para $0{,}375\,\mathrm{m}$. Desconsiderando os períodos de aceleração e desaceleração, deseja-se deslocar-se em um período de $10\,s$. Traça-se a relação com a equação (19), encontrando $0{,}0375\ {}^{m}/_{s}$:

$$\vec{v} = \frac{0{,}375}{10} \Rightarrow \vec{v} = 0{,}0375\ {}^{m}/_{s}$$

Figura 43 - Deslocamento horizontal útil



Fonte: Autores (2022)

Isso posto, define-se a velocidade nominal horizontal como $0{,}0375\ {}^{m}/_{s}$:

$$v_{nom} = 0{,}0375\ {}^{m}/_{s} \qquad (22)$$

2.5.2.1.1. RPM Nominal

2.5.2.1.1.1. Avanço do Parafuso

O passo de um parafuso é definido como a distância entre dois filetes consecutivos. Já o avanço de um parafuso é caracterizado pelo deslocamento linear por revolução que uma porca realiza na transmissão de movimento por fuso. Geralmente, o passo é igual ao avanço, havendo diferenciação quando o número de entradas for maior que um cuja relação é definida por:

$$Avanço = n° \; de \; entradas \; x \; passo$$

Uma vez que o passo do fuso horizontal é $2\,mm$, e o número de entradas igual a 4, aplicando a relação entre eles, tem-se:

$$Avanço = 4\ x\ 2 \Rightarrow Avanço = 8\ mm$$

Desse modo, a cada revolução do motor, $8\ mm$ lineares são percorridos.

$$Avanço = 8\ mm \quad (23)$$

2.5.2.1.1.2. Cálculo do RPM Nominal

Sabe-se que o avanço do parafuso determina o deslocamento linear realizado a cada revolução. Dessa forma, o produto entre ele e a quantidade de rotações por segundo (RPS) resulta na velocidade.

$$velocidade = avanço\ x\ RPS$$

Isola-se o RPS:

$$RPS = \frac{velocidade}{avanço}$$

Uma vez que o RPM segue a relação $RPM = 60\ x\ RPS$, multiplica-se os dois membros da relação acima, com finalidade de colocar a expressão em função do RPM:

$$60\ x\ RPS = 60\ x\ \frac{velocidade}{avanço} \Rightarrow RPM = 60\ x\ \frac{velocidade}{avanço}$$

Desse modo, traça-se a equação (24):

$$RPM = \frac{60\ x\ velocidade}{avanço} \quad (24)$$

Como exposto, a velocidade é igual a $0{,}0375\ {}^{m}/_{s}$, equação (22), e o avanço $8\ mm$, equação (23), que convertido para metros é $0{,}008\ mm$. Altera-se os valores na equação (24):

$$RPM = \frac{60\ x\ 0{,}0375}{0{,}008} \Rightarrow RPM = 281{,}25\ {}^{1}/_{min}$$

Portanto, o RPM horizontal é igual a $281{,}25\ {}^{1}/_{min}$.

2.5.2.2. Aceleração

2.5.2.2.1. Tempo de Aceleração

Define-se o tempo de aceleração (Δt_a) como de 600 milissegundos. Dividindo-o por 100, faz-se a conversão dele para segundos, resultando em 0,6 segundos.

$$\Delta t_a = 0{,}6\ s$$

O tempo final de aceleração (t_f) é tido como a soma do tempo inicial (t_i) de aceleração somado com o intervalo de aceleração:

$$t_f = t_i + \Delta t_a$$

Relaciona-se as expressões:

$$t_f = t_i + 0{,}6 \qquad (25)$$

2.5.2.2.2. Cálculo da Abscissa Ponto Médio

De acordo com a equação (12), o ponto médio da função de aceleração é tido como a média aritmética entre os instantes inicial e final. Relaciona-se essa com a equação (25), desse modo:

$$x_m = \frac{t_f + t_i}{2} \Rightarrow x_m = \frac{t_i + \ 0{,}6\ + t_i}{2} \Rightarrow x_m = \frac{2t_i + 0{,}6}{2} \Rightarrow x_m = t_i + 0{,}3$$

Conclui-se que a abscissa do ponto médio é igual ao tempo inicial mais 0,3 segundo.

$$x_m = t_i + 0{,}3 \qquad (26)$$

2.5.2.2.3. Cálculo Coeficiente de Aclive

O coeficiente de aclive leva em consideração as seguintes variáveis: velocidade inicial, velocidade nominal, tempos final e inicial de aceleração. Sendo esses descritos respectivamente pelas equações (19), (22) e (25). Relacionando-as com a equação (14), que calcula o coeficiente de aclive, tem-se:

$$k_a = \frac{2\ln\left(\frac{0{,}0375}{99\ x\ 0{,}0375 - 100\ x\ 0{,}005}\right)}{ti + 0{,}6 - t_i} \Rightarrow k_a = \frac{2\ln\left(\frac{0{,}0375}{3{,}7125 - 0{,}5}\right)}{0{,}6} \Rightarrow$$

$$k_a = \frac{\ln\left(\frac{0,0375}{3,2125}\right)}{0,3} \Rightarrow$$

$$k_a = \frac{10\ln\left(\frac{3}{257}\right)}{3}\ {}^{1}/_{s} \qquad (27)$$

Aproxima-se $\ln\left(\frac{3}{257}\right)$ para -4,45046, assim:

$$k_a = \frac{10\ x\ (-4,45046)}{3} \Rightarrow k_a = \frac{-44,5046}{3}\,{}^{1}/_{s} \approx -14,83486\ {}^{1}/_{s}$$

Portanto, o coeficiente de aclive é igual a $-14,83486\ {}^{1}/_{s}$.

$$k_a = -14,83486\ {}^{1}/_{s} \qquad (28)$$

2.5.2.2.4. Aceleração Máxima

A aceleração máxima é descrita pela equação (4). Relaciona-se essa com as relações descritas pelas equações (20), (22) e (28):

$$a_{max} = \frac{-(0,0375 - 0,005)\ x\ (-14,83486)}{4} \Rightarrow a_{max} = \frac{0,0325x\ 14,83486}{4} \Rightarrow$$

$$a_{max} = \frac{0,48213295}{4} \approx 0,12\ {}^{m}/_{s^2}$$

Desse modo, tem-se a aceleração máxima horizontal é igual a $0,12\ {}^{m}/_{s^2}$.

$$a_{max} = 0,12\ {}^{m}/_{s^2} \qquad (29)$$

2.5.2.2.5. Deslocamento

Define-se o deslocamento através da equação (10). Ao relacioná-la com as equações (20), (22), (25), (26), (27) e (28), tem-se:

$$\Delta s = 0,005(\ t_i\ + 0,6 - t_i) + \frac{0,0375 - 0,005}{-14,83486}$$

$$\left\{-14,83486(t_i + 0,6 - t_i) - \ln\left[\frac{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}(t_i+0,6-ti-0,3)} + 1}{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}(t_i-ti-0,3)} + 1}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}005\ x\ 0{,}6 + \frac{0{,}0325}{-14{,}83486}\left\{-14{,}83486\ x\ 0{,}6 - \ln\left[\frac{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}x\ 0{,}3}+1}{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}x\ (-0{,}3)}+1}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}005\ x\ 0{,}6 + \frac{0{,}0325}{-14{,}83486}\left\{-14{,}83486\ x\ 0{,}6 - \ln\left[\frac{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}x\ 0{,}3}+1}{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}x\ (-0{,}3)}+1}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\left\{-8{,}900916 - \ln\left[\frac{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}x\ 0{,}3}+1}{e^{\frac{10\ln\left(\frac{3}{257}\right)}{3}x\ (-0{,}3)}+1}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\left\{-8{,}900916 - \ln\left[\frac{\left(\frac{3}{257}\right)^{\frac{10}{3}x\ 0{,}3}+1}{\left(\frac{3}{257}\right)^{\frac{10}{3}x(-0{,}3)}+1}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\left\{-8{,}900916 - \ln\left[\frac{\left(\frac{3}{257}\right)+1}{\left(\frac{257}{3}\right)+1}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\left\{-8{,}900916 - \ln\left[\frac{\frac{260}{257}}{\frac{260}{3}}\right]\right\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\left\{-8{,}900916 - \ln\left[\frac{3}{257}\right]\right\} \Rightarrow$$

Aproxima-se $\ln\left[\frac{3}{257}\right]$ para -4,450463, assim:

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\{-8{,}900916 - (-4{,}450463)\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\{-8{,}900916 + 4{,}450463\} \Rightarrow$$

$$\Delta s = 0{,}003 - \frac{0{,}0325}{14{,}83486}\{-4{,}450452204\}$$

Aproxima-se $\frac{0{,}0325}{14{,}83486}$ para 0,0021908, desse modo:

$$\Delta s = 0{,}003 + 0{,}0021908\ x\ 4{,}450452204 \Rightarrow \Delta s = 0{,}003 + 0{,}009750051$$

$$\Delta s = 0{,}1275\ m$$

Conclui-se que o deslocamento horizontal é igual a 0,1275 m, que convertidos para milímetros resulta em 12,75 mm.

2.4.2.3. Desaceleração

2.4.2.3.1. Tempo de Desaceleração

Semelhante ao período de aceleração, o período de desaceleração (Δt_d) tem duração de 0,6 segundos. Consoante a isso, afirma-se que o período final de desaceleração (t_f) é igual ao período inicial de desaceleração (t_i) somado 0,6.

$$t_f = t_i + 0{,}6 \tag{30}$$

2.4.2.3.2. Cálculo da Abscissa do Ponto Médio

O ponto médio da função de desaceleração é descrito através da equação (16). Substitui-se o tempo final de desaceleração pela expressão presente no membro direito da equação (30):

$$x_m = \frac{t_f + t_i}{2} \Rightarrow x_m = \frac{t_i + \ 0{,}6 \ + t_i}{2} \Rightarrow x_m = \frac{2t_i + 0{,}6}{2} \Rightarrow x_m = t_i + 0{,}3$$

Desse modo, tem-se a abscissa do ponto médio como a soma entre o instante inicial de desaceleração e 0,3 segundo.

$$x_m = t_i + 0{,}3 \tag{31}$$

2.4.2.3.3. Cálculo do Coeficiente de Declive

A equação (16) descreve a fórmula para o cálculo do coeficiente de declive, aplicando nela os valores dados pelas equações (21, (22) e (31).

$$k_d = \frac{2\ln\left(\frac{0{,}0375}{99\ x\ 0{,}0375 - 100\ x\ 0{,}005}\right)}{t_i - (t_i + 0{,}6)} \Rightarrow k_d = \frac{2\ln\left(\frac{3}{257}\right)}{t_i - t_i - 0{,}6} \Rightarrow k_d = \frac{2\ln\left(\frac{3}{257}\right)}{-0{,}6} \Rightarrow$$

$$k_d = \frac{-10\ln\left(\frac{3}{257}\right)}{3}\ {}^{1}/_{s} \tag{32}$$

Aproxima-se $\ln\left(\frac{3}{257}\right)$ para -4,45046, assim:

$$k_a = \frac{-10\ x\ (-4{,}45046)}{3} \Rightarrow k_a = \frac{44{,}5046}{3}\ {}^{1}/_{s} \approx 14{,}83486\ {}^{1}/_{s}$$

Portanto, o coeficiente de declive é igual a $14{,}83486\ {}^{1}/_{s}$.

$$k_d = 14{,}83486\ {}^{1}/_{s} \tag{33}$$

2.4.2.3.4. Desaceleração Máxima

A aceleração máxima é descrita pela equação (4). Relaciona-se essa com as relações descritas pelas equações(21, (22) e (33):

$$a_{max} = \frac{-(0{,}0375 - 0{,}005)\ x\ (14{,}83486)}{4} \Rightarrow a_{max} = \frac{0{,}0325x(-14{,}83486)}{4} \Rightarrow$$

$$a_{max} = \frac{-0{,}48213295}{4} \approx -0{,}12\ {}^{m}/_{s^2}$$

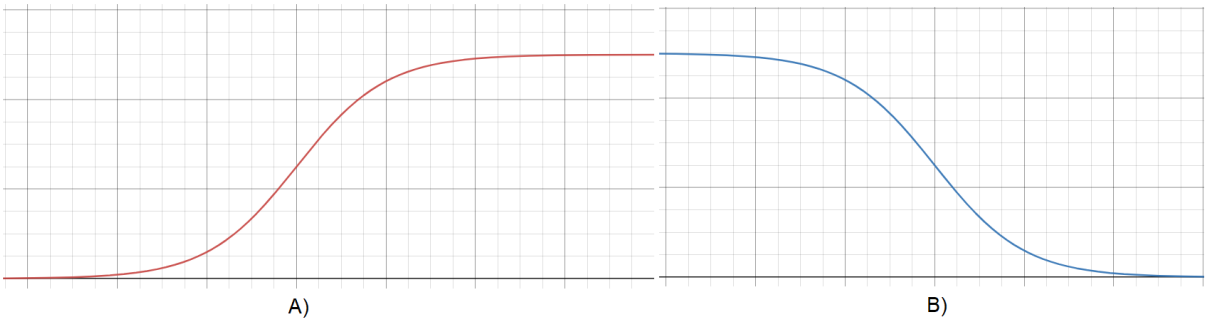
Observa-se que a aceleração máxima tem o sinal negativo, transformando-se em uma desaceleração, dessa maneira afirma-se que a desaceleração máxima horizontal (d_{max}) é igual à $0{,}12\ {}^{m}/_{s^2}$.

$$d_{max} = 0{,}12\ {}^{m}/_{s^2} \tag{34}$$

2.4.2.3.5. Deslocamento

A Figura 44A representa o esboço da função aceleração, enquanto a Figura 44B a mesma função, entretanto com o coeficiente aclive com o sinal invertido. Uma vez que a relação entre o coeficiente de aclive, equação (27), e coeficiente de declive, equação (32) segue o enunciado e as variáveis e restrições são análogas. Afirma-se que o deslocamento durante a desaceleração é igual ao deslocamento durante a aceleração.

Figura 44 - Relação entre a função aceleração e a função desaceleração



Fonte: Autores (2022)

Em razão disso, define-se o deslocamento durante a desaceleração como 12,75 mm.

2.5.3. Motor vertical

2.5.3.1. Velocidade Nominal

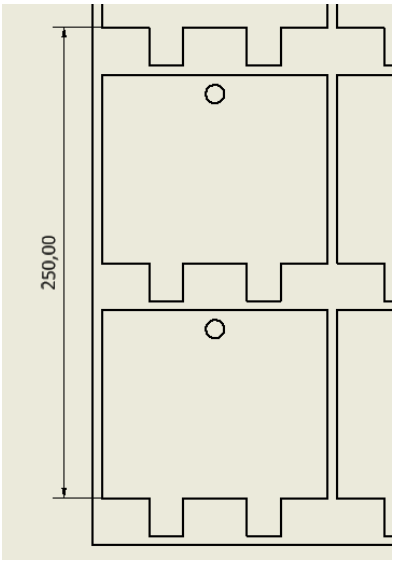
O deslocamento vertical útil do projeto equivale a 250 mm, Figura 45, que quando convertido para metros resulta em 0,25 m. Utiliza-se o novamente o intervalo de tempo de 10 s e aplicando os valores na equação (19), tem-se:

$$\vec{v} = \frac{0{,}25}{10} = 0{,}025 \ {}^{m}/_{s}$$

Assim, determina-se a velocidade nominal vertical como 0,025 m/s.

$$v_{nom} = 0{,}025 \ {}^{m}/_{s} \qquad (35)$$

Figura 45 - Deslocamento vertical útil



Fonte: Autores (2022)

2.4.3.1.1 Cálculo do RPM

Sabendo que o fuso vertical é igual ao fuso horizontal, tem-se o avanço como $0{,}008\,m$. Uma vez possuído isso e a velocidade nominal vertical, equação (35), pode-se encontrar o RPM vertical através da equação (23).

$$RPM = \frac{60\ x\ 0{,}025}{0{,}008} \Rightarrow RPM = \frac{1{,}5}{0{,}008} \Rightarrow RPM = 187{,}5\ {}^{1}/_{min}$$

Assim, o RPM vertical é $187{,}5\ {}^{1}/_{min}$.

2.5.3.2. Aceleração

Utiliza-se o mesmo procedimento realizada na subseção Aceleração para descobrir a aceleração durante o movimento vertical. Isso posto, tem-se a abscissa do ponto médio descrito de maneira igual a equação (26) e o coeficiente de aclive como:

$$k_a = \frac{10 \ln\left(\frac{1}{79}\right)}{3} \quad (36)$$

Onde aproximando $\ln\left(\frac{1}{79}\right)$ para $-4{,}36944$, obtém-se:

$$k_a = \frac{10\ \text{x}\ (-4{,}36944)}{3} \Rightarrow k_a = \frac{-43{,}6944}{3} \Rightarrow k_a = -14{,}5648\ {}^{1}/_{s}$$

Portanto, também se define o coeficiente de aclive como:

$$k_a = -14{,}5648\ {}^{1}/_{s} \quad (37)$$

Já aceleração máxima tem módulo aproximado de $0{,}073\ {}^{m}/_{s^2}$.

$$a_{max} = 0{,}073\ {}^{m}/_{s^2} \quad (38)$$

E o deslocamento sendo igual a 9 mm.

2.5.3.3. Desaceleração

De maneira análoga a aceleração, realiza-se o mesmo procedimento realizado na subseção 2.5.2.2, de modo que a abscissa do ponto médio é dada através da equação (31) e o coeficiente de declive é dado por:

$$k_d = \frac{-10 \ln\left(\frac{1}{79}\right)}{3} \quad (39)$$

Ou aproximando novamente $\ln\left(\frac{1}{79}\right)$ para $-4{,}36944$:

$$k_d = 14{,}5648\ {}^{1}/_{s} \tag{40}$$

Do mesmo modo, a desaceleração máxima tem o módulo aproximado de $0{,}073\ {}^{m}/_{s^2}$.

$$d_{max} = 0{,}073\ {}^{m}/_{s^2} \tag{41}$$

E deslocamento de 9 mm.

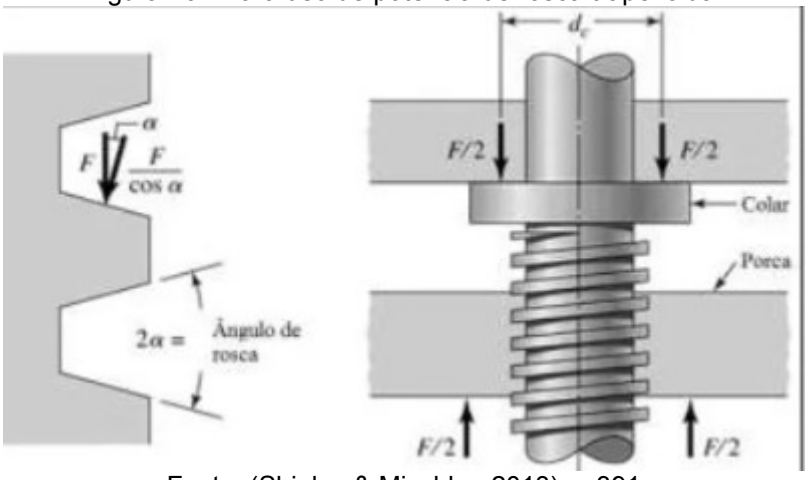
$$\Delta s = 9\ mm$$

2.5.4. Torque do motor

Para determinar o torque, o esforço para mover a carga, o esforço para a rotação do parafuso e suas características geométricas devem ser levados em consideração. Sendo assim, os resultados são diferentes para o movimento vertical ascendente, o movimento vertical descendente e o movimento horizontal.

A Figura 46 representa um parafuso de rosca trapezoidal, modelo escolhido no projeto. Observe-se que a força axial (F) possui uma componente devido ao ângulo de rosca, sendo a abertura entre elas igual a metade do ângulo de rosca do fuso.

Figura 46 - Parafuso de potência de rosca trapezoidal



Fonte: (Shigley & Mischke, 2013), p.391

Para roscas trapezoidais o torque para movimento vertical ascendente é definido pela equação (42). Onde M_t representa o torque necessário para erguer a carga, m a soma das massas dos componentes movidos, a a aceleração, R a força

de resistência: caso a carga não seja guiada, apenas a força peso deve ser considerada, caso contrário, a força de atrito entre a carga e a guia deve ser levada em consideração; μ o coeficiente de atrito entre a flange e o parafuso, d_m o diâmetro externo do parafuso, α a metade do ângulo de rosca e A o passo do parafuso.

$$M_t = \left[ma + \frac{(ma + R)(\mu\pi d_m \sec(\alpha) + A)}{\pi d_m - \mu A \sec(\alpha)}\right]\frac{d_m}{2} \quad (42)$$

Para o movimento descendente, a equação (43) descreve o quanto de torque é necessário.

$$M_t = \left[ma + \frac{(ma + R)(\mu\pi d_m \sec(\alpha) - A)}{\pi d_m + \mu A \sec(\alpha)}\right]\frac{d_m}{2} \quad (43)$$

Já para torque necessário para o movimento horizontal, descrito pela equação (44), fatores como o coeficiente de atrito entre a carga e a superfície de contato (μ_c) e a aceleração da gravidade (g) devem ser levados em consideração.

$$M_t = \left[ma + \frac{m(a + \mu_c g)(\mu\pi d_m \sec(\alpha) + A)}{\pi d_m - \mu A \sec(\alpha)}\right]\frac{d_m}{2} \quad (44)$$

2.5.4.1. Massa Máxima da Caixa

Os motores utilizados no movimento horizontal e no movimento vertical são os mesmos, o motor de passo Nema 17. Sabe-se que ele possui $4\ kgf.cm$, multiplica-se o valor por 0,098, convertendo-o para Newton Metros:

$$Torque = 4\ x\ 0{,}098 = 0{,}392\ Nm$$

Desse modo tem-se que o torque máximo do motor (M_t) é igual a $0{,}392\ Nm$.

$$M_t = 0{,}392\ Nm \quad (45)$$

Isso posto, deve-se calcular a carga máxima da caixa analisando três tipos de movimento, o vertical ascendente, o vertical descendente e o horizontal.

2.5.4.1.1. Movimento Vertical Ascendente

A

Tabela 2 descreve as massas de todos os componentes envolvidos no movimento vertical ascendente e a soma total das massas, sendo essa 1525 g. Converte-se esse para quilogramas dividindo-o por 1000, resultando em 1,525 kg.

Tabela 2 - Componentes movidos no movimento ascendente e descendente

Componentes	**Quantidade**	**Massa Unitária (g)**	**Massa Total (g)**
Motor Nema 17	2	220	440
Carro vertical	1	600	600
Fuso 8 x 700 mm	1	280	280
Rolamento LM8UU	4	36	144
Garra	1	25	25
Flange	2	13	26
Engrenagem	1	10	30
Total:			**1525**

Fonte: Autores (2022)

Para encontrar o torque durante o movimento vertical ascendente utiliza-se a equação (42) que leva em consideração a massa transportada, a aceleração, a metade do ângulo de rosca, o coeficiente de atrito entre o parafuso e a porca, a força resistiva e o diâmetro e avanço do parafuso.

Define-se a massa da caixa como a carga máxima transportada pelo motor (m_{max}) e a aceleração como $0{,}073\ {}^{m}/_{s^2}$, conforme a equação (38). O fuso utilizado tem diâmetro igual a 0,008 m, o avanço de $0{,}008m$, equação (23), e ângulo de rosca igual a 30º. Como α equivale à metade do ângulo de rosca, tem-se α como 15º.

$$\alpha = \frac{30°}{2} = 15°$$

Ainda no mesmo componente, o fuso é feito de aço e a flange de latão. Utiliza-se a Tabela 3 para constatar qual o coeficiente de atrito entre esses materiais, e constata-se que gira em torno de 0,15 e 0,19. Utiliza-se o maior valor do intervalo, 0,19.

Tabela 3 - Coeficiente de atrito entre porca e parafuso

Material do Parafuso	Material da Porca			
	Aço	**Bronze**	**Latão**	**Ferro Fundido**
Aço (seco)	0,15 - 0,25	0,15 - 0,23	0,15 - 0,19	0,15 - 0,25
Aço (lubrificado)	0,11 - 0,17	0,10 - 0,16	0,10 - 0,15	0,11 - 0,17
Bronze	0,08 - 0,12	0,04 - 0,06	-	0,06 - 0,09

Fonte: Elaborada pelo autor com base em (Shigley & Mischke, 2013, p. 396)

Uma vez que o coeficiente de atrito entre as barras e o rolamento linear é praticamente nulo, tem-se a força resistiva R definida pela força peso.

$$R = mg$$

Como a massa é igual à carga máxima transportada e a aceleração da gravidade, aproxima-se a gravidade para $9{,}8\ {}^{m}/_{s^2}$. Traça-se a seguinte expressão:

$$R = 9{,}8m_{max}$$

Uma vez que o torque do motor é igual a 0,392, equação (45). Relaciona-se os valores descritos com a equação (42), aproximando π para 3,1415, assim:

$$0{,}392 = \left[0{,}073m_{max} + \frac{(0{,}073m_{max} + 9{,}8m_{max})(0{,}19\ x\ 3{,}1415\ x\ 0{,}008 \text{ x } \sec(15°) + 0{,}008)}{3{,}1415\ x\ 0{,}008 - 0{,}19 \text{ x } 0{,}008 \text{ x } \sec(15°)}\right] \frac{0{,}008}{2}$$

Aproxima-se a $\sec(15°)$ para 1,035:

$$0{,}392 = \left[0{,}073m_{max} + \frac{(0{,}073m_{max} + 9{,}8m_{max})(0{,}19\ x\ 3{,}1415\ x\ 0{,}008\ x\ 1{,}035 + 0{,}008)}{3{,}1415\ x\ 0{,}008 - 0{,}19 \text{ x } 0{,}008 \text{ x } 1{,}035}\right] \frac{0{,}008}{2} \Rightarrow$$

$$0{,}392 = \left[0{,}073m_{max} + \frac{9{,}873m_{max}(0{,}004942208 + 0{,}008)}{0{,}025132 - 0{,}0015732}\right] 0{,}004 \Rightarrow$$

$$0{,}392 = \left[0{,}073m_{max} + \frac{9{,}873m_{max}(0{,}012942208)}{0{,}0235588}\right] 0{,}004 \Rightarrow$$

$$0{,}392 = [0{,}073m_{max} + 9{,}873m_{max}\ x\ 0{,}549357692]0{,}004 \Rightarrow$$

$$0{,}392 = [0{,}073m_{max} + 5{,}423808493m_{max}]0{,}004 \Rightarrow$$

$$0{,}392 = [5{,}496808493m_{max}]0{,}004$$

Isola-se m_{max}:

$$98 = 5{,}496808493 m_{max}$$

$$\frac{98}{5{,}496808493} = m_{max}$$

Aproxima-se $\frac{98}{5{,}496808493}$ para 17,8:

$$m_{max} = 17{,}8\ kg$$

Dessa maneira, a carga máxima que o motor pode transportar é de 17,8 kg.

Define-se carga máxima transportada (m_{max}) pelo motor como a soma da massa da estrutura mecânica movida (m_e) e a massa máxima da caixa (m_c):

$$m_{max} = m_e + m_c \qquad (46)$$

Como a massa máxima que o motor consegue içar é $17{,}8\ kg$, e a estrutura possui massa de $1{,}515$ kg. Aplica-se a equação (46), assim a massa máxima da caixa é igual à aproximadamente $16{,}25$ kg.

$$17{,}8\ kg = 1{,}525 + c_{max} \Rightarrow c_{max} = 16{,}275\ kg$$

2.5.4.1.2. Movimento Vertical Descendente

Por se tratar de mesmo tipo de movimento, mas em sentido contrário, utiliza-se o mesmo procedimento realizado no movimento vertical ascendente, entretanto substituindo a equação (42) pela equação (43) e invertendo o sinal do torque, uma vez que o sentido de giro é contrário:

$$-0{,}392 = \left[0{,}073 m_{max} + \frac{(0{,}073 m_{max} + 9{,}8 m_{max})(0{,}19\ x\ 3{,}1415\ x\ 0{,}008 \text{ x } \sec(15°) - 0{,}008)}{3{,}1415\ x\ 0{,}008 + 0{,}19 \text{ x } 0{,}008 \text{ x } \sec(15°)}\right]$$

$$\frac{0{,}008}{2}$$

Aproxima-se a $\sec(15°)$ para 1,035:

$$-0{,}392 = \left[0{,}073 m_{max} + \frac{(0{,}073 m_{max} + 9{,}8 m_{max})(0{,}19\ x\ 3{,}1415\ x\ 0{,}008 \text{ x } 1{,}035 - 0{,}008)}{3{,}1415\ x\ 0{,}008 + 0{,}19 \text{ x } 0{,}008 \text{ x } 1{,}035}\right]$$

$$0{,}004 \Rightarrow$$

$$-0{,}392 = \left[0{,}073m_{max} + \frac{9{,}873m_{max}(0{,}19\ x\ 3{,}1415\ x\ 0{,}008\ \text{x}\ 1{,}035 - 0{,}008)}{3{,}1415\ x\ 0{,}008 + 0{,}19\ \text{x}\ 0{,}008\ \text{x}\ 1{,}035}\right] 0{,}004 \Rightarrow$$

$$-0{,}392 = \left[0{,}073m_{max} + \frac{9{,}873m_{max}(0{,}004942208 - 0{,}008)}{0{,}025132 + 0{,}0015732}\right] 0{,}004 \Rightarrow$$

$$-0{,}392 = \left[0{,}073m_{max} + \frac{9{,}873m_{max}(-0{,}003057792)}{0{,}0267052}\right] 0{,}004 \Rightarrow$$

$$-0{,}392 = [0{,}073m_{max} + 9{,}873m_{max}(-0{,}114501745)]0{,}004 \Rightarrow$$

$$-0{,}392 = [0{,}073m_{max} - 1{,}130475728m_{max}]0{,}004 \Rightarrow$$

$$-0{,}392 = [-1{,}057475728m_{max}]0{,}004 \Rightarrow$$

Isola-se m_{max}:

$$-98 = -1{,}057475728m_{max}$$

$$\frac{98}{1{,}057475728} = m_{max}$$

Aproxima-se $\frac{98}{1{,}057475728}$para 92,67:

$$m_{max} = 92{,}67\ kg$$

Aplica-se a equação (46), de modo a relacionar a carga máxima de transporte e a massa máxima da caixa e constata-se que essa é igual a aproximadamente 91 kg.

$$96{,}67 = 1{,}525 + m_c \Rightarrow m_c = 91{,}145\ kg$$

Conclui-se que como 91 kg é superior ao encontrado no movimento de sentido contrário a massa máxima da caixa continuará sendo de 16,25 kg.

2.5.4.1.3. Movimento Horizontal

A

Tabela 4 informa os componentes mecânicos movidos no movimento horizontal e respectivas massas. Observa-se que a soma das massas dos componentes

mecânicos é igual a 3003 g que, quando convertido para quilogramas, o resultado é de aproximadamente 3 kg.

Tabela 4 - Componentes movidos no movimento horizontal

Componente	Quantidade	Massa Unitária (g)	Massa Total (g)
Motor Nema 17	2	220	440
Carro vertical	1	660	660
Carro horizontal	1	196	196
Suporte superior	1	218	218
Flange	4	13	52
Barra 8x700 mm	2	280	560
Fuso M8x700 mm	2	280	560
Garra	1	25	25
Engrenagem	1	10	10
Rolamento LM8UU	6	36	216
Mancal KFL08	2	33	66
Total:			3003

Fonte: Autores (2022)

Através da equação (44), calcula-se o torque necessário para a realização do movimento horizontal. Dentre as variáveis da equação tem-se a massa, a aceleração, o coeficiente de atrito entre a superfície e a peça movida, o coeficiente de atrito entre a porca e o parafuso, metade do ângulo de rosca e o diâmetro e avanço do fuso.

A massa é definida como a carga máxima transportada pelo motor (m_{max}) e a aceleração, conforme a equação (29), $0{,}12\ {}^{m}/_{s^2}$. O fuso e as flanges correspondem aos mesmos utilizados no movimento vertical, dessa forma, o diâmetro, o avanço, o coeficiente de atrito entre eles e α é de, respectivamente, $0{,}008\ \mathrm{m}$, $0{,}008\ \mathrm{m}$ e 0,19 e 15º.

Considerando o atrito entre as superfícies de contato nulo, uma vez que o rolamento tem coeficiente de atrito praticamente insignificante, e π como 3,1415, aplica-se os valores na equação (44):

$$0{,}392 = \left[0{,}12m_{max} + \frac{0{,}12m_{max}(0{,}19\ x\ 3{,}1415\ x\ 0{,}008\ x\ \sec(15°) + 0{,}008)}{3{,}1415\ x\ 0{,}008 - \ 0{,}19\ x\ 0{,}008\ x\ \sec(15°)}\right]\frac{0{,}008}{2}$$

Aproxima-se $\sec(15°)$ para 1,035:

$$0{,}392 = \left[0{,}12m_{max} + \frac{0{,}12m_{max}(0{,}19\ x\ 3{,}1415\ x\ 0{,}008\ x\ 1{,}035 + 0{,}008)}{3{,}1415\ x\ 0{,}008 - \ 0{,}19\ x\ 0{,}008\ x\ 1{,}035}\right]\frac{0{,}008}{2} \Rightarrow$$

$$0{,}392 = \left[0{,}12m_{max} + \frac{0{,}12m_{max}(0{,}004942208 + 0{,}008)}{0{,}025132 - \ 0{,}0015732}\right]0{,}004 \Rightarrow$$

$$0{,}392 = \left[0{,}12m_{max} + \frac{0{,}12m_{max}(0{,}012942208)}{0{,}0235588}\right]0{,}004 \Rightarrow$$

$$0{,}392 = [0{,}12m_{max} + 0{,}12m_{max}\ x\ 0{,}549357692]0{,}004 \Rightarrow$$

$$0{,}392 = [0{,}12m_{max} + 0{,}065922923m_{max}]0{,}004 \Rightarrow$$

$$0{,}392 = [0{,}185922923m_{max}]0{,}004$$

Isola-se m_{max}:

$$98 = 0{,}185922923m_{max} \Longrightarrow m_{max} = \frac{98}{0{,}185922923}$$

Aproxima-se $\frac{98}{0{,}185922923}$ para 527:

$$m_{max} = 527\ kg$$

Utilizando a equação (46), tem-se que a massa máxima da caixa é igual a 524 kg, valor que excede muito os 16,25 kg determinados pelo movimento vertical ascendente. Desse modo, determina-se que a massa máxima da caixa deve ser de 16,25 kg.

2.6. Operações Eletromecânicas

Para o funcionamento do projeto, tem-se três operações eletromecânicas, a referenciação, a coleta e a entrega.

2.6.1. Referenciação

O processo de referenciação baseia-se em definir todas as coordenadas cartesianas dos eixos como zero, ou seja, posicionar o carrinho na origem possibilitando a realização das futuras operações que necessitam de coordenadas.

Ao energizar o projeto faz-se necessário a referenciação, uma vez que não é possível saber as coordenadas exatas do carrinho no momento. Em todos os eixos o processo é semelhante, define-se o sentido em que o carrinho se movimentará para ir de encontro ao sensor fim de curso correspondente. Enquanto o sensor não

estiver pressionado, realiza-se um pulso no motor correspondente, fazendo-o ir de encontro ao sensor. Quando pressionado, zera-se a coordenada correspondente ao eixo.

A Figura 47 apresenta um fluxograma do procedimento de referenciação, enquanto a Figura 48 as lógicas do processo em cada eixo.

Figura 47 - Fluxograma do processo de referenciação do carro de transporte

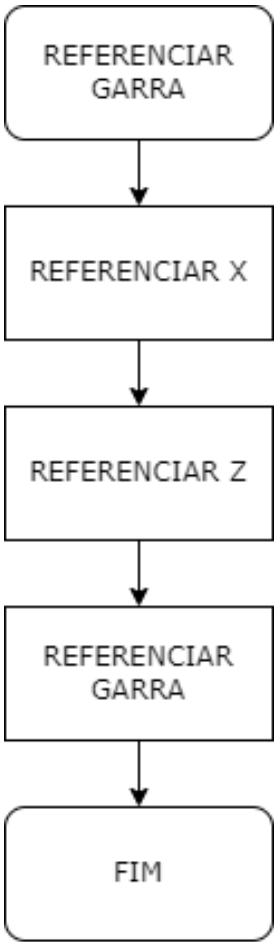
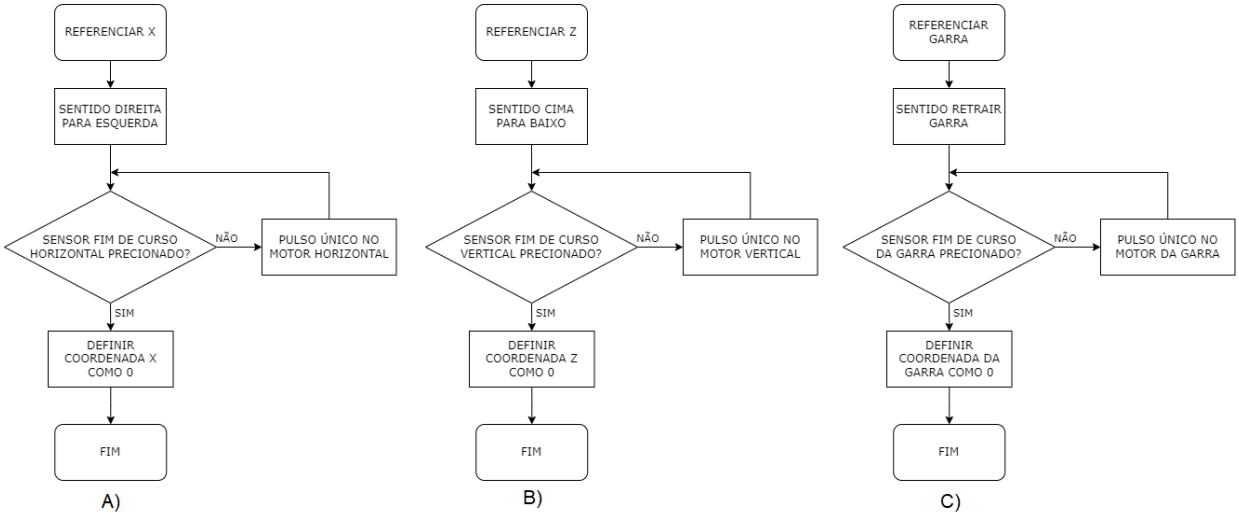


Figura 48 - Fluxograma do processo de referenciação individual dos eixos



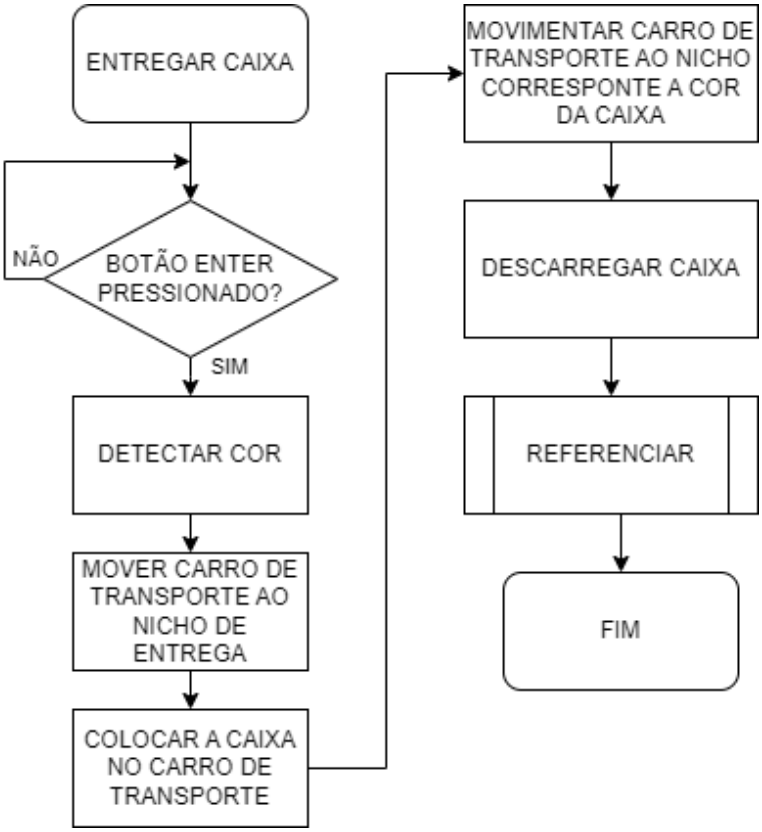
Fonte: Autores (2022)

2.6.2. Entrega

O procedimento de entrega funciona através da detecção da cor da caixa presente no nicho de entrega e, posteriormente, o encaminhamento ao nicho correspondente à cor.

Pressiona-se o botão ENTER de modo a instruir o sensor de cor a ler a caixa. Detecta-se sua cor e tem-se início o processo de encaminhamento dela ao local condizente à cor. Descarrega-a lá e, após isso, o carro transportador retorna à origem. Procedimento visualizado através da Figura 49.

Figura 49 - Fluxograma do processo de entrega da caixa



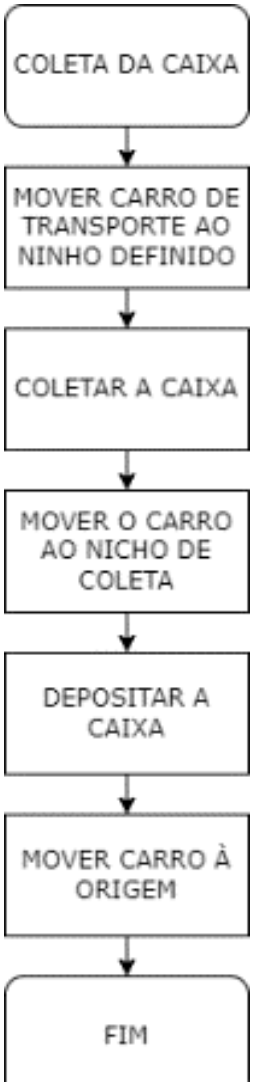
Fonte: Autores (2022)

2.6.3. Coleta

A execução da coleta baseia-se na retirada da caixa do nicho selecionado transportando-a ao nicho de coleta.

Quando selecionado o nicho, o carrinho move-se até lá, coleta a caixa e a deposita no nicho de coleta.

Figura 50 - Fluxograma do processo de coleta da caixa



Fonte: Autores (2022)

2.7. Botões de Navegação e Seleção

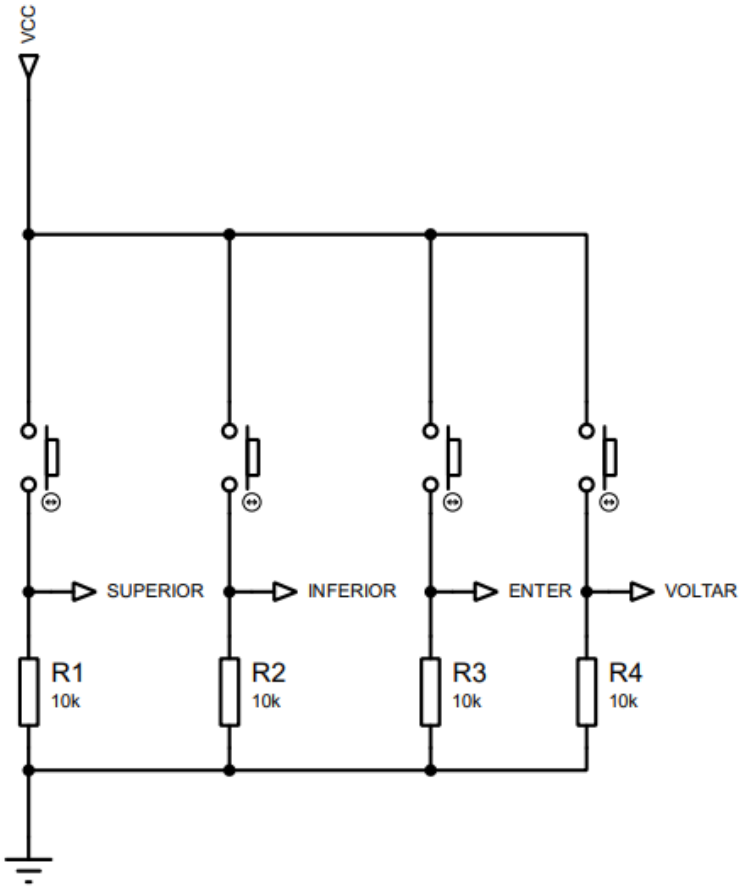
Para interação homem-máquina o projeto consta com a presença de 4 botões: SUPERIOR, INFERIOR, ENTER e VOLTAR. Os botões SUPERIOR e INFERIOR têm função de navegação dentro da tela de visualização. Já os botões ENTER e VOLTAR são responsáveis por mudança de telas e, no caso do primeiro, a confirmação de determinada operação eletromecânica.

2.7.1. Esquema Elétrico

Representa-se o esquema elétrico dos botões de seleção através da Figura 51.

.

Figura 51 - Desenho do circuito elétrico dos botões de seleção



Fonte: Autores (2022)

Quatro botões estão ligados em paralelo. Enquanto não pressionados, sua respectiva saída apresentará nível lógico 0 graças ao resistor de *pull-down*. Ao pressionar algum deles, sua respectiva saída terá nível lógico 1. O *input* VCC trata-se da alimentação do circuito proporcionada pelo microcontrolador, geralmente 3,3 V ou 5 V.

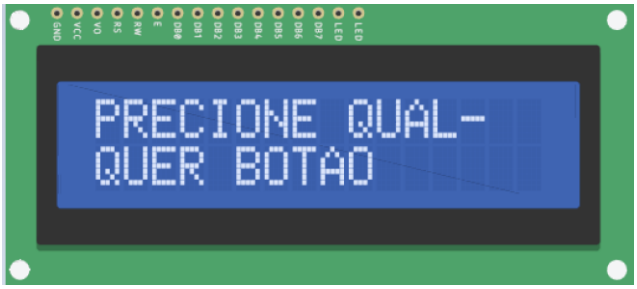
2.8. Painel de Visualização

A interface homem-máquina é realizada através de um painel LCD, um componente eletrônico capaz de exibir informações através de textos. Nele o operador define qual operação qual operação eletromecânica será realizada através dos botões de navegação e controle.

2.8.1. Tela Início

Logo ao energizar o projeto, o painel LCD apresenta a mensagem “PRECIONE QUALQUER BOTÃO”. Ao pressionar algum deles, inicia-se o processo de referenciação, garantindo que todas as demais operações ocorram sem erros de referência e, após isso, a tela modo é visualizada.

Figura 52 - Tela início

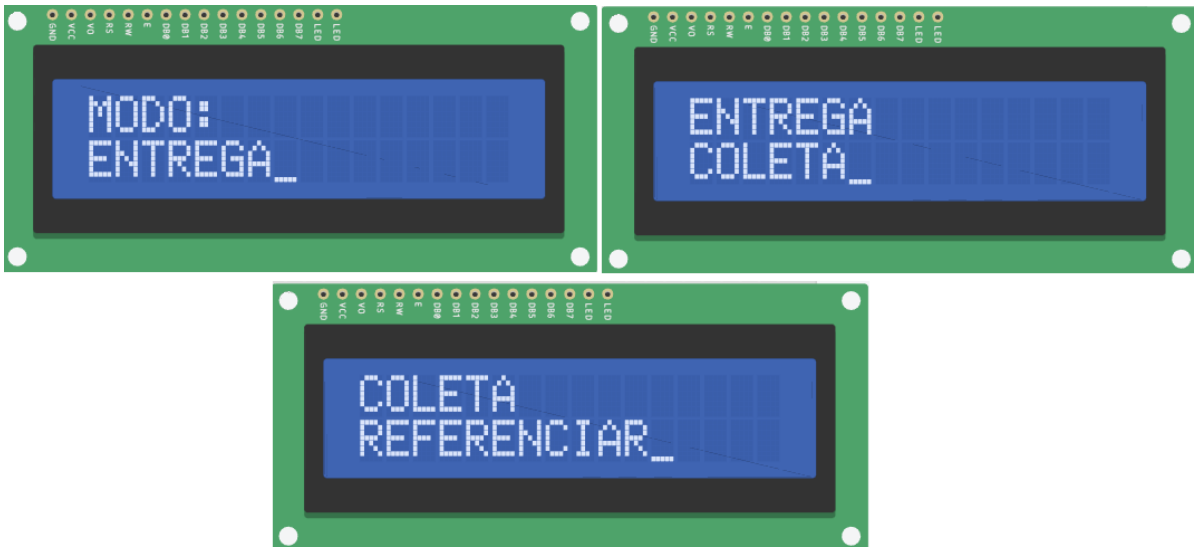


Fonte: Autores (2022)

2.8.2. Tela Modo

A tela modo baseia-se na escolha da operação eletromecânica. Ao pressionar o botão SUPERIOR e INFERIOR navega-se sobre ela e ao pressionar o botão ENTER, seleciona-se a operação indicada pelo cursor, direcionando-o para a tela de respectivo nome.

Figura 53 - Tela modo



Fonte: Autores (2022)

2.8.3. Tela Entrega

Quando selecionada, inicia-se a detecção do sensor de presença de caixa, exibindo a mensagem “AGUARDANDO CAIXA”. Iniciando o processo descrito pela Subseção 2.6.2..

Figura 54 - Tela aguardando caixa

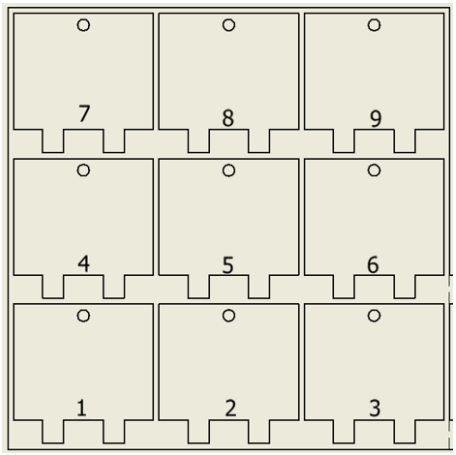
Fonte: Autores (2022)

Quando detectada a cor da caixa, muda-se a mensagem da tela para “ENTREGANDO CAIXA...” e quando finalizado o procedimento, para “ENTREGA CONCLUÍDA”, onde espera-se a ação de algum botão para o retorno para tela modo.

2.8.4. Tela Coleta

A tela de coleta tem função de apresentar todos os nichos numerados possibilitando a seleção para realização do procedimento de coleta explicado pela Subseção 2.6.3. Descreve-se os números e seus respectivos nichos através da Figura 55.

Figura 55 - Numeração dos nichos



Fonte: Autores (2022)

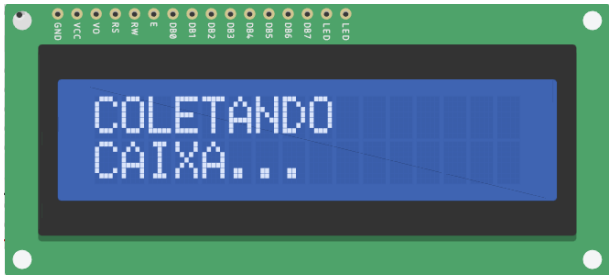
Apresenta-se ao adentrá-la a numeração dos nichos, ilustrado pela Figura 56, cuja interação é realizada novamente pelos botões SUPERIOR e INFERIOR que assumem papel de navegação e o botão ENTER o de seleção.

Figura 56 - Tela coleta

Fonte: Autores (2022)

Quando selecionado o nicho, a mensagem “COLETANDO CAIXA...” é exposta, de modo a iniciar o processo de coleta.

Figura 57 – Tela coletando caixa

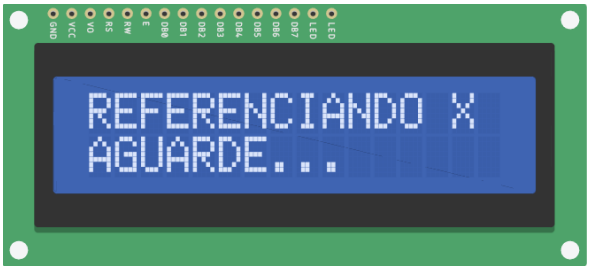


Fonte: Autores (2022)

2.7.5 Tela Referenciação

Caso deseje-se realizar novamente a referenciação por qualquer motivo, seleciona-se a tela de referenciação, ocorrendo o processo explicado pela Subseção 2.6.1.
Antes de iniciada a referenciação do eixo X, exibe-se a mensagem “REFERENCIANDO X”.

Figura 58 - Tela referenciando eixo X



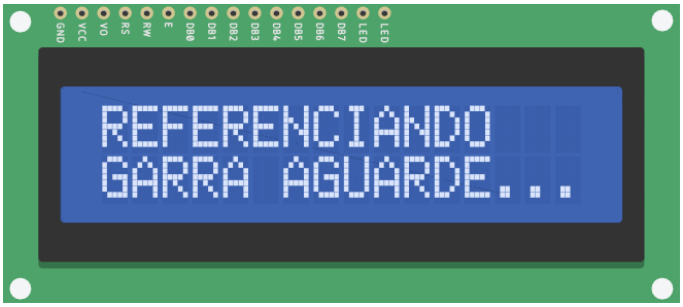
Fonte: Autores (2022)

Posteriormente, o mesmo princípio repete-se para Z e a garra, ilustrados pelas figuras Figura 59 e Figura 60, respectivamente.

Figura 59 - Tela referenciando eixo Z

Fonte: Autores (2022)

Figura 60 - Tela referenciando garra



Fonte: Autores (2022)

E ao finalizado o procedimento, tem-se o texto da tela como “REFERENCIAÇÃO COMPLETA”, aguardando algum botão ser pressionado.

Figura 61 – Tela referenciação completa

Fonte: Autores (2022)

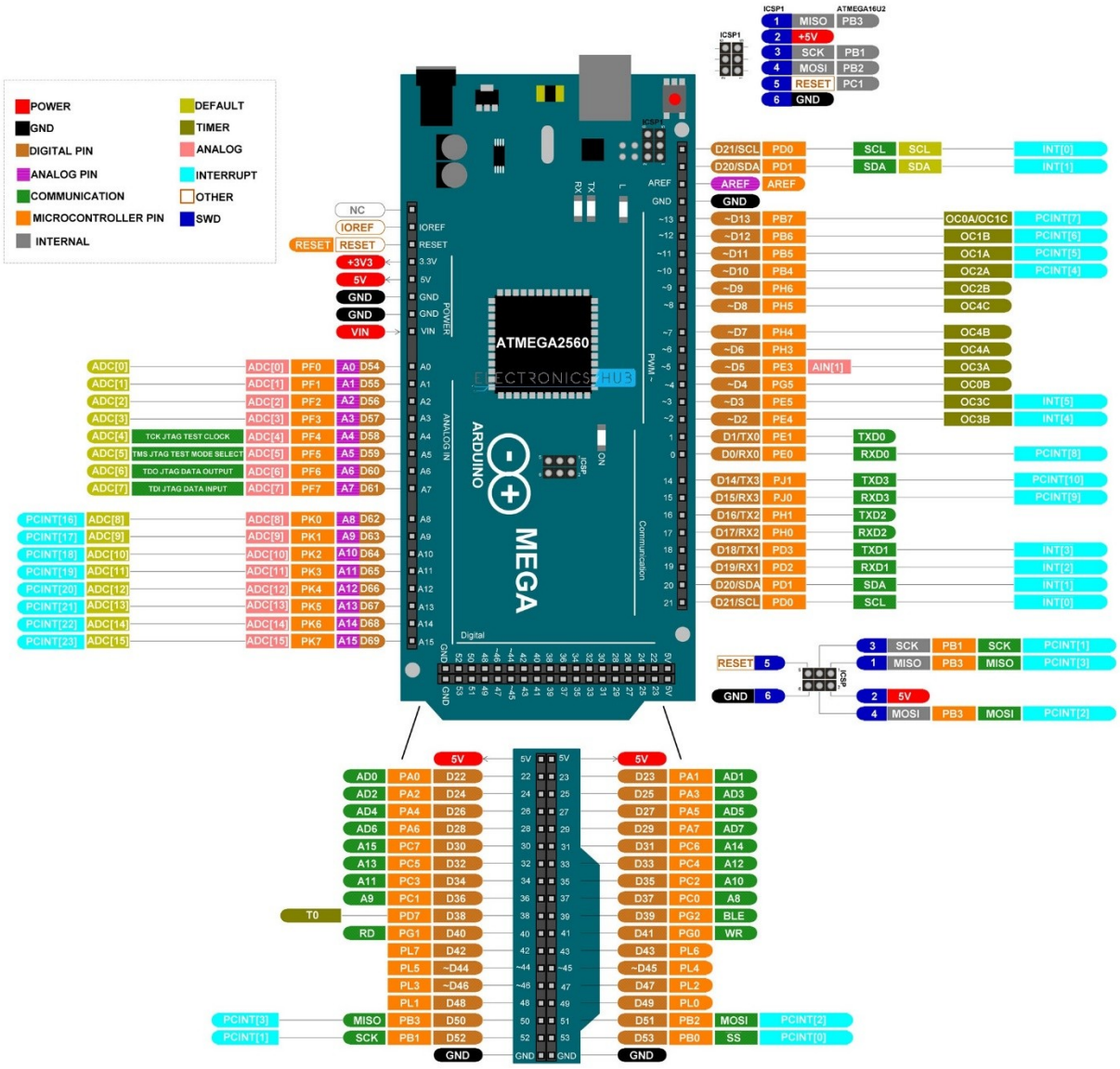
2.9. Circuito Elétrico

O Arduino é uma plataforma de prototipagem idealizada para o desenvolvimento de automações através de uma programação e montagem eletrônica simples, através das diversas portas contidas em sua estrutura que permitem a conexão direta de

atuadores e sensores, dispensando a operação de soldagem ou qualquer outra operação similar. Devido a isso, utiliza-se o a placa Arduino Mega como o microcontrolador empregado pelo projeto.

A Figura 62 revela através de uma ilustração a pinagem do Arduino Mega. Percebe-se que a principal característica da placa é o elevado número de pinos contidos em sua estrutura, entre eles, 54 digitais e 16 analógicos, permitindo o controle e leitura de diversos componentes.

Figura 62 - Pinagem do Arduino Mega



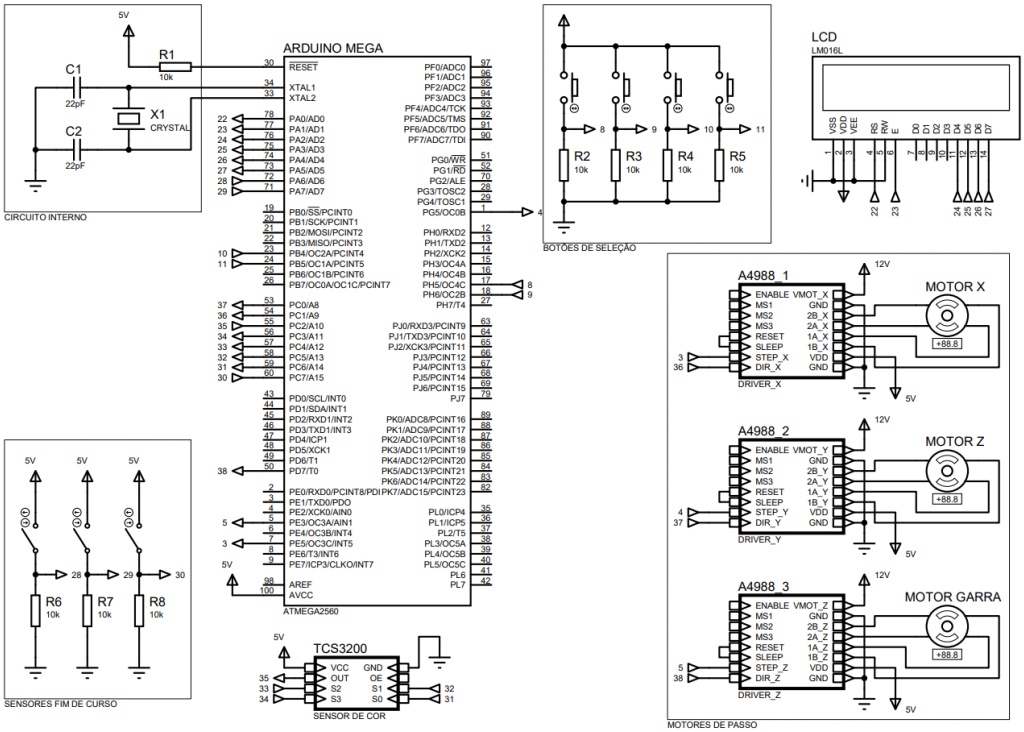
Fonte: (Teja, 2021)

E devido a extensa quantidade de portas contidas no Arduino Mega, necessita-se de um processamento que possibilite a leitura de todas elas em instantes praticamente

insignificantes. Portanto, o processador empregado é o ATMEGA2560 que possui frequência de 16 MHz, ou seja, ele é capaz executar 16 000 000 de instruções em apenas um segundo.

Dessa maneira, através da utilização da Figura 62, a qual apresenta as conexões entre as portas e o microprocessador, desenvolve-se o esquema entre todos os componentes eletrônicos envolvidos no projeto, sendo ele ilustrado através da Figura 63.

Figura 63 - Esquema Elétrico do circuito elétrico



Fonte: Autores (2022)

2.10. Programação

A programação do projeto é realizada na linguagem Arduino, uma vez que o placa de prototipagem utilizada é o Arduino Mega. Essa linguagem de programação baseia-se no controle e leitura dos níveis lógicos no caso das portas digitais, e nas portas analógicas, a leitura e controle dos valores. Por padrão, escreve-se o código na linguagem C++, utilizando modelos e funções definidos pela linguagem Arduino. Como a linguagem baseia-se em C++, utiliza-se o paradigma de programação orientada a objetos.

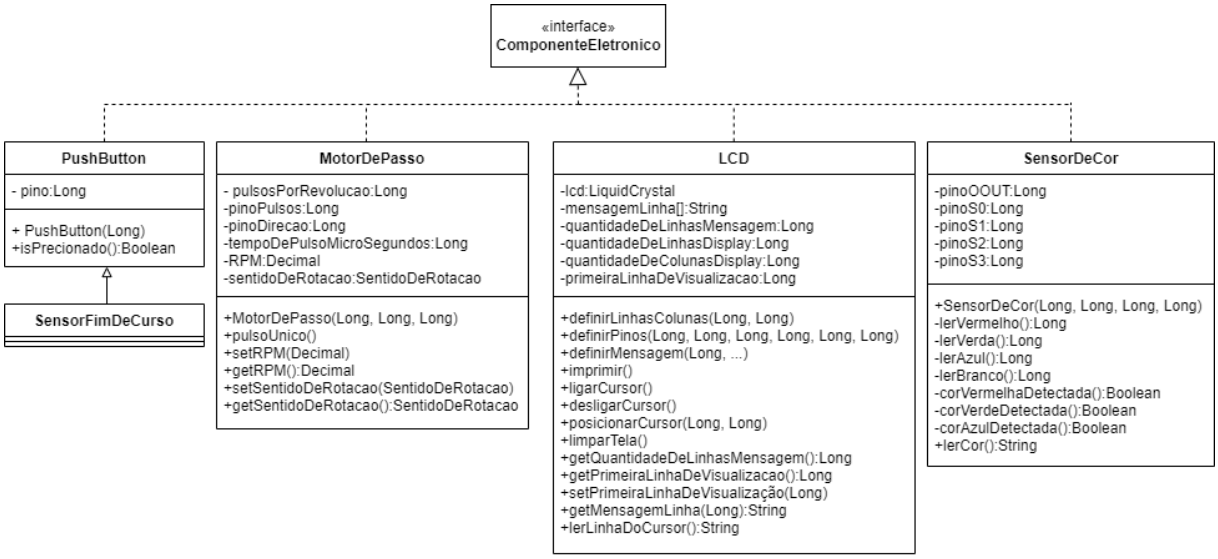
Basicamente a orientação a objetos trata-se da definição de classes que definem um objeto da vida real seja ela concreto ou abstrato, no caso do projeto, componentes e circuitos mecatrônicos. E através de métodos e atributos, suas características e comportamentos são definidos.

2.10.1. Componentes Eletrônicos

Inicialmente cria-se uma interface alcunhada de “ComponenteEletronico”. Essa interface deverá ser implementada a cada nova classe de componente eletrônico criada de modo a respeitar o polimorfismo, um dos pilares da orientação a objetos. Dessa maneira, cria-se cada classe relacionada a um componente eletrônico implementado “ComponenteEletronico”.

A Figura 64 trata-se da representação das classes através de um diagrama UML, nota-se que o sensor fim de curso herda todas as características do *pushbutton*, uma vez que seus funcionamentos são os mesmos.

Figura 64 - Diagrama de classes UML: componentes eletrônicos



Fonte: Autores (2022)

2.10.2. Circuitos Eletrônicos

Os circuitos eletrônicos tratam-se- na interação entre componentes eletrônicos com finalidade de realizar alguma ação. Dentro da programação, pode-se visualizá-los através da agregação entre classes em razão do circuito depender diretamente de

cada componente, entretanto, os componentes não dependem do circuito para sua existência. Seguindo novamente o polimorfismo, cria-se uma interface intitulada de “CircuitosEletronicos” a qual deve ser implementada em sempre que houver a criação de um novo circuito.

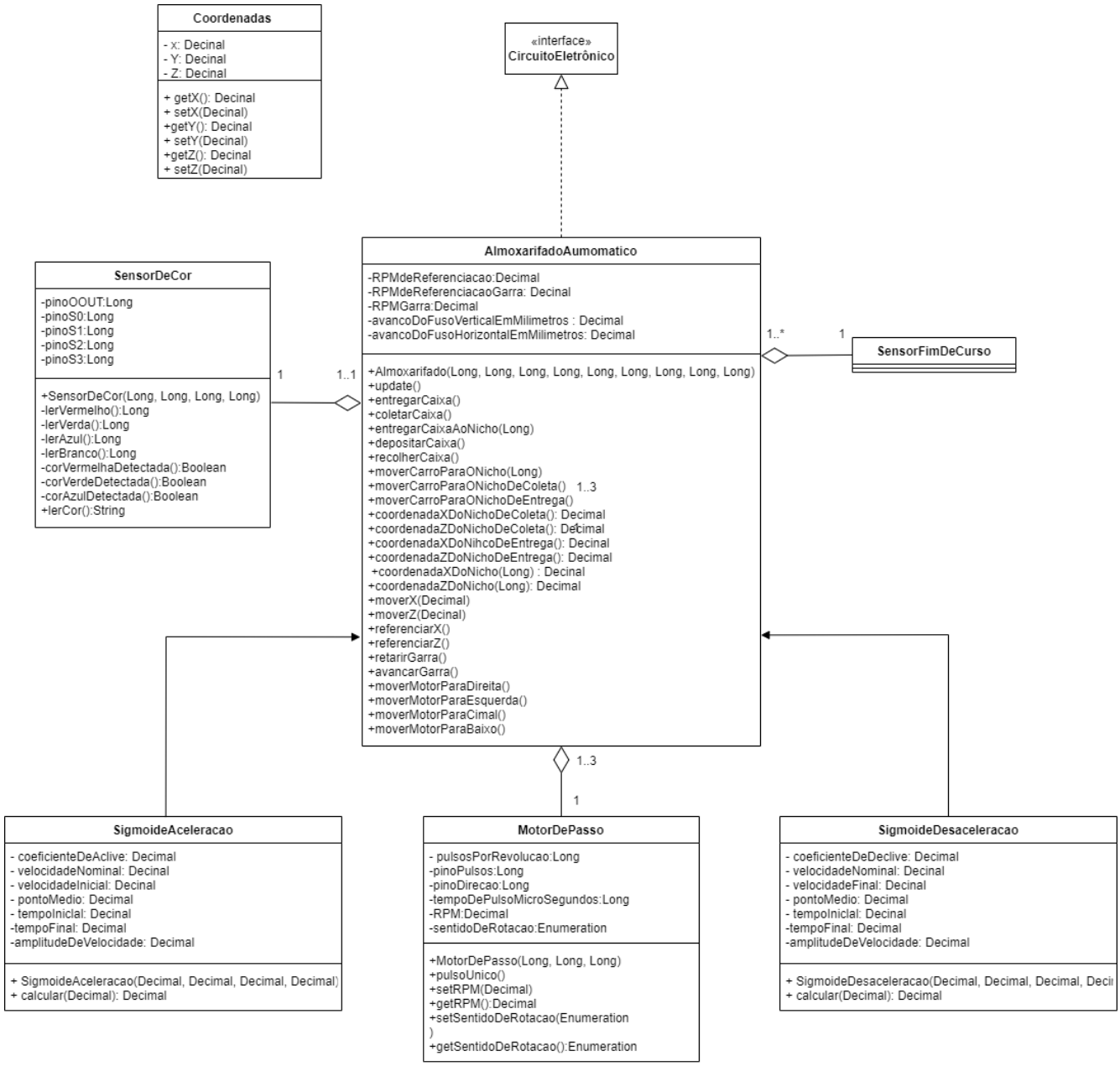
2.10.2.1. Almoxarifado

Os componentes que compõem o almoxarifado são:

a) Motor de passo
b) Sensor de cor
c) Sensor de curso

Dessa maneira, associasse as classes desses respectivos componentes, de modo a formar o almoxarifado. Interação demostrada através da Figura 65.

Figura 65 - Diagrama de classes UML: almoxarifado

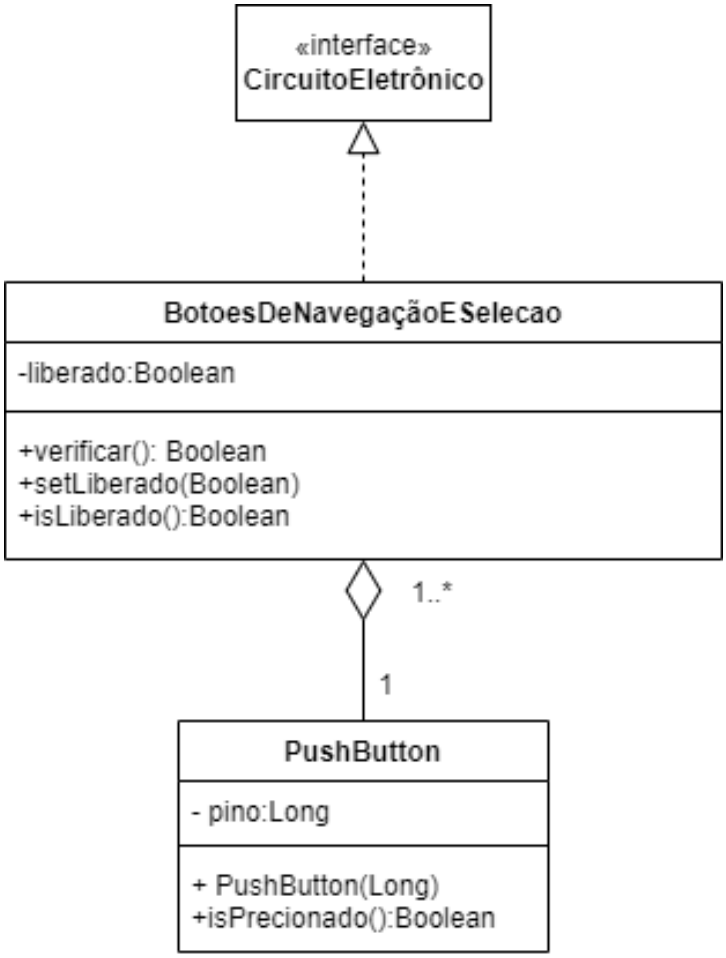


Fonte: Autores (2022)

2.10.2.2. Botões de Navegação e Seleção

Dentre os componentes envolvidos na montagem dos botões de navegação e controle, Figura 51, estão *pushbuttons* e resistores. Entretanto os resistores não têm aplicação direta na programação, dessa maneira, para respeitar o primeiro pilar da orientação a objetos, a abstração, abstrai-se os resistores da classe dos botões.

Figura 66 - Diagrama de classes UML: botões de navegação e seleção

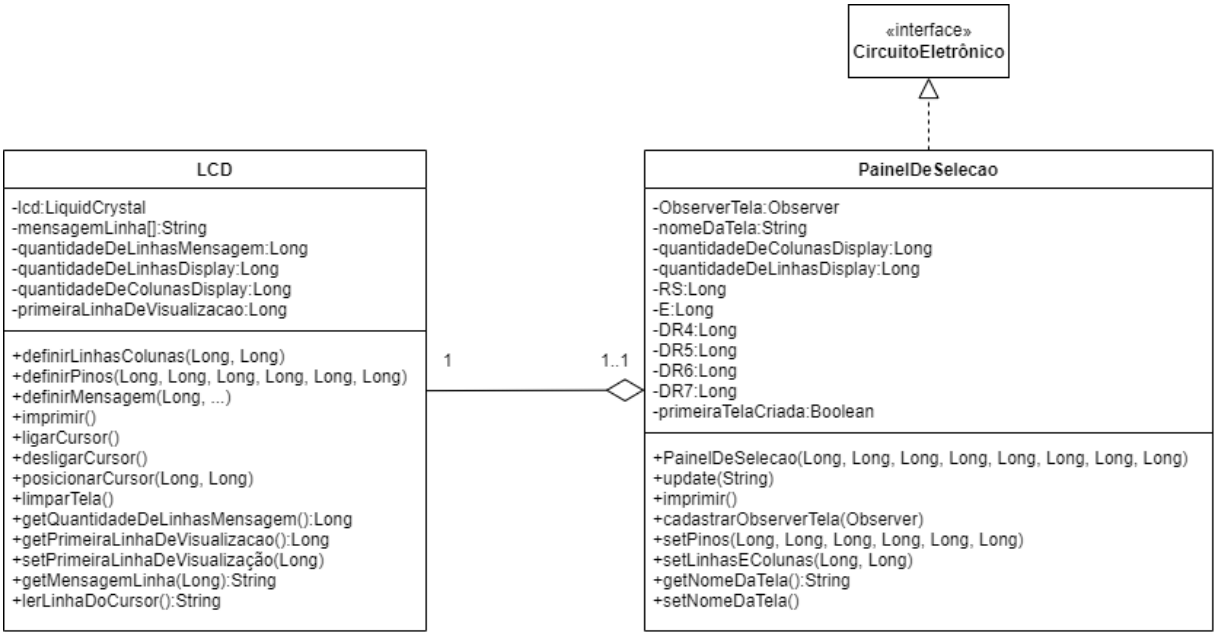


Fonte: Autores (2022)

2.10.2.3. Painel de Seleção

No contexto programação, a classe referente ao painel de seleção é simplesmente um display LCD, mas com outros métodos de aplicação mais específica para o projeto, separação realizada novamente para respeitar o conceito de abstração.

Figura 67 - Diagrama de classes UML: painel de seleção

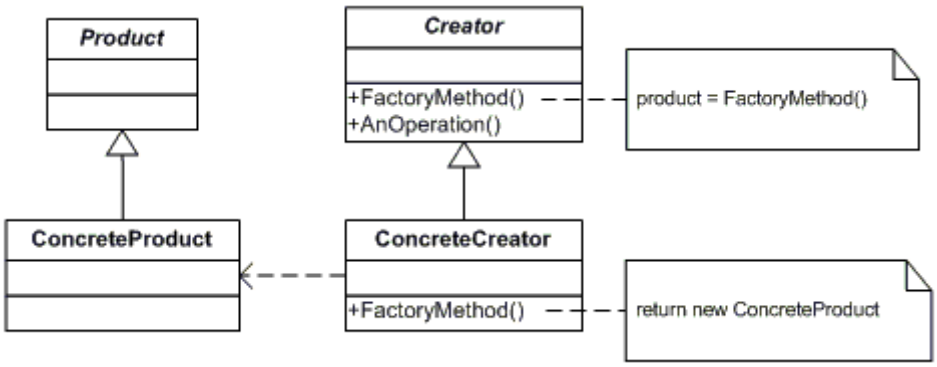


Fonte: Autores (2022)

2.10.2.3.1. Criação de Novas Telas

A criação de telas é realizada através de do padrão de projeto *factory*. Cria-se uma interface que dita os métodos criação de objetos em uma nova superclasse, entretanto, as subclasses possuem a capacidade de alterarem o tipo de objeto que será criado.

Figura 68 - Diagrama de classes UML: padrão de projeto *factory*

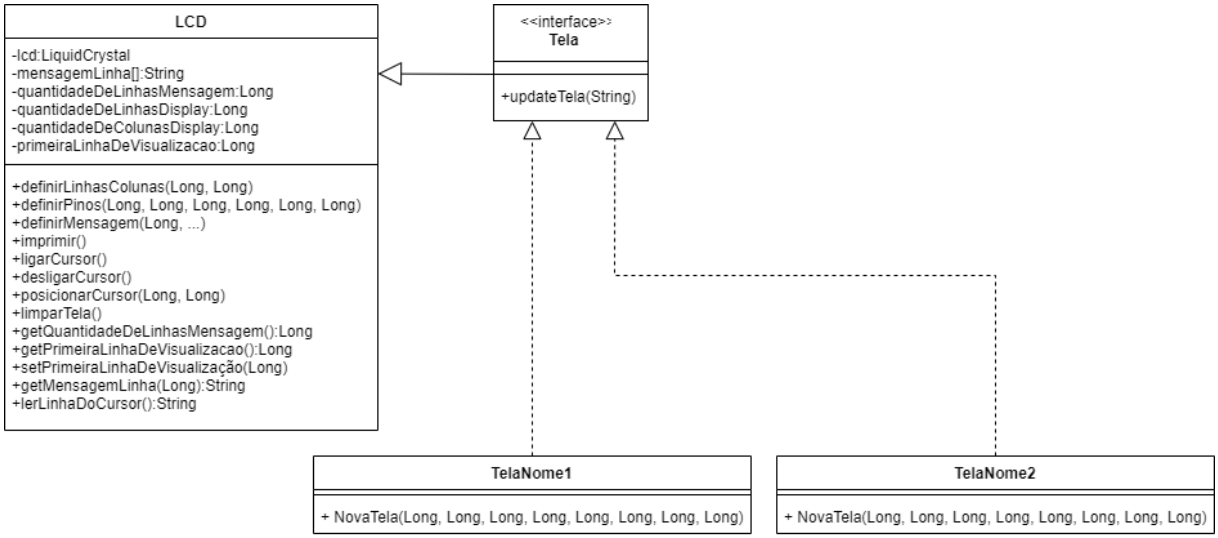


Fonte: (dofactory)

No projeto utiliza-se o padrão de projeto para a criação de novas telas. Inicialmente, cria-se uma interface que herda o componente eletrônico LCD e denote os métodos necessários para criação de uma nova tela, sendo eles “imprimir()” e “updateTela(mensagem)” cujas funções são indicar como a tela será impressa no

painel e a ação mediante a algum evento escrito por “mensagem”, respectivamente, além do método construtor o qual define-se os pinos de ligação e a mensagem.

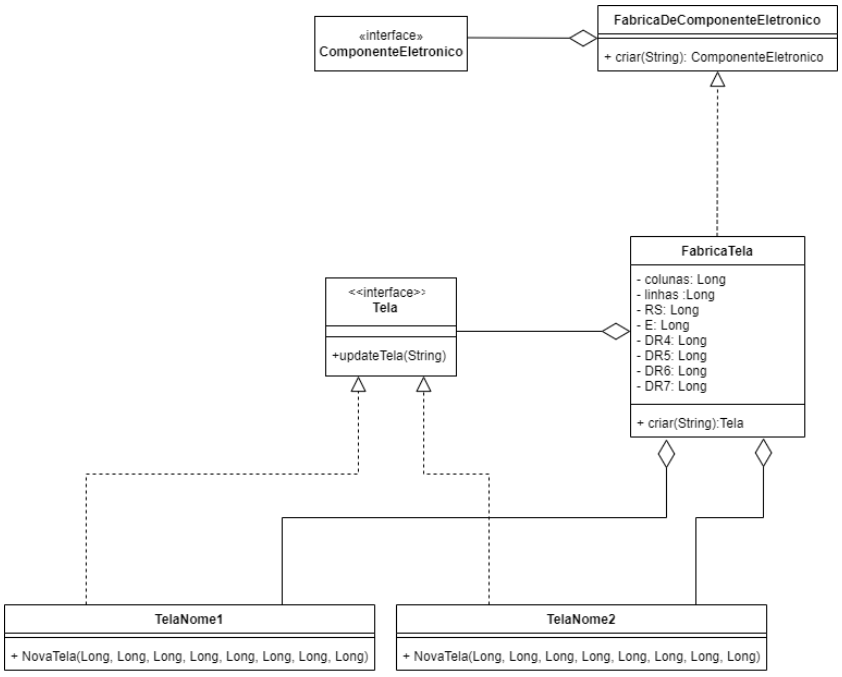
Figura 69 - Diagrama de classes UML: criação de novas telas



Fonte: Autores (2022)

Após, define-se outra interface nomeada “FabricaComponentesEletronicos” responsável por especificar o método de criação de objetos realizando a agregação da interface “ComponenteEletronico” e implementa-a a classe “FabricaTela”, sendo essa quem escreve a lógica do método de criação. A interface “Tela” e todas as telas definidas devem ser agregadas a essa classe.

Figura 70 - Diagrama UML: classe criadora de telas

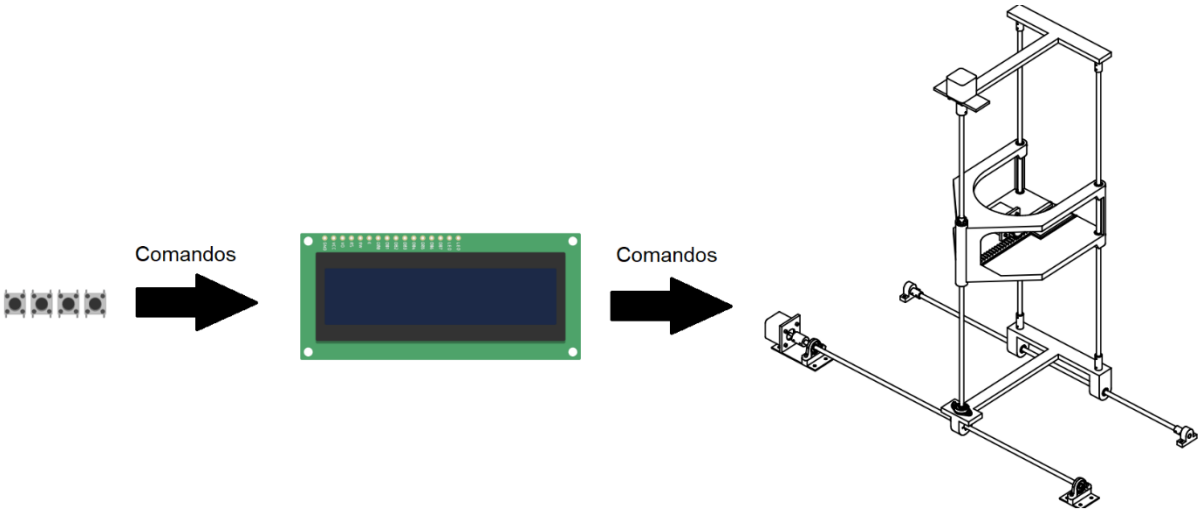


Fonte: Autores (2022)

2.10.3. Interação entre os Circuitos

A Figura 71 ilustra como deve ocorrer a interação entre os circuitos eletrônicos presentes no projeto. Os botões de seleção enviam comandos ao painel de seleção e esse, por sua vez, envia comandos ao circuito do almoxarifado.

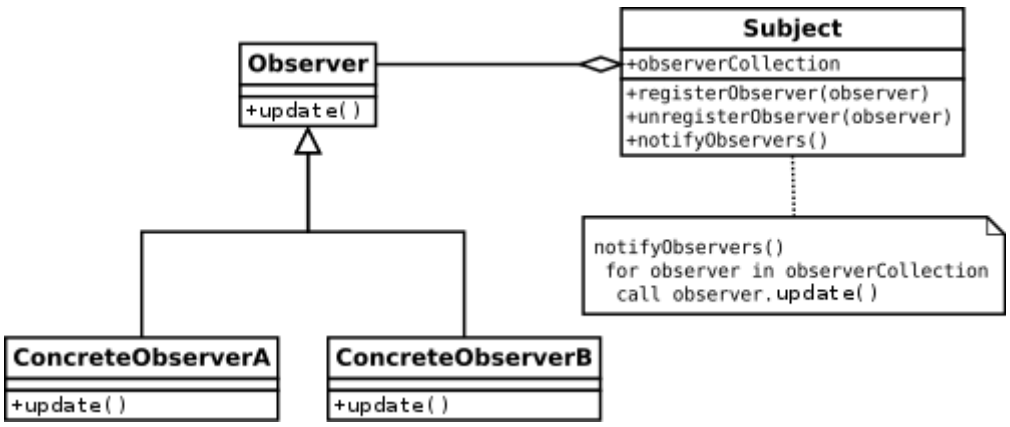
Figura 71 - Representação gráfica da interação entre os circuitos eletrônicos



Fonte: Autores (2022)

Um padrão de projeto amplamente utilizado nas áreas de desenvolvimento de software é o *observer*. Nele uma classe nomeada *"subject"* contém dentre seus atributos uma lista de classes contendo os *"observers*" os quais são notificados quando desejado, chamando por um de seus métodos.

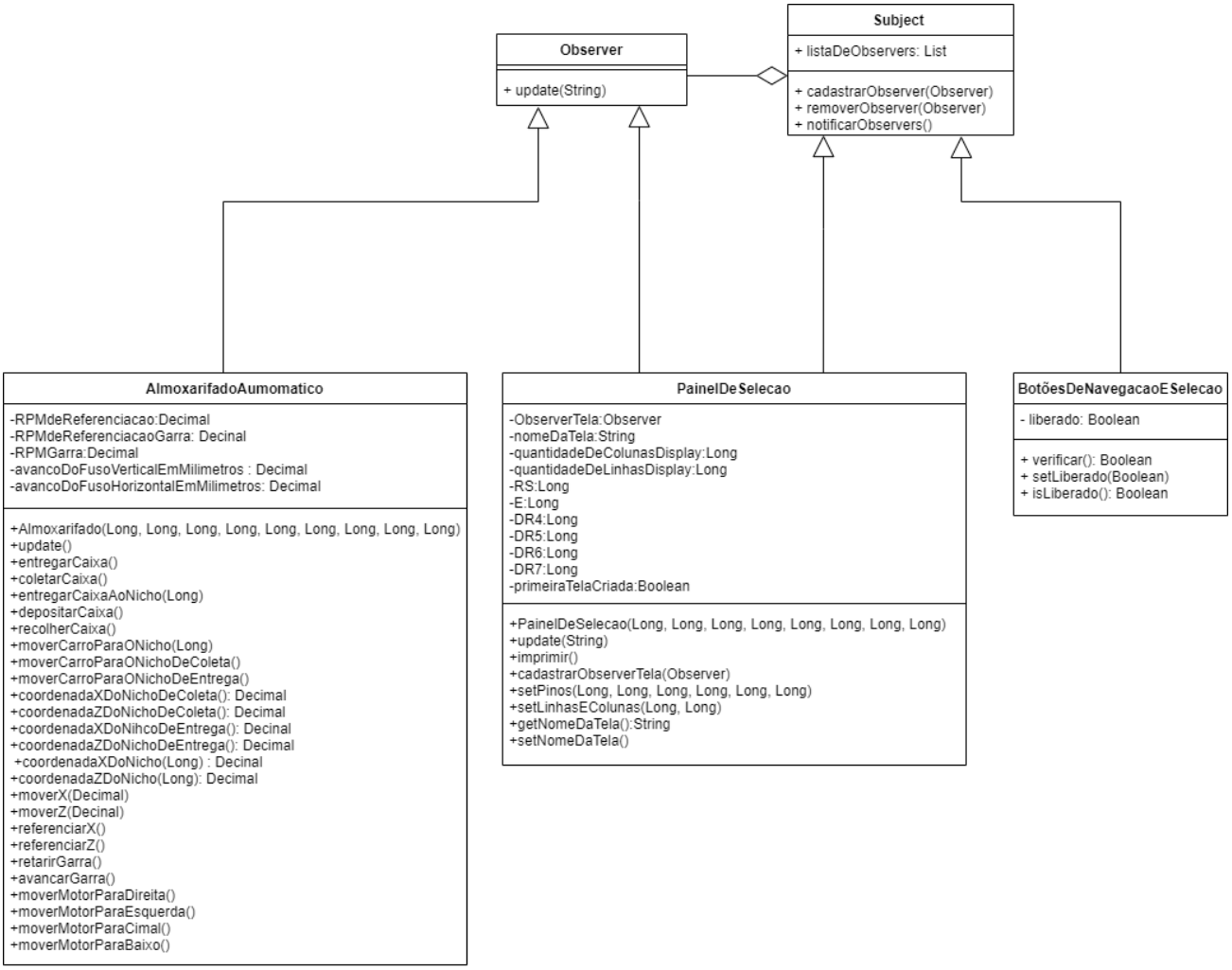
Figura 72 - Diagrama de classes UML: padrão de projeto *observer*



Fonte: (Tyagi, 2019)

Dessa maneira se dá a comunicação entre os três circuitos elétricos. A todo momento verifica-se o estado dos botões de navegação e seleção. Uma vez pressionados, notifica-se o painel de seleção o qual realiza a troca de tela ou, no caso de uma operação eletromecânica, a notificação do almoxarifado.

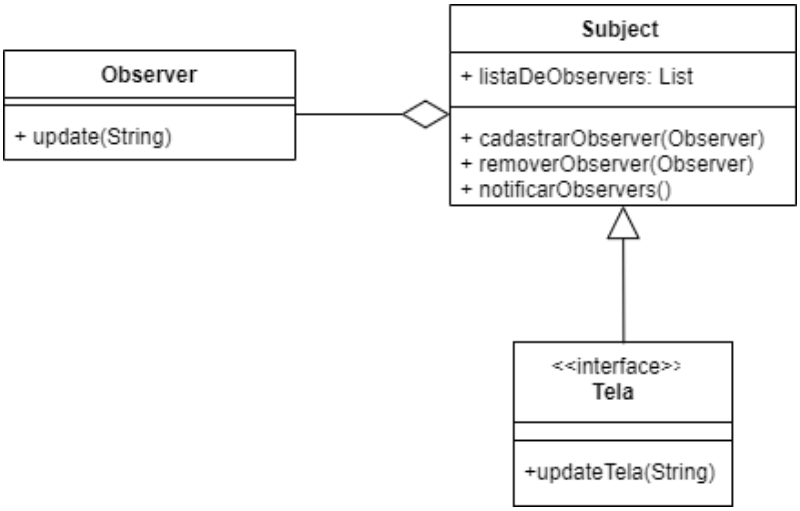
Figura 73 - Diagrama de classes UML: interação entre os circuitos eletrônicos



Fonte: Autores (2022)

Além do mais, a interface tela também deverá ter a classe *“subject”* como pai.

Figura 74 - Diagrama de classes UML: relação entre tela e *subject*

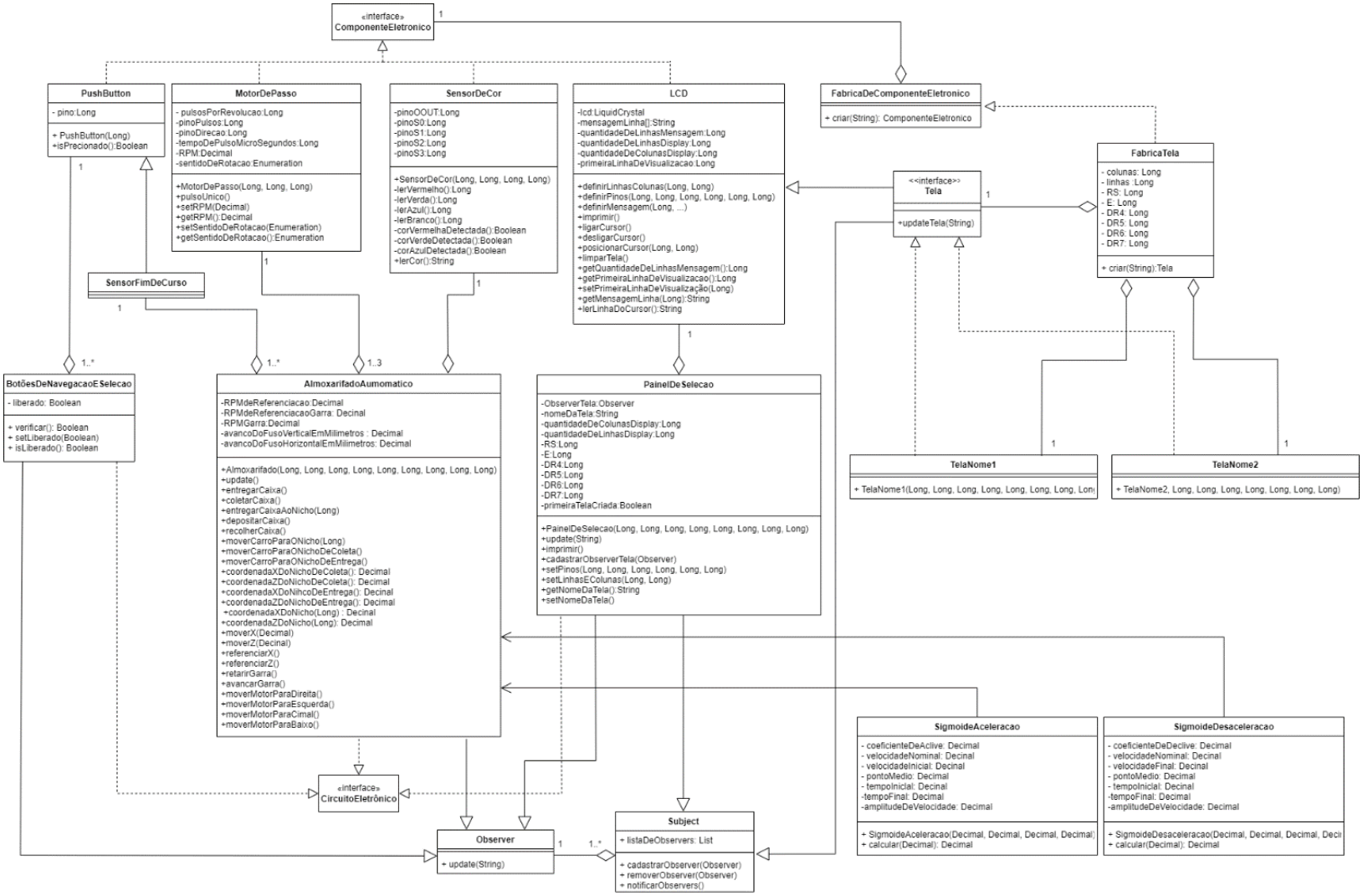


Fonte: Autores (2022)

2.10.4. Diagrama UML Completo

Relaciona-se todos os diagramas de classes UML referentes a programação, de modo a criar o diagrama de classes UML completo do projeto cujo resultado é descrito pela Figura 75.

Figura 75 - Diagrama de classes UML: final



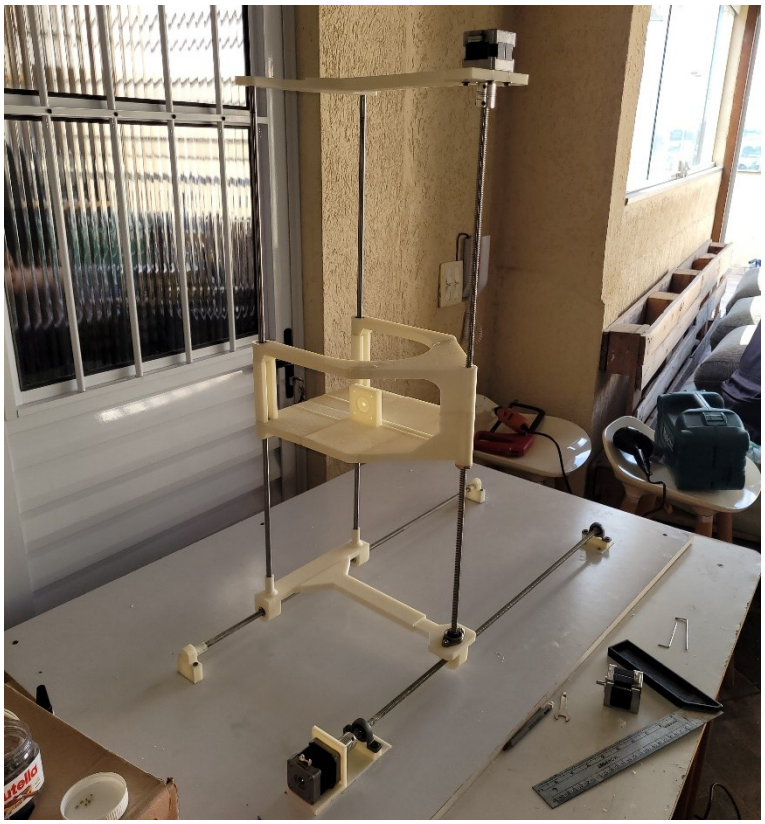
Fonte: Autores (2022)

2.11. Prototipagem

Para o teste final de toda concepção do projeto realiza-se a produção de um protótipo para aplicação dos conceitos teóricos e a correção dos possíveis erros não previstos na teoria.

A compra dos materiais especificados na subseção 2.2 foi realizada. Com os materiais em mãos, foi-se então colocadas as 8 peças estruturais para o processo de impressão 3D, procedimento concluído após duas semanas devido as grandes dimensões das peças.

Figura 76- Montagem do conjunto



Fonte: Autores (2022)

Posteriormente encontra-se o local onde os suportes horizontais seriam posicionados. A maior dificuldade foi o alinhamento, uma vez que modo que eles deveriam permanecer paralelos.

Coloca-se os suportes sem torque nos parafusos de fixação, de modo a realizar o alinhamento. Quando alinhado, realiza-se o procedimento de fixação dos suportes.

Figura 77- Alinhamento do conjunto

Fonte: Autores (2022)

Figura 78- Fixação do mecanismo a base

Fonte: Autores (2022)

Figura 79- Fixação final conforme e torque

Fonte: Autores (2022)

Após realizado o alinhamento, faz-se necessário a fixação dos motores. No caso do motor horizontal e do motor vertical utiliza-se das juntas elásticas para a fixação desses aos fusos do modo que o alinhamento entre os componentes seja perfeito. Vale ressaltar que deve haver uma distância de segurança entre o eixo motor e o eixo movido com fim de prevenir a quebra da algum deles.

Figura 80- Conexão dos motores

Fonte: Autores (2022)

Define-se que todos os fios desceriam pelo meio do carro vertical e iriam de encontro à direita da base.

Figura 81- Concepção das passagens de fiação

Fonte: Autores (2022)

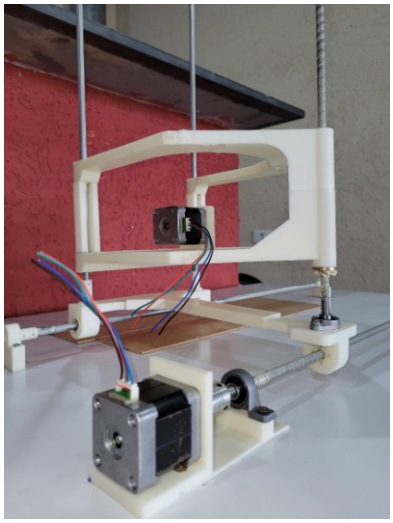
Fez-se necessário modificações no suporte do motor de passo da garra, pois ele não comportaria o encaixe da engrenagem, uma vez que ele não comportaria o encaixe da engrenagem e a fixação do motor.

Figura 82- Conexão do motor da garra

Fonte: Autores (2022)

Realiza-se o engraxamento das peças para evitar a corrosão das peças e evitar os desgastes pelo contato entre os materiais.

Figura 83- Engraxamento das peças



Fonte: Autores (2022)

Definidos a posição do conjunto e da fiação foi necessário estudar qual seria a melhor distância entre a estante e o carro, estabelecida em 10 mm entre ambas as peças. Através disso consegue-se uma locomoção segura para a caixa e a garra em caso de erro.

Figura 84- Definição da posição da estante

Fonte: Autores (2022)

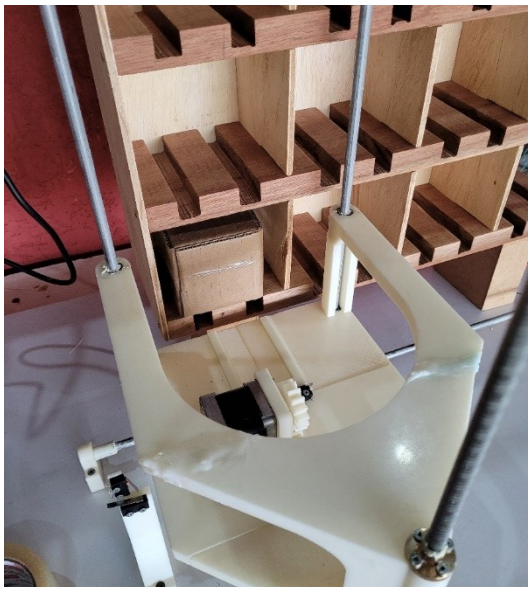
Com finalidade de manter os componentes eletrônicos em segurança, monta-se uma tela de segurança simples com MDF. Aproveita-se dele para a fixação da interface homem-máquina.

Figura 85- Confecção da célula eletroeletrônica de controle

Fonte: Autores (2022)

Realiza-se a confecção de um suporte simples para alojar dois sensores de fim de curso que realizavam a função de referenciação.

Figura 86- Definição do sensor de fim de curso



Fonte: Autores (2022)

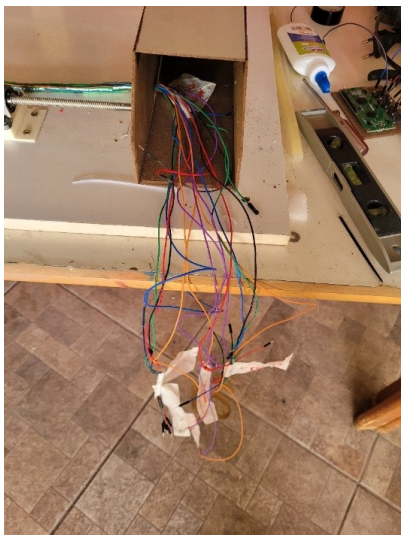
Confecciona-se caixas de papelão coloridas para o reconhecido deles pelo sensor de cor instalado no nicho de coleta da máquina.

Figura 87- Confecção das caixas e posição do sensor de cor

Fonte: Autores (2022)

Realiza-se soldas nos plugs e testes de continuidade em cada um dos cabos os quais foram encaminhados para a célula onde foi feito as devidas ligações aos componentes eletrônicos.

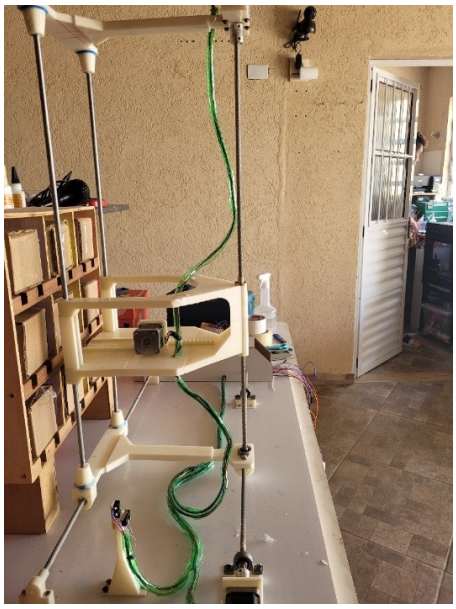
Figura 88- Processo de encabiamento até a célula



Fonte: Autores (2022)

Para que não houvesse problemas de cortes e escapadas dos cabos, eles foram enrolados em garrafas PETs reutilizadas melhorando a organização e a prevenção de futuros problemas.

Figura 89- Organização dos cabos elétricos



Fonte: Autores (2022)

Realiza-se os testes da programação. Houve problemas que foram superados e adaptados com fim de garantir o funcionamento pleno do projeto.

Figura 90- Montagem e teste do programa

Fonte: Autores (2022)

3. CONCLUSÃO

A junção entre mecânica, eletrônica e programação possibilitou a criação de um processo de automação através de um equipamento mecatrônico o qual foi exposto neste projeto. Ao utilizarmos motores de passo, juntas e fusos a área mecânica faz sua parte ao gerar movimento e transmiti-lo de forma rápida e eficiente. Já a eletrônica, juntamente da programação trouxe a parte lógica e controlável do processo ao permitir que todos os componentes como sensores e drivers tenham uma interação completa com as partes mecânicas, possibilitando que todo o projeto consiga atender e cumprir a tarefa de manuseio de cargas.

A interação entre essas áreas forma a mecatrônica, a qual possbilitou a o trabalho inteligente com finalidade de atender os objetivos traçados.

Houve várias dificuldades no decorrer desses últimos dois semestres, mas conseguimos ultrapassar todas elas, em meio a várias alterações tanto nas estruturas mecânicas, nos projetos elétricos e na programação. Uma longa linha de aprendizagem foi traçada graças a ajuda dos professores e de várias pesquisas.

Com essa experiência nos foi mostrado o verdadeiro significado de trabalho em equipe e o desenvolvimento de nossa capacidade como técnicos. Assim esperando também ter atingido as expectativas dos professores que nos apoiaram na execução deste projeto. Com diferentes pensamentos, mas com um único propósito de alcançar o resultado esperado. Por mais que esse trabalho seja demorado e precise de muita resiliência, é algo extremamente necessário para nossa formação como técnicos. Por motivos financeiros não se foi possível realizar a produção de esteiras nos nichos de retirada e saída das caixas, com o implemento das esteiras todo o sistema se tornaria mais inteligente e produtivo pela rapidez com que se conseguiria transportar retirando ou colocando o elemento em armazenamento.

REFERÊNCIAS

Acelera 3D. (2022). *acelera3d.com*. Acesso em 04 de 12 de 2022, disponível em Acelera 3D: https://acelera3d.com/produto/rolamento-lm8uu/

Aço Rag. (2022). *www.acorag.com.br/*. Acesso em 04 de 12 de 2022, disponível em Aço Rag: https://www.acorag.com.br/barra-retificada

Applied Motion. (2022). *www.applied-motion.com*. Acesso em 12 de 04 de 2022, disponível em Applied Motion: https://www.applied-motion.com/products/stepper-motors/ht17-268

Autodesk. (2022). *www.autodesk.com.br*. Acesso em 04 de 12 de 2022, disponível em Autodesk: https://www.autodesk.com.br/products/fusion-360/overview?mktvar002=5555270|SEM|19028026506|143497597403|kwd-45199965798&utm_source=GGL&utm_medium=SEM&utm_campaign=GGL_D-M_Fusion-360_AMER_BR_eComm_SEM_BR_New_EX_ADSK_5555270_General&utm_id=5555270&utm_term=

Barth, S. (2020). *www.escolamultiverso.com.br*. Acesso em 29 de 09 de 2022, disponível em Escola Multiverso: https://www.escolamultiverso.com.br/post/fatores-de-risco-ergonomicos-no-transporte-manual-de-cargas

Baú da Eletronica. (2022). *www.baudaeletronica.com.br*. Acesso em 04 de 12 de 2022, disponível em Baú da Eletronica: https://www.baudaeletronica.com.br/chave-micro-switch-kw-11-3z-5a-3-terminais-1.html

Confea. (19 de 09 de 2019). *www.confea.org.br*. Acesso em 22 de 09 de 2022, disponível em CONFEA: https://www.confea.org.br/sites/default/files/uploads-imce/Contecc2019/Experi%C3%AAncia%20Profissional/ANALISE%20E%20IDENTIFICA%C3%87%C3%83O%20DOS%20RISCOS%20PRESENTES%20NO%20ALMOXARIFADO%20DA%20UFAP-MEDIDAS%20CORRETIVAS-PROTETIVAS.pdf

dofactory. (s.d.). *www.dofactory.com*. Acesso em 20 de 11 de 2022, disponível em dofactory: https://www.dofactory.com/net/factory-method-design-pattern

Eletrônica Faria. (2022). *www.eletronicafaria.com.br*. Acesso em 04 de 12 de 2022, disponível em Eletrônica Faria: https://www.eletronicafaria.com.br/inicio/399-modulo-sensor-de-cor-tcs3200-arduino.html

Fetropar. (2022). *fetropar.org.br*. Acesso em 29 de 08 de 2022, disponível em Fetropar: https://fetropar.org.br/movimentacao-de-pesos-inadequada-pode-causar-danos-irreparaveis-aos-trabalhadores/#:~:text=A%20m%C3%A1%20condu%C3%A7%C3%A3o%20de%20pesos,doen%C3%A7as%20localizadas%20no%20aparelho%20circulat%C3%B3rio.&text=Ao%20levantar%20a%20carga

Filipe Flop. (2022). *www.filipeflop.com*. Acesso em 04 de 12 de 2022, disponível em Filipe Flop: https://www.filipeflop.com/produto/filamento-abs-branco/

Filipe Flop. (2022). *www.filipeflop.com*. Acesso em 04 de 12 de 2022, disponível em Filipe Flop: https://www.filipeflop.com/produto/impressora-3d-creality-ender-3/

Filipe Flop. (2022). *www.filipeflop.com*. Acesso em 04 de 12 de 2022, disponível em Filipe Flop: https://www.filipeflop.com/produto/placa-mega-2560-r3-cabo-usb-para-arduino/

Galdino, L. (19 de 10 de 2014). *pt.slideshare.net*. Acesso em 04 de 09 de 2022, disponível em SlidesShare: https://pt.slideshare.net/lucianogaldino/dimensionamento-motor-fuso

Khan Academy. (s.d.). *pt.khanacademy.org*. Acesso em 2022 de 06 de 24, disponível em Khan Academy: https://pt.khanacademy.org/science/physics/torque-angular-momentum/torque-tutorial/a/torque#:~:text=Torque%20%C3%A9%20uma%20medida%20de,sentido%20da%20for%C3%A7a%20no%20eixo

Loja 3D. (2022). *www.loja3d.com.br*. Acesso em 04 de 12 de 2022, disponível em Loja 3D: https://www.loja3d.com.br/escola-3d/curso-de-fatiamento-3d-ultimaker-cura-online

Loja da Robótica. (2022). *www.lojadarobotica.com.br*. Acesso em 04 de 12 de 2022, disponível em Loja da Robótica: https://www.lojadarobotica.com.br/mancal-kp08-para-eixo-8mm-com-rolamento

Lopes, B., Reis, A., & Bissol, L. (2021). *Brazilian Journal of Development*. Acesso em 2022 de 11 de 19, disponível em ojs.brazilianjournals.com.br: https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/25950

Maganize Luiza. (2022). *www.magazineluiza.com.br*. Acesso em 04 de 12 de 2022, disponível em Maganize Luiza: https://www.magazineluiza.com.br/kfl08-mancal-com-rolamento-auto-compensador-interno-8mm-3d4all/p/ggdg814b8j/au/mncl/

Mattede, H. (s.d.). *www.mundodaeletrica.com.br*. Acesso em 24 de 06 de 2022, disponível em Mundo da Elétrica: https://www.mundodaeletrica.com.br/o-que-e-motor-de-passo-funcionamento-aplicacoes/#:~:text=O%20motor%20de%20passo%20%C3%A9,sem%20a%20necessidade%20de%20escovas.

Mendes, D. (2013). *temseguranca.com*. Acesso em 13 de 08 de 2022, disponível em Tem Segurança: https://temseguranca.com/investigacao-de-acidente-com-empilhadeira/

Microwat. (2022). *www.microwat.com.br/*. Acesso em 04 de 12 de 2022, disponível em Microwat: https://www.microwat.com.br/arduino-prototipagem/modulos-e-componentes/flange-tr8-passo-8mm

Ministério do Trabalho e Previdência. (22 de 10 de 2020). *www.gov.br*. Acesso em 05 de 09 de 2022, disponível em gov.br: https://www.gov.br/trabalho-e-previdencia/pt-br/composicao/orgaos-especificos/secretaria-de-trabalho/inspecao/seguranca-e-saude-no-trabalho/ctpp-nrs/normas-regulamentadoras-nrs

MP MULTIPEÇAS. (2022). *www.multipecas.curitiba.br*. Acesso em 04 de 12 de 2022, disponível em MP MULTIPEÇAS: https://www.multipecas.curitiba.br/loja/produto/acoplamento-flexivel-8-x-8-mm

Portaria MTP. (07 de 10 de 2021). *gov.br*. Acesso em 2022 de 09 de 2022, disponível em www.gov.br: https://www.gov.br/trabalho-e-previdencia/pt-br/composicao/orgaos-especificos/secretaria-de-trabalho/inspecao/seguranca-e-saude-no-trabalho/normas-regulamentadoras/nr-17-atualizada-2021.pdf

Shigley, J., & Mischke, C. (2013). *Mechanical Engineering Design.* McGraw-Hill Science.

Silva, S. (2007). *www.ipef.br*. Acesso em 16 de 04 de 2022, disponível em IPEF: https://www.ipef.br/publicacoes/scientia/nr74/cap02.pdf

Souza, F. (2013). *embarcados.com.br*. Acesso em 02 de 04 de 2022, disponível em Embarcados: https://embarcados.com.br/arduino-mega-2560/

Teja, R. (20 de 01 de 2021). */www.electronicshub.org*. Acesso em 19 de 11 de 2022, disponível em Electronics Hub: https://www.electronicshub.org/arduino-mega-pinout/

THK Tech. (s.d.). *www.tech.thk.com*. Acesso em 24 de 06 de 2022, disponível em THK: https://tech.thk.com/pt/products/pdf/br_b15_069.pdf

Thomsen, A. (2013). *www.filipeflop.com*. Acesso em 24 de 06 de 2022, disponível em Filipe Flop: https://www.filipeflop.com/blog/controlando-um-motor-de-passo-5v-com-arduino/

TME. (2022). *www.tme.eu*. Acesso em 04 de 12 de 2022, disponível em TME: https://www.tme.eu/rs/en/details/pololu-2986/motor-control-modules/pololu/a4988-stepper-motor-driver-carrier-black/

Tyagi, A. (29 de 03 de 2019). *www.c-sharpcorner.com*. Acesso em 19 de 11 de 2022, disponível em C# Corner: https://www.c-sharpcorner.com/article/observer-design-pattern-using-c-sharp/

Unesp. (11 de 03 de 2013). *www.feis.unesp.com.br*. Acesso em 19 de 11 de 2022, disponível em Unesp: https://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/aula3-motor-de-passo-2013-1-13-03-2013-final.pdf

GLOSSÁRIO

ABS: Plástico acrilonitrila butadieno estireno.

AET: Análise Ergonômica do Trabalho.

CNC: Comando numérico computadorizado é um sistema que permite o controle automático de máquinas através do controle simultâneo de seus eixos.

C++: Linguagem de programação de uso geral.

E-commerce: Comércio eletrônico.

Export: Exportar ou transferir.

Extrude: Expandir.

Factory: Padrão de projeto utilizado para a criação de objetos através da intermediação de outro objeto.

Fetropar: Federação dos Trabalhadores em Transportes Rodoviários do Estado do Paraná.

G-Code: Linguagem de programação utilizada em máquinas CNC como impressoras 3D, centros de usinagem e centro de torneamento.

Input: Entrada.

LCD: Tela feita de cristal líquido capaz de exibir informações dependendo da instrução realizada pelo microcontrolador.

LED/LEDs: Diodo emissor de luz.

NIOSH National Institute for Occupational Safety and Health (Instituto Nacional de Segurança e Saúde Ocupacional).

NR: Normas Reguladoras.

PET Resina termoplástica da família do poliéster.

PLA Polímero termoplástico feito com ácido lático.

Pull-down: Resistor responsável por enviar 0 V ao microcontrolador quando a chave estiver aberta.

Pushbutton: Chave tátil.

RPM: Unidade de medida que indica a quantidade de rotações revoluções realizadas um minuto.

RPS: Unidade de medida que indica a quantidade de rotações revoluções realizadas um segundo.

Slicer: Fatiador.

Split: Repartir, dividir, separar.

.stl: Do inglês, Standard Triangle Language, é o modelo de arquivo utilizado para salvar desenhos em 3D.

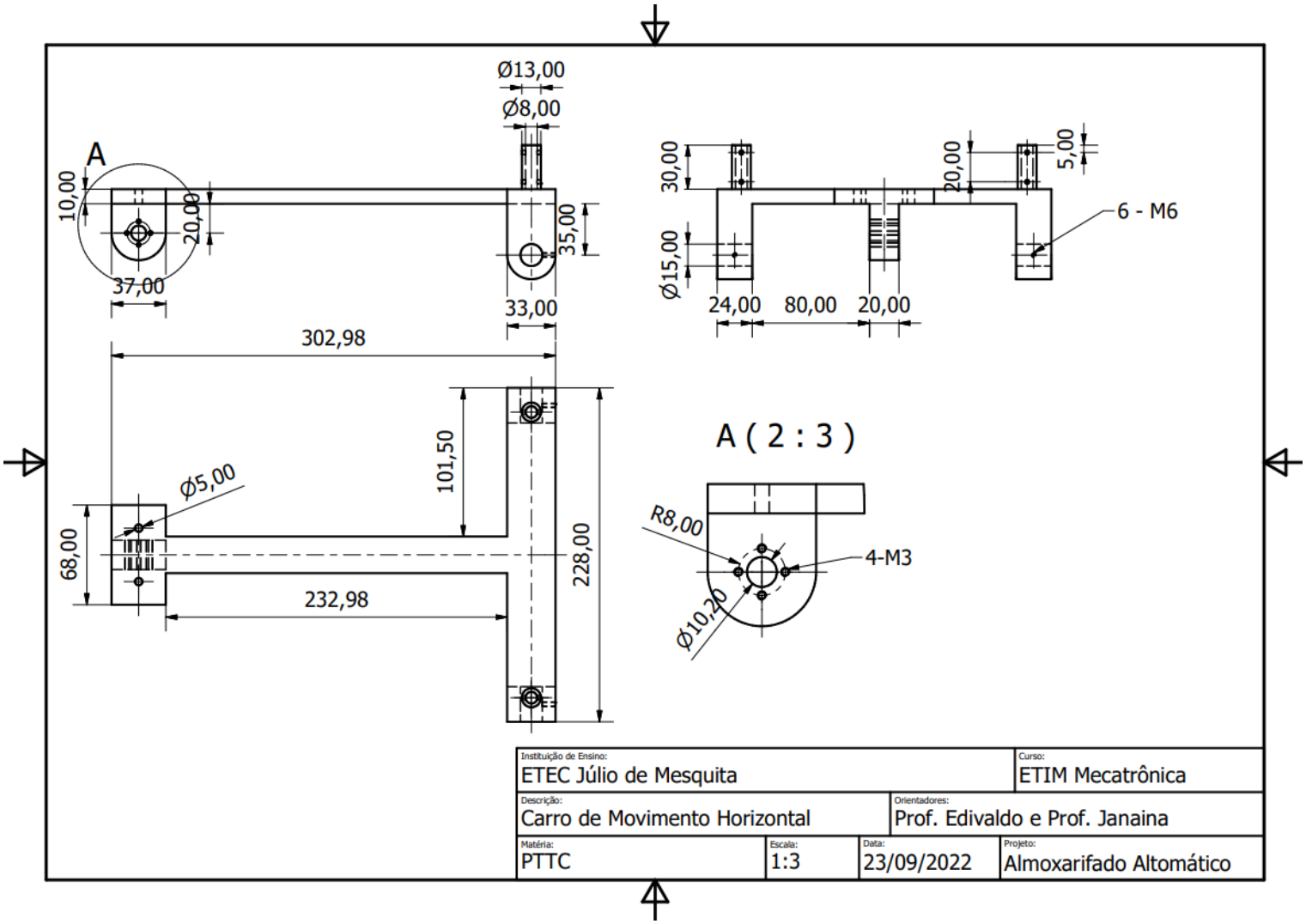
UML: Digrama utilizado para descrever o limite, a estrutura e o comportamento de sistema e objetos nele contido.

V: Unidade de medida utilizada descrever a diferença de potencial elétrico entre dois pontos.

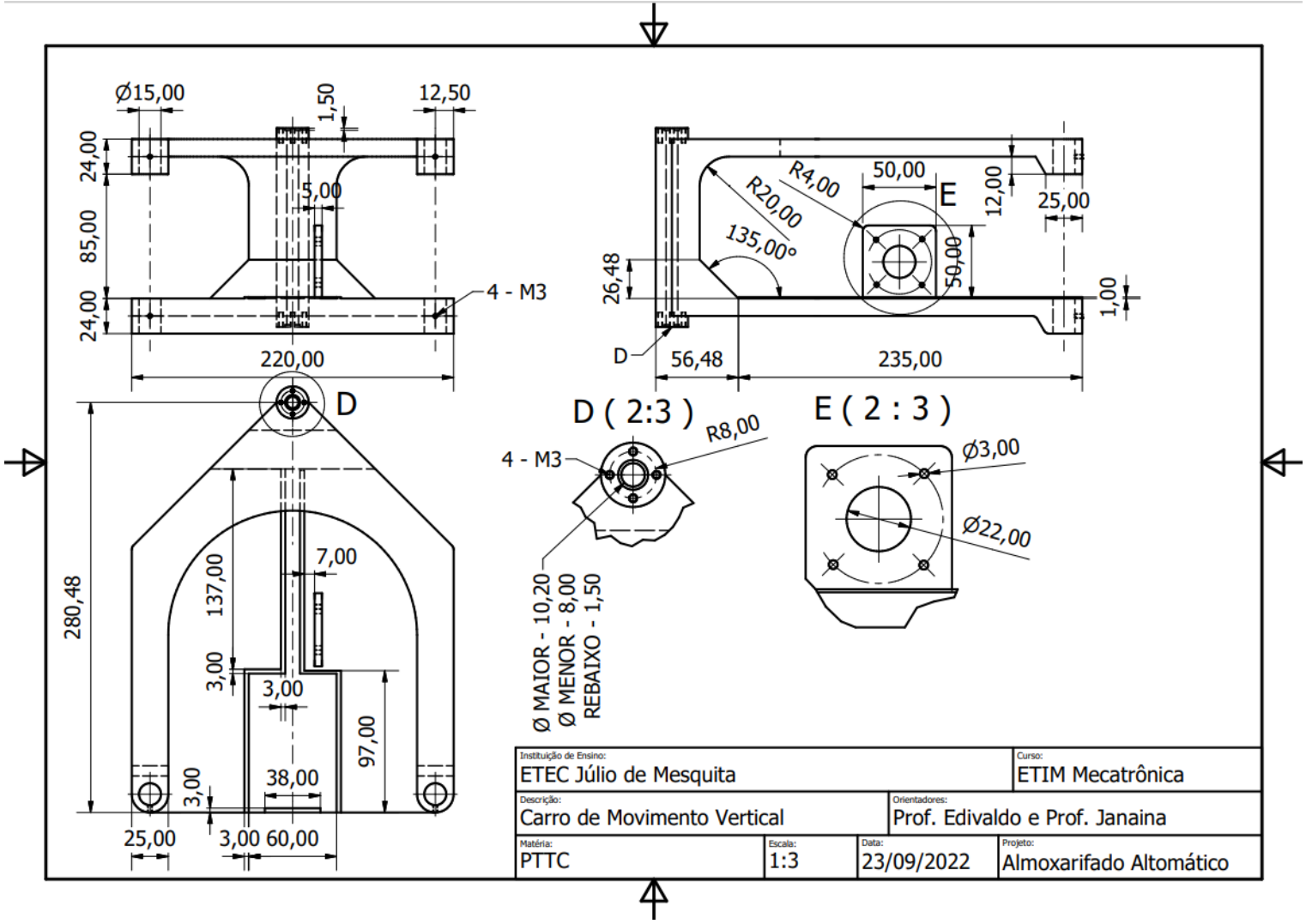
VCC: Tensão de corrente contínua.

W: Unidade de medida utilizada para calcular as capacidades na qual a energia elétrica é capaz de realizar trabalho.

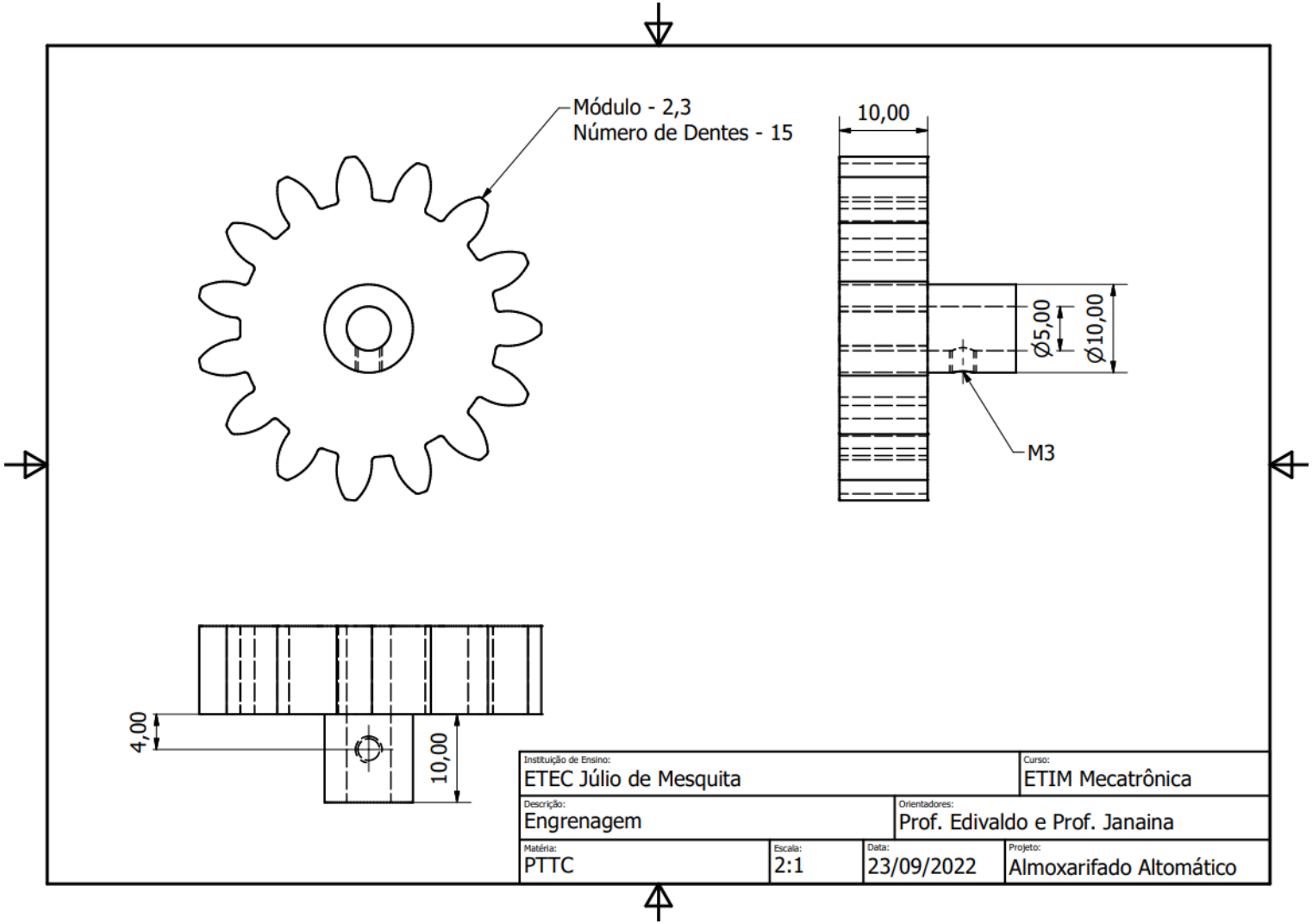
APÊNDICE A - Carro Horizontal



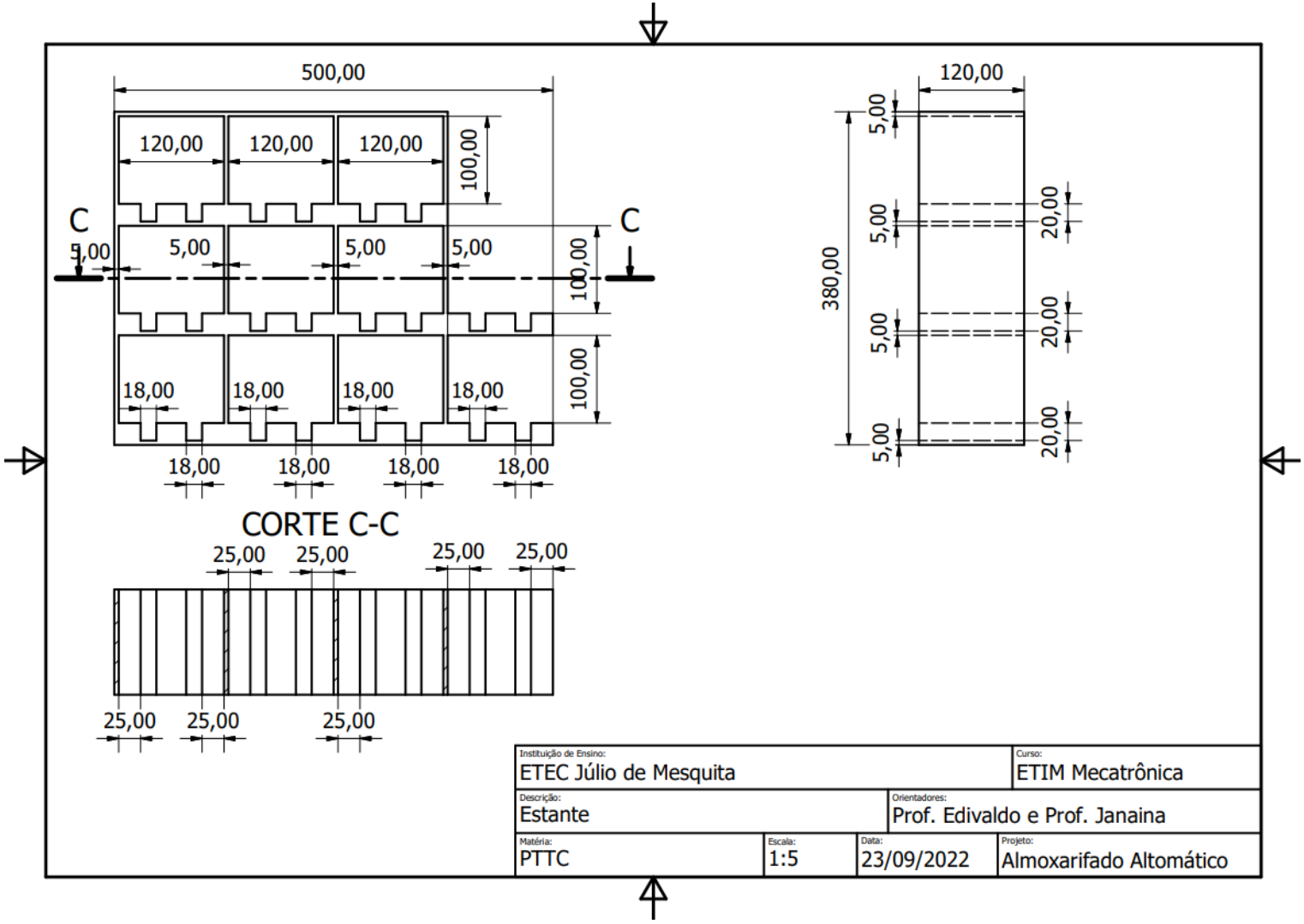
APÊNDICE B-Carro Vertical



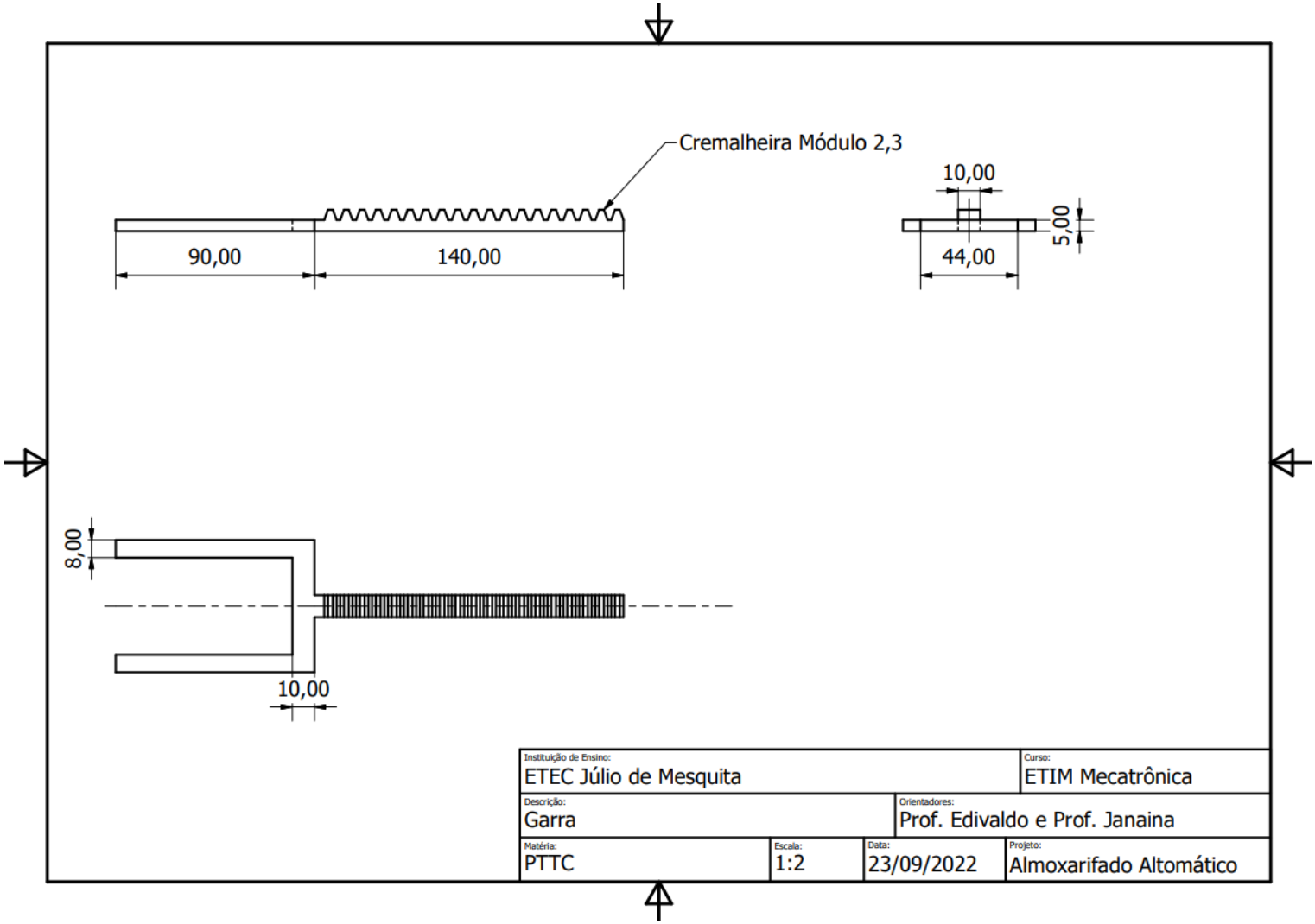
APÊNDICE C-Engrenagem



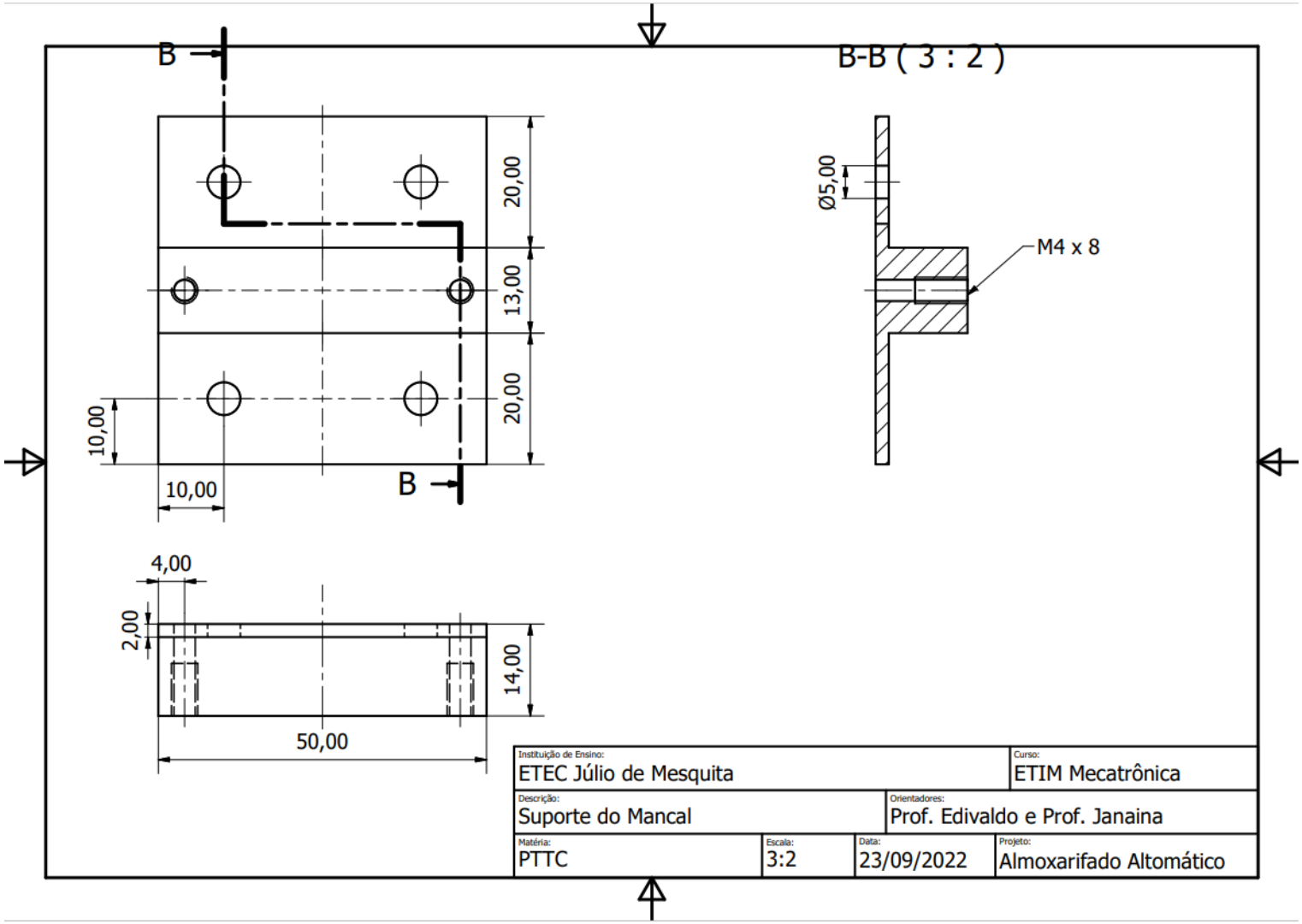
APÊNDICE D-Estante



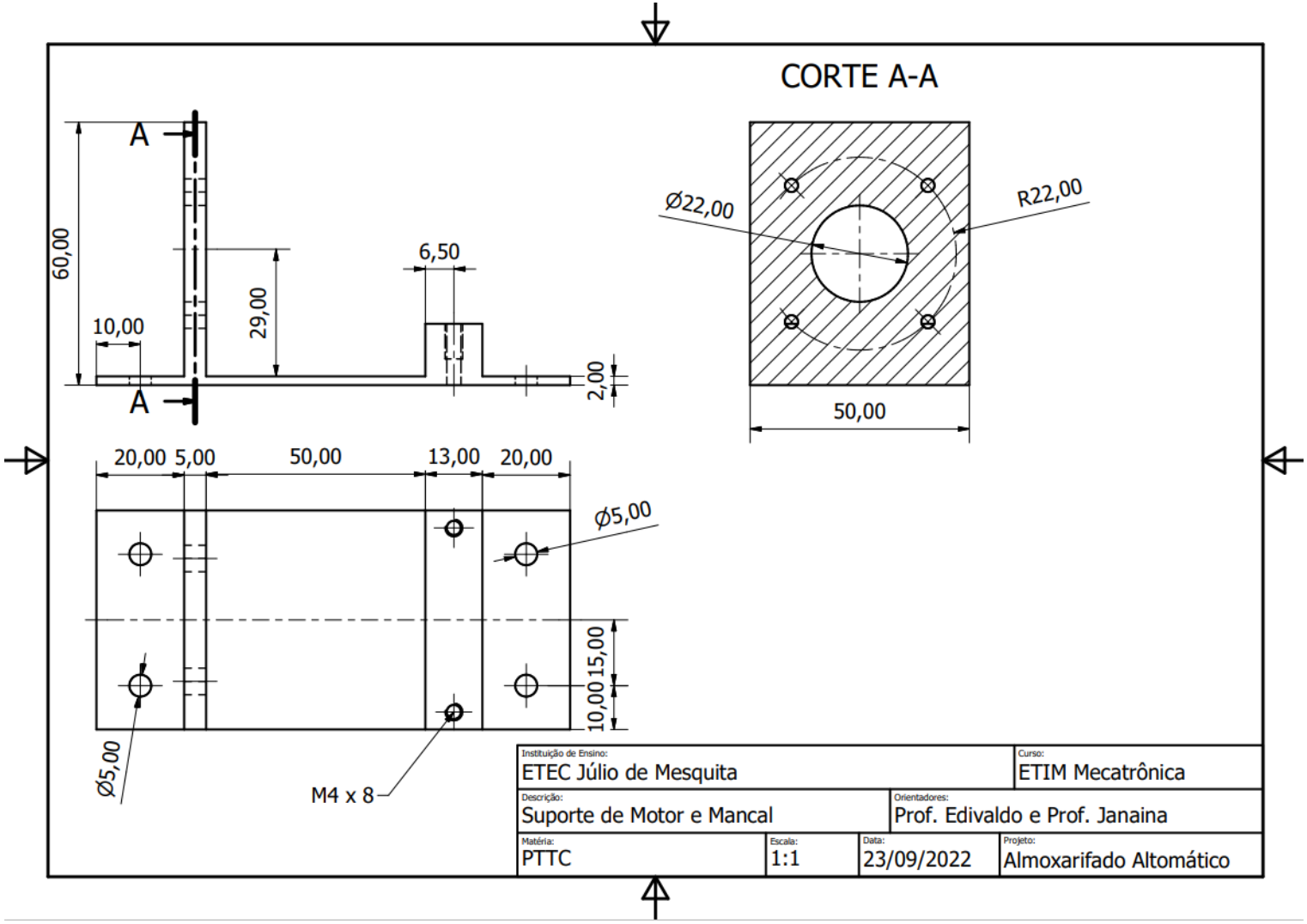
APÊNDICE E - Garra



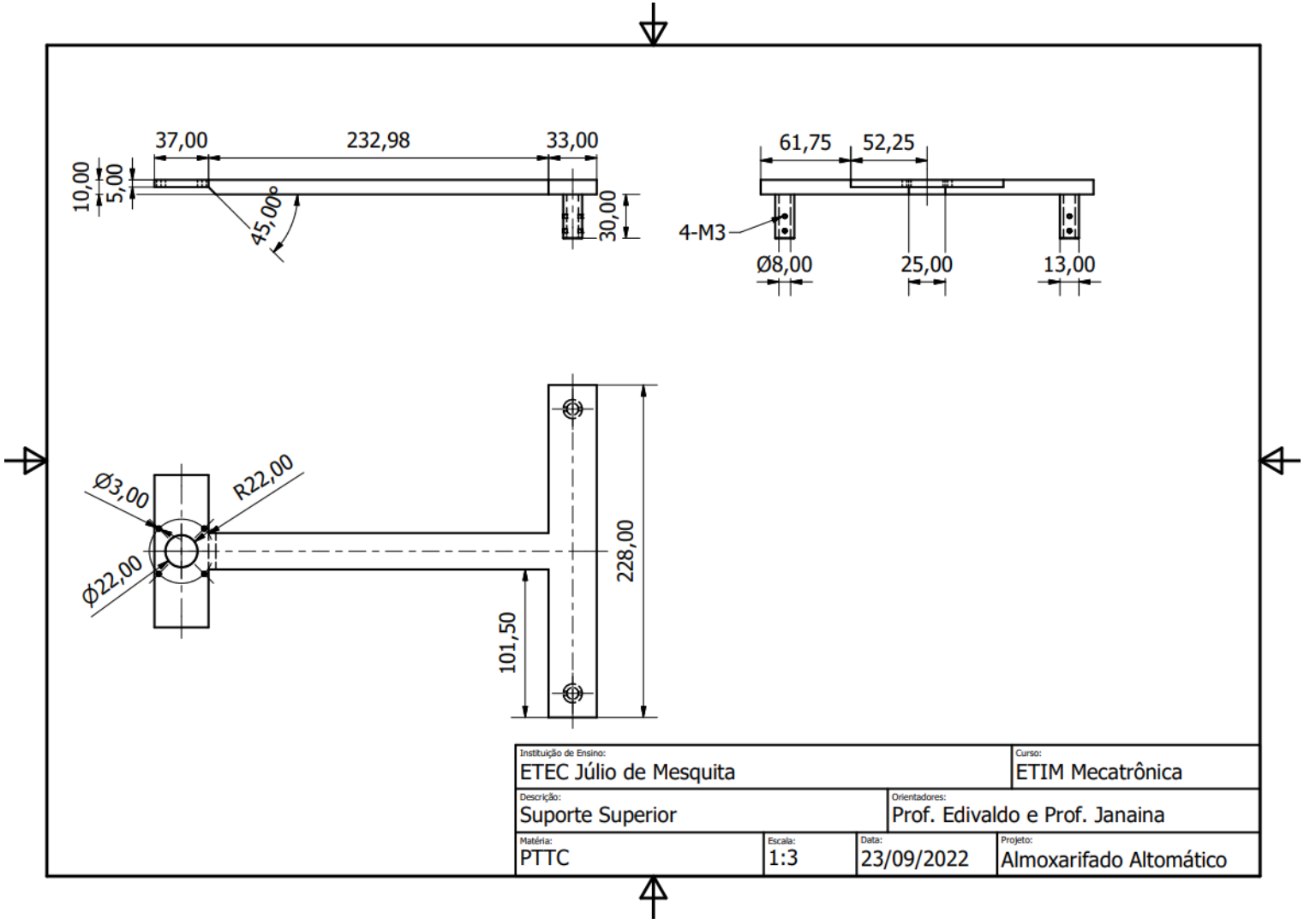
APÊNDICE F - Suporte do Mancal



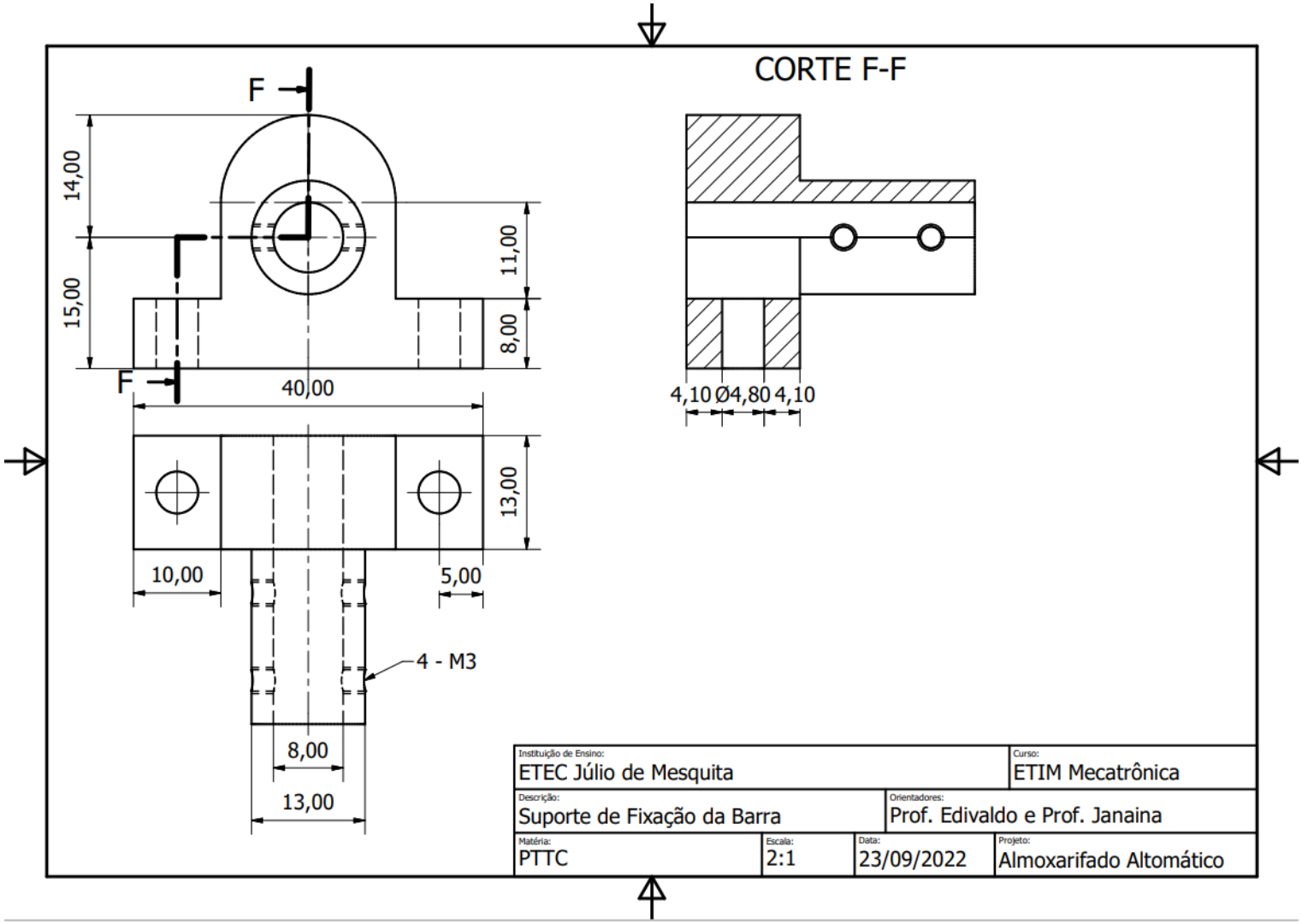
APÊNDICE G - Suporte do Motor

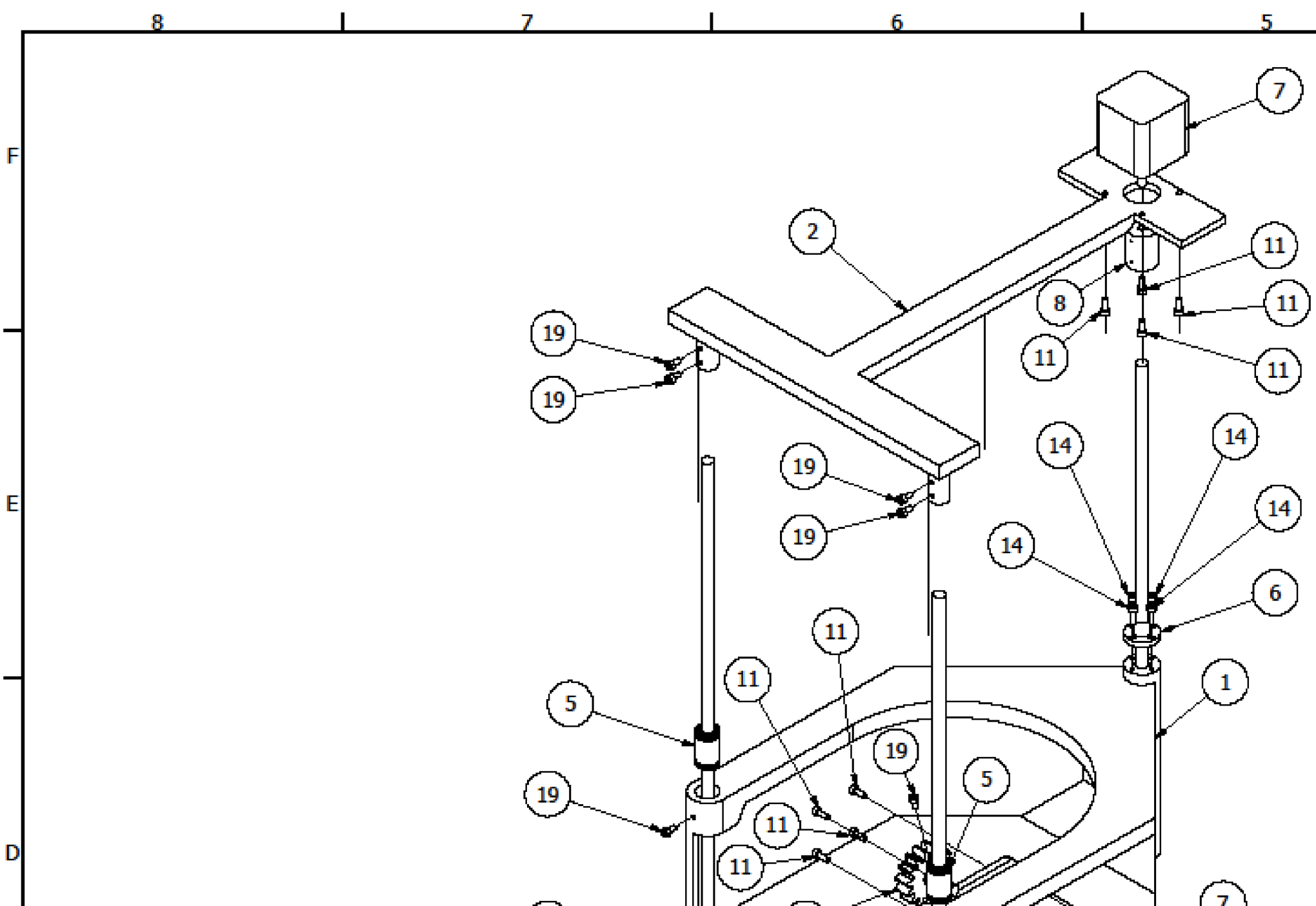


APÊNDICE H - Suporte Superior



APÊNDICE I-Suporte




8
7
6
5
F
E
D
7
2
11
19
19
8
11
11
19
19
14
14
14
14
6
11
11
5
1
19
19
5
11
11
7

APÊNDICE K-Estimativa de Custos

Tabela 5 - Estimativas de custo da prototipagem

Componentes	**Quantidade**	**Preço Unitário**	**Preço Total**
Arduino Mega	1	R$189,90	R$189,90
Motor Nema 17	3	R$83,32	R$249,96
Fuso retificado M8x700mm	2	R$71,91	R$153,75
Barra retificada 8x700Mmm	3	R$37,00	R$111,00
Acoplamento flexível 5x8mm	3	R$19,90	R$59,70
Pacote de jumpers	3	R$15,00	R$45,00
Sensor de cor TCS3200	1	R$45,00	R$45,00
Driver p/ motor de passo A4988	3	R$19,90	R$59,70
Mancal KP08	2	R$16,76	R$33,52
Mancal KFL08	1	R$17,93	R$17,93
Flange TR8	2	R$13,26	R$26,52
Rolamento linear LM8UU	8	R$5,12	R$41,00
Pushbutton	4	R$5,00	R$20,00
Filamento ABS	1	R$45,00	R$45,00
Placas de MDF	8	R$79,80	R$638,40
Valor “homem-hora”	10 horas	R$150,00	R$1.500,00
Total			R$3.195,11

Fonte: Autores (2022)

APÊNDICE L – Programação Integral

Arquivo TCC-PROG.ino

```
#include "./Classes/CircuitosEletronicos/PainelDeSelecao.hpp"
#include "./Classes/CircuitosEletronicos/AlmoxarifadoAutomatico.hpp"
#include "./Classes/CircuitosEletronicos/BotoesSelecao.hpp"

AlmoxarifadoAutomatico *almoxarifado = new AlmoxarifadoAutomatico(3, 36, 4, 37, 5, 38, 28, 29, 30, 39, 35, 31, 32, 33, 34);
PainelSelecao *painel = new PainelSelecao(16, 2, 22, 23, 24, 25, 26, 27);
BotoesSelecao *botoes = new BotoesSelecao(8, 9, 10, 11);

void setup() {
  Serial.begin(9600);
  botoes->cadastrarObserver(painel);
  painel->cadastrarObserverTela(painel);
  painel->cadastrarObserver(almoxarifado);
  painel->setTela("INICIO");
  painel->imprimir();
}

void loop() {
  if (botoes->verificar()) {
    botoes->setLiberado(false);
  } else {
    botoes->setLiberado(true);
  }
}
```

Arquivo AlmoxarifadoAutomatico.hpp

```
#ifndef AUTOMAXARIFADOAUTOMATICO
```

```
#define AUTOMAXARIFADOAUTOMATICO

//Componentes Eletronicos
#include "../ComponentesEletronicos/SensorFimDeCurso.hpp"
#include "../ComponentesEletronicos/MotorDePasso.hpp"
#include "../ComponentesEletronicos/SensorDeCor.hpp"

//Adicionais
#include "../Observer/Interfaces/interface_observer.hpp"
#include "../Matematica/Coordenadas.hpp"
#include "../Matematica/SigmoideAceleracao.hpp"
#include "../Matematica/SigmoideDesaceleracao.hpp"

//Classe
class AlmoxarifadoAutomatico : public Coordenadas, public Observer {
   private:
      MotorDePasso *motorHorizontal, *motorVertical, *motorGarra;
      SensorFimDeCurso *sensorFimDeCursoHorizontalEsquerdo,
*sensorFimDeCursoVerticalBaixo, *sensorFimDeCursoGarra,
*sensorFimDeCursoCaixa;
      SensorDeCor *sensorDeCor;
      float RPMdeReferenciacao, RPMdeReferenciacaoGarra;
      const float avancoDoFusoVerticalEmMilimetros = 8,
avancoDoFusoHorizontalEMMilimetros = 8;
   public:
      //Método Construtor
      AlmoxarifadoAutomatico(int pinoPulsosMotorHorizontal, int
pinoDirecaoMotorHorizontal, int pinoPulsosMotorVertical,
      int pinoDirecaoMotorVertical, int pinoPulsosMotorGarra, int
pinoDirecaoMotorGarra, int pinoSensorFimDeCursoHorizontalEsquerdo,
      int pinoSensorFimDeCursoVerticalBaixo, int pinoSensorFimDeCursoGarra, int
pinoSensorFimDeCursoCaixa, int pinoOUT, int pinoS0, int pinoS1,
      int pinoS2, int pinoS3) {
```

```
        this->motorHorizontal = new MotorDePasso(200, pinoPulsosMotorHorizontal,
pinoDirecaoMotorHorizontal);
        this->motorVertical = new MotorDePasso(200, pinoPulsosMotorVertical,
pinoDirecaoMotorVertical);
        this->motorGarra = new MotorDePasso(200, pinoPulsosMotorGarra,
pinoDirecaoMotorGarra);
        this->sensorFimDeCursoHorizontalEsquerdo = new
SensorFimDeCurso(pinoSensorFimDeCursoHorizontalEsquerdo);
        this->sensorFimDeCursoVerticalBaixo = new
SensorFimDeCurso(pinoSensorFimDeCursoVerticalBaixo);
        this->sensorFimDeCursoGarra = new
SensorFimDeCurso(pinoSensorFimDeCursoGarra);
        this->sensorFimDeCursoCaixa = new
SensorFimDeCurso(pinoSensorFimDeCursoCaixa);
        this->sensorDeCor = new SensorDeCor(pinoOUT, pinoS0, pinoS1, pinoS2,
pinoS3);
        this->RPMdeReferenciacao = 200;
        this->RPMdeReferenciacaoGarra = 30;
        this->setX(0);
        this->setZ(0);
    }

    void update(String mensagem) override {
        if (mensagem == "REFERENCIAR X")
            this->referenciarX();
        else if (mensagem == "REFERENCIAR Z")
            this->referenciarZ();
        else if (mensagem == "REFERENCIAR GARRA")
            this->retrairGarra();
        else if (mensagem == "ENTREGAR CAIXA")
            this->entregarCaixa();
        else
            this->coletarCaixaDoNicho(mensagem.toInt());
    }
```

```
void entregarCaixa() {
    String cor = this->sensorDeCor->lerCor();
    Serial.println(cor);
    if (cor == "VERMELHO")
        this->entregarCaixaAoNicho(7);
    else if (cor == "VERDE")
        this->entregarCaixaAoNicho(4);
    else if (cor == "AZUL")
        this->entregarCaixaAoNicho(1);
}

void coletarCaixaDoNicho(int nicho) {
    this->moverCarroParaONicho(nicho);
    this->recolherCaixa();
    this->moverCarroParaONichoDeColeta();
    this->depositarCaixa();
    this->referenciarX();
    this->referenciarZ();
}

void entregarCaixaAoNicho(int nicho) {
    this->moverCarroParaONichoDeEntrega();
    this->recolherCaixa();
    this->moverCarroParaONicho(nicho);
    this->depositarCaixa();
    this->referenciarX();
    this->referenciarZ();
}

void depositarCaixa() {
    this->moverZ(this->getZ() + 30);
    this->avancarGarra();
    this->moverZ(this->getZ() - 20);
```

```
    this->retrairGarra();
}

void recolherCaixa() {
    this->avancarGarra();
    this->moverZ(this->getZ() + 30);
    this->retrairGarra();
}

void moverCarroParaONicho(int nicho) {
    this->moverX(this->coordenadaXDoNicho(nicho));
    this->moverZ(this->coordenadaZDoNicho(nicho));
}

void moverCarroParaONichoDeColeta() {
    this->moverX(this->coordenadaXDoNichoDeColeta());
    this->moverZ(this->coordenadaZDoNichoDeColeta());
}

void moverCarroParaONichoDeEntrega() {
    this->moverX(this->coordenadaXDoNichoDeEntrega());
    this->moverZ(this->coordenadaZDoNichoDeEntrega());
}

float coordenadaXDoNichoDeColeta() {
    return 375;
}

float coordenadaZDoNichoDeColeta() {
    return 125;
}

float coordenadaXDoNichoDeEntrega() {
    return 375;
```

```
    }

    float coordenadaZDoNichoDeEntrega() {
        return 10;
    }

    float coordenadaXDoNicho(int nicho) {
        const float distanciaEmMilimetrosEntreNichosHorizontaisConsecutivos = 125;
        int coluna = --nicho % 3;
        return coluna * distanciaEmMilimetrosEntreNichosHorizontaisConsecutivos;
    }

    float coordenadaZDoNicho(int nicho) {
        const float distanciaEmMilimetrosEntreNichosVerticaisConsecutivos = 125;
        int linha = int(--nicho / 3);
        return linha * distanciaEmMilimetrosEntreNichosVerticaisConsecutivos;
    }

    void moverX(float x) {
        this->getX() < x ? this->moverMotorParaDireita() : this->moverMotorParaEsquerda();
        float deslocamentoEmMilimetros = abs(this->getX() - x);
        float numeroDeRevolucoes = deslocamentoEmMilimetros / this->avancoDoFusoHorizontalEMMilimetros;
        int numeroDePulsosParaCompletarODeslocamento = numeroDeRevolucoes * this->motorHorizontal->getPulsosPorRevolucao();
        int numeroDePulsosDeAceleracaoEDesaceleracao = (12.75 / this->avancoDoFusoHorizontalEMMilimetros) * this->motorHorizontal->getPulsosPorRevolucao();
        SigmoideAceleracao *sigmoideAceleracao;
        SigmoideDesaceleracao *sigmoideDesaceleracao;
        //Loop de aceleracao
        int i;
        if (deslocamentoEmMilimetros < 12.75 * 2) {
```

```
            this->motorHorizontal->setRPM(281.25);
            for (i = 0; i < numeroDePulsosParaCompletarODeslocamento; i++) {
               this->motorHorizontal->pulsoUnico();
            }
         } else {
            sigmoideAceleracao = new SigmoideAceleracao(37.5, 281.25, 0,
numeroDePulsosDeAceleracaoEDesaceleracao);
            for (i = 0; i < numeroDePulsosDeAceleracaoEDesaceleracao; i++) {
               this->motorHorizontal->setRPM(sigmoideAceleracao->calcular(i));
               this->motorHorizontal->pulsoUnico();
            }
            delete sigmoideAceleracao;

            //Loop de velocidadeContinua
            this->motorHorizontal->setRPM(281.25);
            int numeroDePulsosNaVelocidadeContinua =
numeroDePulsosParaCompletarODeslocamento - 2 *
numeroDePulsosDeAceleracaoEDesaceleracao;
            for (i = 0; i < numeroDePulsosParaCompletarODeslocamento; i++) {
               this->motorHorizontal->pulsoUnico();
            }

            //Loop de desaceleração
            sigmoideDesaceleracao = new SigmoideDesaceleracao(37.5, 281.25, 0,
numeroDePulsosDeAceleracaoEDesaceleracao);
            for (i = 0; i < numeroDePulsosDeAceleracaoEDesaceleracao; i++) {
               this->motorHorizontal->setRPM(sigmoideDesaceleracao->calcular(i));
               this->motorHorizontal->pulsoUnico();
            }
            delete sigmoideDesaceleracao;
         }
         this->setX(x);
      }
```

```
    void moverZ(float z) {
        this->getZ() < z ? this->moverMotorParaCima() : this->moverMotorParaBaixo();
        float deslocamentoEmMilimetros = abs(this->getZ() - z);
        float numeroDeRevolucoes = deslocamentoEmMilimetros / this->avancoDoFusoVerticalEmMilimetros;
        int numeroDePulsosParaCompletarODeslocamento = numeroDeRevolucoes * this->motorVertical->getPulsosPorRevolucao();
        int numeroDePulsosDeAceleracaoEDesaceleracao = (9 / this->avancoDoFusoVerticalEmMilimetros) * this->motorVertical->getPulsosPorRevolucao();
        SigmoideAceleracao *sigmoideAceleracao;
        SigmoideDesaceleracao *sigmoideDesaceleracao;

        //Loop de aceleracao
        int i;
        if (deslocamentoEmMilimetros < 9 * 2) {
            this->motorVertical->setRPM(187.5);
            for (i = 0; i < numeroDePulsosParaCompletarODeslocamento; i++) {
                this->motorVertical->pulsoUnico();
            }
        } else {
            sigmoideAceleracao = new SigmoideAceleracao(37.5, 187.5, 0, numeroDePulsosDeAceleracaoEDesaceleracao);
            for (i = 0; i < numeroDePulsosDeAceleracaoEDesaceleracao; i++) {
                this->motorVertical->setRPM(sigmoideAceleracao->calcular(i));
                this->motorVertical->pulsoUnico();
            }
            delete sigmoideAceleracao;

            //Loop de velocidadeContinua
            this->motorVertical->setRPM(187.5);
```

```
            int numeroDePulsosNaVelocidadeContinua =
numeroDePulsosParaCompletarODeslocamento - 2 *
numeroDePulsosDeAceleracaoEDesaceleracao;
            for (i = 0; i < numeroDePulsosParaCompletarODeslocamento; i++) {
               this->motorVertical->pulsoUnico();
            }

            //Loop de desaceleração
            sigmoideDesaceleracao = new SigmoideDesaceleracao(37.5, 187.5, 0,
numeroDePulsosDeAceleracaoEDesaceleracao);
            for (i = 0; i < numeroDePulsosDeAceleracaoEDesaceleracao; i++) {
               this->motorVertical->setRPM(sigmoideDesaceleracao->calcular(i));
               this->motorVertical->pulsoUnico();
            }
            delete sigmoideDesaceleracao;
         }

         this->setZ(z);
      }

      void referenciarX() {
         this->motorHorizontal->setRPM(this->RPMdeReferenciacao);
         this->moverMotorParaEsquerda();
         while (!this->sensorFimDeCursoHorizontalEsquerdo->isPrecionado()) {
            this->motorHorizontal->pulsoUnico();
         }
         this->setX(0);
      }

      void referenciarZ() {
         this->motorVertical->setRPM(this->RPMdeReferenciacao);
         this->moverMotorParaBaixo();
         while (!this->sensorFimDeCursoVerticalBaixo->isPrecionado()) {
            this->motorVertical->pulsoUnico();
```

```
    }
    this->setY(0);
}

void retrairGarra() {
    this->motorGarra->setRPM(this->RPMdeReferenciacaoGarra);
    this->motorGarra->setSentidoDeRotacao(horario);
    while (!this->sensorFimDeCursoGarra->isPrecionado()) {
        this->motorGarra->pulsoUnico();
    }
}

void avancarGarra() {
    this->motorGarra->setRPM(this->RPMdeReferenciacaoGarra);
    this->motorGarra->setSentidoDeRotacao(antihorario);
    for (int i = 0; i < 185; i++) {
        this->motorGarra->pulsoUnico();
    }
}

void moverMotorParaDireita() {
    this->motorHorizontal->setSentidoDeRotacao(horario);
}

void moverMotorParaEsquerda() {
    this->motorHorizontal->setSentidoDeRotacao(antihorario);
}

void moverMotorParaCima() {
    this->motorVertical->setSentidoDeRotacao(horario);
}

void moverMotorParaBaixo() {
    this->motorVertical->setSentidoDeRotacao(antihorario);
```

```
        }

        //Método Destrutor
        ~AlmoxarifadoAutomatico() {
            delete motorHorizontal, motorVertical, motorGarra, sensorDeCor;
        }
};

#endif
```

Arquivos BotoesSelecao.hpp

```
#ifndef BOTOESSELECAO
#define BOTOESSELECAO

#include "../ComponentesEletronicos/PushButton.hpp"
#include "../Observer/Interfaces/interface_ouvinte.hpp"

class BotoesSelecao : public Ouvinte {
    private:
        //Atributos
        PushButton *botaoSuperior;
        PushButton *botaoInferior;
        PushButton *botaoEnter;
        PushButton *botaoVoltar;
        bool liberado;
    public:
        //Método Construtor
        BotoesSelecao(int pinoBotaoSuperior, int pinoBotaoInferior, int pinoBotaoEnter,
int pinoBotaoVoltar) {
            this->botaoSuperior = new PushButton(pinoBotaoSuperior);
            this->botaoInferior = new PushButton(pinoBotaoInferior);
            this->botaoEnter = new PushButton(pinoBotaoEnter);
            this->botaoVoltar = new PushButton(pinoBotaoVoltar);
```

```
}

bool verificar() {
    bool houveEvento = false;

    if (this->botaoSuperior->isPrecionado()) {
        houveEvento = true;
        if (this->liberado)
            this->notificar("SUPERIOR");
    }
    if (this->botaoInferior->isPrecionado()) {
        houveEvento = true;
        if (this->liberado)
            this->notificar("INFERIOR");
    }
    if (this->botaoEnter->isPrecionado()) {
        houveEvento = true;
        if (this->liberado)
            this->notificar("ENTER");
    }
    if (this->botaoVoltar->isPrecionado()) {
        houveEvento = true;
        if (this->liberado)
            this->notificar("VOLTAR");
    }
    return houveEvento;
}

void setLiberado(bool liberado) {
    this->liberado = liberado;
}

bool isLiberado() {
    return this->liberado;
```

```
        }
};

#endif
```

Arquivo PainelDeSelecao.hpp

```
#ifndef PAINELSELECAO
#define PAINELSELECAO

//Bibliotecas
#include <List.hpp>

#include "../Observer/Interfaces/interface_observer.hpp"
#include "../Observer/Interfaces/interface_ouvinte.hpp"
#include "../ComponentesEletronicos/LCD.hpp"
#include "../Factorys/Fabrica_Tela.hpp"

#include "../Telas/Interface/Tela.hpp"

class PainelSelecao : public Observer, public Ouvinte {
    private:
        //Atributos
        Tela *telaLCD;
        Observer *observerTela;
        String tela;

        int quantidadeLinhas, quantidadeColunas;
        int RS, E, DR4, DR5, DR6, DR7 ;

        bool primeiraTelaCriada;
    public:
        void update(String mensagem) override{
```

```
        if (mensagem == "SUPERIOR" || mensagem == "INFERIOR" || mensagem ==
"ENTER" || mensagem == "VOLTAR") {
            this->telaLCD->updateTela(mensagem);
        } else if (mensagem == "FORCAR UPDATE") {
            this->telaLCD->updateTela("");
        } else if (mensagem == "REFERENCIAR X" || mensagem ==
"REFERENCIAR Z" || mensagem == "REFERENCIAR GARRA" || mensagem ==
"ENTREGAR CAIXA" ||
        mensagem == "1" || mensagem == "2" || mensagem == "3" || mensagem ==
"4" || mensagem == "5" ||
        mensagem == "6" || mensagem == "7" || mensagem == "8" || mensagem ==
"9") {
            this->notificar(mensagem);
        } else {
            this->setTela(mensagem);
        }
        this->imprimir();
    }

    //Método Construtor
    PainelSelecao(int colunas, int linhas, int RS, int E, int DR4, int DR5, int DR6, int
DR7) {
        this->setPinos(RS, E, DR4, DR5, DR6, DR7);
        this->setLinhasColunas(linhas, colunas);
        this->primeiraTelaCriada = true;
        this->setTela("INICIO");
    }

    //Métodos
    void imprimir() {
        this->telaLCD->limparTela();
        this->telaLCD->imprimir();
        this->telaLCD->posicionarCursor(0, 1);
    }
```

```
void cadastrarObserverTela(Observer *observer) {
    this->telaLCD->cadastrarObserver(observer);
    this->observerTela = observer;
}

//Métodos Getters e Setters
void setPinos(int RS, int E, int DR4, int DR5, int DR6, int DR7) {
    this->RS = RS;
    this->E = E;
    this->DR4 = DR4;
    this->DR5 = DR5;
    this->DR6 = DR6;
    this->DR7 = DR7;
}

void setLinhasColunas(int linhas, int colunas) {
    this->quantidadeLinhas = linhas;
    this->quantidadeColunas = colunas;
}

String getTela() {
    return this->tela;
}

void setTela(String tela) {
    if (!this->primeiraTelaCriada) {
        Tela* telaAntiga = this->telaLCD;
        this->telaLCD = nullptr;
        this->primeiraTelaCriada = false;
        delete telaAntiga;
    }
    FabricaTela *fabricaTela = new FabricaTela;
```

```
        fabricaTela->setColunasELinhas(this->quantidadeColunas, this->quantidadeLinhas);
        fabricaTela->setPinos(this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
        this->telaLCD = fabricaTela->criar(tela);
        this->telaLCD->setPrimeiraLinhaDeVisualizacao(1);
        this->cadastrarObserverTela(this->observerTela);
        this->tela = tela;
        delete fabricaTela;
    }
};

#endif
```

Arquivo ComponenteEletronico.hpp

```
#ifndef COMPONENTEELETRONICO
#define COMPONENTEELETRONICO

class ComponenteEletronico {};

#endif
```

Arquivo LCD.hpp

```
 #ifndef _LCD
#define _LCD

#include <LiquidCrystal.h>
#include <stdarg.h>
#include "./Interface/ComponenteEletronico.hpp"


class LCD : public ComponenteEletronico {
```

```
protected:
    LiquidCrystal *lcd;
    String mensagemLinha[15];
    int quantidadeDeLinhasMensagem;
    int quantidadeLinhas, quantidadeColunas;
    int primeiraLinhaDeVisualizacao;
public:
//Método Construtor

void definirLinhasColunas(int linhas, int colunas) {
    this->lcd->begin(colunas, linhas);
    this->quantidadeLinhas = linhas;
    this->quantidadeColunas = colunas;
}

void definirPinos(int RS, int E, int DR4, int DR5, int DR6, int DR7) {
    this->lcd = new LiquidCrystal(RS, E, DR4, DR5, DR6, DR7);
}

//Métodos
void definirMensagem(int quantidadeDeLinhasMensagem, ...) {
    this->quantidadeDeLinhasMensagem = quantidadeDeLinhasMensagem;
    va_list mensagem;
    va_start (mensagem, quantidadeDeLinhasMensagem);
    for (int i = 0; i < quantidadeDeLinhasMensagem; i++) {
        this->mensagemLinha[i] = va_arg(mensagem, char*);
    }
    va_end(mensagem);
}

void imprimir() {
    this->lcd->clear();
    this->posicionarCursor(0, 0);
    this->lcd->print(this->mensagemLinha[this->primeiraLinhaDeVisualizacao - 1]);
```

```
        this->posicionarCursor(0, 1);
        this->lcd->print(this->mensagemLinha[this->primeiraLinhaDeVisualizacao]);
    }

    void ligarCursor() {
        this->lcd->cursor();
    }

    void desligarCursor() {
        this->lcd->noCursor();
    }

    void posicionarCursor(int coluna, int linha) {
        this->lcd->setCursor(coluna, linha);
    }

    void limparTela() {
        this->lcd->clear();
    }

    //Métodos Getters e Setters
    int getQuantidadeDeLinhasMensagem() {
        return this->quantidadeDeLinhasMensagem;
    }

    int getPrimeiraLinhaDeVisualizacao() {
        return this->primeiraLinhaDeVisualizacao;
    }

    void setPrimeiraLinhaDeVisualizacao(int primeiraLinhaDeVisualizacao) {
        if (primeiraLinhaDeVisualizacao < 1 || primeiraLinhaDeVisualizacao > this-
>getQuantidadeDeLinhasMensagem() - 1)
            this->primeiraLinhaDeVisualizacao = 1;
        else
```

```
        this->primeiraLinhaDeVisualizacao = primeiraLinhaDeVisualizacao;
    }

    //Métodos Getters e Setters
    String getMensagemLinha(int linha) {
        return this->mensagemLinha[linha];
    }

    String lerLinhaDoCursor() {
        return this->getMensagemLinha(this->getPrimeiraLinhaDeVisualizacao());
    }

    //Método Destrutor
    ~LCD() {
        delete this->lcd;
    }
};

#endif
```

Arquivo MotorDePasso.hpp

```
#ifndef MOTORDEPASSO
#define MOTORDEPASSO

#include "./Interface/ComponenteEletronico.hpp"

enum SentidoDeRotacao {horario = 0, antihorario = 1};

class MotorDePasso : public ComponenteEletronico {
    private:
        //Atributos
        int pulsosPorRevolucao;
        int pinoPulsos;
```

```
    int pinoDirecao;
    float RPM;
    SentidoDeRotacao sentidoDeRotacao;
    int tempoDePulsoMicroSegundos;
  public:
    //Método Construtor
    MotorDePasso(int pulsosPorRevolucao, int pinoPulsos, int pinoDirecao) {
      this->pulsosPorRevolucao = pulsosPorRevolucao;
      this->pinoPulsos = pinoPulsos;
      this->pinoDirecao = pinoDirecao;
      pinMode(this->pinoPulsos, OUTPUT);
      pinMode(this->pinoDirecao, OUTPUT);
      this->setRPM(200);
      this->setSentidoDeRotacao(horario);
    }

    //Métodos
    void pulsoUnico() {
      digitalWrite(this->pinoPulsos, HIGH);
      delayMicroseconds(4);
      digitalWrite(this->pinoPulsos, LOW);
      delayMicroseconds(this->tempoDePulsoMicroSegundos);
    }

    //Métodos Getters e Setters
    void setRPM(float RPM) {
      this->RPM = RPM;
      this->tempoDePulsoMicroSegundos = 60000000 / (this->RPM * this-
>pulsosPorRevolucao);
    }

    float getRPM() {
      return this->RPM;
    }
```

```
        void setSentidoDeRotacao(SentidoDeRotacao sentidoDeRotacao) {
            this->sentidoDeRotacao = sentidoDeRotacao;
            digitalWrite(this->pinoDirecao, sentidoDeRotacao == horario? LOW : HIGH);
        }

        int getPulsosPorRevolucao() {
            return this->pulsosPorRevolucao;
        }

        int getPinoPulsos() {
            return this->pinoPulsos;
        }
};

#endif
```

Arquivo PushButton.hpp

```
#ifndef PUSHBUTTON
#define PUSHBUTTON

#include "./Interface/ComponenteEletronico.hpp"

class PushButton : public ComponenteEletronico {
    private:
        //Atrubutos
        int pino;
    public:
        //Métodos Construtores
        PushButton(int pino) {
            this->pino = pino;
            pinMode(pino, INPUT);
        }
```

```
        //Métodos
        bool isPrecionado() {
            return digitalRead(this->pino) == HIGH ? true : false;
        }
};

#endif
```

Arquivo SensorDeCor.hpp

```
#ifndef SENSORDECOR
#define SENSORDECOR

class SensorDeCor {
    private:
        //Atributos
        int pinoOUT, pinoS0, pinoS1, pinoS2, pinoS3;

        //Métodos Privados
        unsigned long lerVermelho(int pinoS2, int pinoS3, int pinoOUT) {
            digitalWrite(pinoS2, LOW);
            digitalWrite(pinoS3, LOW);
            delay(50);
            return pulseIn(pinoOUT, digitalRead(pinoOUT) == HIGH ? LOW : HIGH);
        }

        unsigned long lerAzul(int pinoS2, int pinoS3, int pinoOUT) {
            digitalWrite(pinoS2, LOW);
            digitalWrite(pinoS3, HIGH);
            delay(50);
            return pulseIn(pinoOUT, digitalRead(pinoOUT) == HIGH ? LOW : HIGH);
        }
```

```
    unsigned long lerVerde(int pinoS2, int pinoS3, int pinoOUT) {
      digitalWrite(pinoS2, HIGH);
      digitalWrite(pinoS3, HIGH);
      delay(50);
      return pulseIn(pinoOUT, digitalRead(pinoOUT) == HIGH ? LOW : HIGH);
    }

    unsigned long lerBranco(int pinoS2, int pinoS3, int pinoOUT) {
      digitalWrite(pinoS2, HIGH);
      digitalWrite(pinoS3, LOW);
      delay(50);
      return pulseIn(pinoOUT, digitalRead(pinoOUT) == HIGH ? LOW : HIGH);
    }

    bool corVermelhaDetectada(unsigned int valorVermelho, unsigned int
valorVerde, unsigned int valorAzul, unsigned int valorBranco) {
      return valorVermelho < valorVerde && valorVermelho < valorAzul &&
valorBranco < 100 ? true : false;
    }

    bool corVerdeDetectada(unsigned int valorVermelho, unsigned int valorVerde,
unsigned int valorAzul, unsigned int valorBranco) {
      return valorVerde < valorVermelho && valorVerde < valorAzul &&
valorBranco < 100 ? true : false;
    }

    bool corAzulDetectada(unsigned int valorVermelho, unsigned int valorVerde,
unsigned int valorAzul, unsigned int valorBranco) {
      return valorAzul < valorVermelho && valorAzul < valorVerde && valorBranco
< 100 ? true : false;
    }
  public:
    //Método Construtor
    SensorDeCor(int pinoOUT, int pinoS0, int pinoS1, int pinoS2, int pinoS3) {
```

```
        this->pinoOUT = pinoOUT;
        this->pinoS0 = pinoS0;
        this->pinoS1 = pinoS1;
        this->pinoS2 = pinoS2;
        this->pinoS3 = pinoS3;

        pinMode(pinoOUT, INPUT);
        pinMode(pinoS0, OUTPUT);
        pinMode(pinoS1, OUTPUT);
        pinMode(pinoS2, OUTPUT);
        pinMode(pinoS3, OUTPUT);

        digitalWrite(this->pinoS0, HIGH);
        digitalWrite(this->pinoS1, LOW);
    }

    //Métodos Públicos
    String lerCor() {
        unsigned int valorVermelho = lerVermelho(this->pinoS2, this->pinoS3, this->pinoOUT);
        unsigned int valorVerde = lerVerde(this->pinoS2, this->pinoS3, this->pinoOUT);
        unsigned int valorAzul = lerAzul(this->pinoS2, this->pinoS3, this->pinoOUT);
        unsigned int valorBranco = lerBranco(this->pinoS2, this->pinoS3, this->pinoOUT);

        if (this->corVermelhaDetectada(valorVermelho, valorVerde, valorAzul, valorBranco)) {
            return "VERMELHO";
        }
        if (this->corVerdeDetectada(valorVermelho, valorVerde, valorAzul, valorBranco)) {
            return "VERDE";
        }
```

```
        if (this->corAzulDetectada(valorVermelho, valorVerde, valorAzul,
valorBranco)) {
            return "AZUL";
        }
        return "";
    }
};

#endif
```

Arquivo SensorFimDeCurso.hpp

```
#ifndef SENSORDEFIMDECURSO
#define SENSORDEFIMDECURSO

#include "./PushButton.hpp"

class SensorFimDeCurso : public PushButton {
  public:
    using PushButton::PushButton;
};

#endif
```

Arquivo Fabrica_ComponentesEletronicos.hpp

```
#ifndef INTERFACE_FABRICACOMPONENTESELETRONICOS
#define INTERFACE_FABRICACOMPONENTESELETRONICOS

#include "../../ComponentesEletronicos/Interface/ComponenteEletronico.hpp"

class FabricaComponentesEletronicos : public ComponenteEletronico {
  public:
    virtual ComponenteEletronico* criar(String nome) = 0;
```

```
};

#endif
```

Arquivo Fabrica_Tela.hpp

```
#ifndef FABRICA_TELA
#define FABRICA_TELA

#include "../ComponentesEletronicos/LCD.hpp"
#include "./Interfaces/Fabrica_ComponentesEletronicos.hpp"

//Telas
#include "../Telas/Inicio.hpp"
#include "../Telas/Modo.hpp"
#include "../Telas/Coleta.hpp"
#include "../Telas/Referenciar.hpp"
#include "../Telas/ReferenciacaoCompleta.hpp"
#include "../Telas/ReferenciandoEixoX.hpp"
#include "../Telas/ReferenciandoEixoZ.hpp"
#include "../Telas/ReferenciandoGarra.hpp"
#include "../Telas/AguardandoCaixa.hpp"
#include "../Telas/EntregandoCaixa.hpp"
#include "../Telas/EntregaFinalizada.hpp"
#include "../Telas/ColetaFinalizada.hpp"
#include "../Telas/ColetandoCaixa.hpp"

class FabricaTela : FabricaComponentesEletronicos {
    private:
        int RS, E, DR4, DR5, DR6, DR7;
        int colunas, linhas;
    public:

        Tela* criar(String nome) {
```

```
        Tela *tela;

        if (nome == "INICIO") {
            tela =  new TelaInicio(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "MODO") {
            tela =  new TelaModo(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->ligarCursor();
        } else if (nome == "COLETA") {
            tela =  new TelaColeta(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->ligarCursor();
        } else if (nome == "REFERENCIAR") {
            tela =  new TelaReferenciar(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "REFERENCIANDO X"){
            tela =  new TelaReferenciandoX(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "REFERENCIANDO Z") {
            tela =  new TelaReferenciandoZ(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "REFERENCIACAO COMPLETA") {
            tela =  new TelaReferenciacaoCompleta(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "REFERENCIANDO GARRA") {
            tela = new TelaReferenciandoGarra(this->colunas, this->linhas, this->RS, this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
```

```
        } else if (nome == "ENTREGA") {
            tela = new TelaAguardandoCaixa(this->colunas, this->linhas, this->RS,
this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "ENTREGANDO CAIXA") {
            tela = new TelaEntregandoCaixa(this->colunas, this->linhas, this->RS, this-
>E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "ENTREGA FINALIZADA") {
            tela = new TelaEntregaFinalizada(this->colunas, this->linhas, this->RS,
this->E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "COLETA FINALIZADA") {
            tela = new TelaColetaFinalizada(this->colunas, this->linhas, this->RS, this-
>E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        } else if (nome == "COLETANDO CAIXA") {
            tela = new TelaColetandoCaixa(this->colunas, this->linhas, this->RS, this-
>E, this->DR4, this->DR5, this->DR6, this->DR7);
            tela->desligarCursor();
        }

        return tela;
    }

    void setPinos(int RS, int E, int DR4, int DR5, int DR6, int DR7) {
        this->RS = RS;
        this->E = E;
        this->DR4 = DR4;
        this->DR5 = DR5;
        this->DR6 = DR6;
        this->DR7 = DR7;
    }
```

```
        void setColunasELinhas(int colunas, int linhas) {
            this->colunas = colunas;
            this->linhas = linhas;
        }
};

#endif
```

Arquivo Coordenadas.hpp

```
#ifndef COORDENADAS
#define COORDENADAS

class Coordenadas {
    protected:
        float x, y, z;
    public:
        void setX(float x) {
            this->x = x;
        }

        void setY(float y) {
            this->y = y;
        }

        void setZ(float z) {
            this->z = z;
        }

        float getX() {
            return this->x;
        }

        float getY() {
```

```
        return this->y;
    }

    float getZ() {
        return this->z;
    }
};

#endif
```

Arquivo SigmoideAceleracao.hpp

```
#ifndef SIGMOIDE_ACELERACAO
#define SIGMOIDE_ACELERACAO

#include <math.h>

#define E 2.71828182845904523536028747

class SigmoideAceleracao {
  private:
    float coeficienteDeAclive, velocidadeNominal, velocidadeInicial, pontoMedio, tempoInicial, tempoFinal;
    float amplitudeDeVelocidade;
  public:
    SigmoideAceleracao(float velocidadeInicial, float velocidadeNominal, float tempoInicial, float tempoFinal) {
        this->tempoInicial = tempoInicial;
        this->tempoFinal = tempoFinal;
        this->velocidadeNominal = velocidadeNominal;
        this->velocidadeInicial = velocidadeInicial;

        this->amplitudeDeVelocidade = (velocidadeNominal - velocidadeInicial);
        this->pontoMedio = (tempoFinal + tempoInicial) / 2;
```

```
        this->coeficienteDeAclive = 2 * log (velocidadeNominal / (99 * velocidadeNominal - 100 * velocidadeInicial)) / (tempoFinal - tempoInicial);
    }

    inline float calcular(float instante) {
        return this->velocidadeInicial + this->amplitudeDeVelocidade / (1 + pow(E, coeficienteDeAclive * (instante - pontoMedio)));
    }

};

#endif
```

Arquivo SigmoideDesaceleracao.hpp

```
#ifndef SIGMOIDE_DESACELERACAO
#define SIGMOIDE_DESACELERACAO

#include <math.h>

#define E 2.71828182845904523536028747

class SigmoideDesaceleracao {
  private:
    float velocidadeNominal, velocidadeFinal, tempoInicial, tempoFinal, coeficienteDeDeclive;
    float pontoMedio, amplitudeDeVelocidade;
  public:
    SigmoideDesaceleracao(float velocidadeFinal, float velocidadeNominal, float tempoInicial, float tempoFinal) {
        this->tempoInicial = tempoInicial;
        this->tempoFinal = tempoFinal;
        this->velocidadeNominal = velocidadeNominal;
```

```
        this->velocidadeFinal = velocidadeFinal;

        this->amplitudeDeVelocidade = (velocidadeNominal - velocidadeFinal);
        this->pontoMedio = (tempoFinal + tempoInicial) / 2;

        this->coeficienteDeDeclive = 2 * log (velocidadeNominal / (99 * velocidadeNominal - 100 * velocidadeFinal)) / (tempoInicial - tempoFinal);
    }

    inline float calcular(float instante) {
        return this->velocidadeFinal + this->amplitudeDeVelocidade / (1 + pow(E, coeficienteDeDeclive * (instante - pontoMedio)));
    }
};

#endif
```

Arquivo interface_observer.hpp

```
#ifndef INTERFACE_OBSERVER
#define INTERFACE_OBSERVER

class Observer {
  public:
    virtual void update(String mensagem) = 0;
};

#endif
```

Arquivo interface_ouvinte.hpp

```
#ifndef INTERFACE_OUVINTE
#define INTERFACE_OUVINTE
```

```
#include <List.hpp>
#include "./interface_observer.hpp"

class Ouvinte {
    private:
        List <Observer*> listaDeObservers;
    public:
        void cadastrarObserver(Observer* observer) {
            this->listaDeObservers.add(observer);
        }

        void removerObserver(Observer* observer) {
            for (int i = 0; i < this->listaDeObservers.getSize(); i++)
                if (this->listaDeObservers[i] == observer)
                    this->listaDeObservers.remove(i);
        }

        void notificar(String mensagem) {
            for (int i = 0; i < this->listaDeObservers.getSize(); i++)
                this->listaDeObservers[i]->update(mensagem);
        }
};

#endif
```

Arquivo Tela.hpp

```
#ifndef TELA
#define TELA

#include "../../ComponentesEletronicos/LCD.hpp"
#include "../../Observer/Interfaces/interface_ouvinte.hpp"
```

```
class Tela : public LCD, public Ouvinte {
   public:
      virtual void updateTela(String botaoPrecionado) = 0;
};

#endif
```

Arquivo AguardandoCaixa.hpp

```
#ifndef TELAAGUARDANDOCAIXA
#define TELAAGUARDANDOCAIXA

#include "./Interface/Tela.hpp"

class TelaAguardandoCaixa : public Tela {
   public:
      TelaAguardandoCaixa(int colunas, int linhas, int RS, int E, int DR4, int DR5, int
DR6, int DR7) {
         this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
         this->definirLinhasColunas(linhas, colunas);
         this->definirMensagem(2, "PRECIONE ENTER", "P/ LER A CAIXA");
      }

      void updateTela(String botaoPrecionado) override {
         this->imprimir();
         if (botaoPrecionado == "VOLTAR") {
            this->notificar("MODO");
         } else if (botaoPrecionado == "ENTER") {
            this->notificar("ENTREGANDO CAIXA");
            this->notificar("FORCAR UPDATE");
         }
      }
};
```

```
#endif
```

Arquivo Coleta.hpp

```
#ifndef TELACOLETA
#define TELACOLETA

#include "./Interface/Tela.hpp"

class TelaColeta : public Tela {
   public:
      TelaColeta(int colunas, int linhas, int RS, int E, int DR4, int DR5, int DR6, int
DR7) {
         this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
         this->definirLinhasColunas(linhas, colunas);
         this->definirMensagem(10, "COLETA NICHO:", "1", "2", "3", "4", "5", "6", "7",
"8", "9");
      }

      void updateTela(String botaoPrecionado) override {
         String nicho;
         if (botaoPrecionado == "SUPERIOR") {
            this->setPrimeiraLinhaDeVisualizacao(this-
>getPrimeiraLinhaDeVisualizacao() - 1);
         } else if (botaoPrecionado == "INFERIOR") {
            this->setPrimeiraLinhaDeVisualizacao(this-
>getPrimeiraLinhaDeVisualizacao() + 1);
         } else if (botaoPrecionado == "VOLTAR") {
            this->notificar("MODO");
         } else if (botaoPrecionado == "ENTER") {
            nicho = this->lerLinhaDoCursor();
            this->notificar("COLETANDO CAIXA");
            this->imprimir();
            this->notificar(nicho);
```

```
            this->notificar("FORCAR UPDATE");
        }
        this->imprimir();
    }
};


#endif
```

Arquivo ColetaFinalizada.hpp

```
#ifndef TELACOLETAFINALIZADA
#define TELACOLETAFINALIZADA

#include "./Interface/Tela.hpp"

class TelaColetaFinalizada : public Tela {
  public:
    TelaColetaFinalizada(int colunas, int linhas, int RS, int E, int DR4, int DR5, int
DR6, int DR7) {
        this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
        this->definirLinhasColunas(linhas, colunas);
        this->definirMensagem(2, "COLETA", "FINALIZADA");
    }

    void updateTela(String botaoPrecionado) override {
        this->imprimir();
        delay(2000);
        this->notificar("MODO");
    }
};


#endif
```

Arquivo ColetandoCaixa.hpp

```
#ifndef TELACOLETANDOCAIXA
#define TELACOLETANDOCAIXA

#include "./Interface/Tela.hpp"

class TelaColetandoCaixa : public Tela {
   public:
      TelaColetandoCaixa(int colunas, int linhas, int RS, int E, int DR4, int DR5, int DR6, int DR7) {
         this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
         this->definirLinhasColunas(linhas, colunas);
         this->definirMensagem(2, "COLETANDO CAI-", "XA...");
      }

      void updateTela(String botaoPrecionado) override {
         this->imprimir();
         delay(500);
         this->notificar("COLETA FINALIZADA");
         this->notificar("FORCAR UPDATE");
      }
};

#endif
```

Arquivo EntregaFinalizada.hpp

```
#ifndef TELAENTREGAFINALIZADA
#define TELAENTREGAFINALIZADA

#include "./Interface/Tela.hpp"
```

```
class TelaEntregaFinalizada : public Tela {
   public:
      TelaEntregaFinalizada(int colunas, int linhas, int RS, int E, int DR4, int DR5, int
DR6, int DR7) {
         this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
         this->definirLinhasColunas(linhas, colunas);
         this->definirMensagem(2, "ENTREGA", "FINALIZADA");
      }

      void updateTela(String botaoPrecionado) override {
         this->imprimir();
         delay(2000);
         this->notificar("MODO");
         this->notificar("FORCAR UPDATE");
      }
};

#endif
```

Arquivo ReferenciandoEixoZ.hpp

```
#ifndef TELAREFZ
#define TELAREFZ

#include "./Interface/Tela.hpp"

class TelaReferenciandoZ : public Tela {
   public:
      TelaReferenciandoZ(int colunas, int linhas, int RS, int E, int DR4, int DR5, int
DR6, int DR7) {
         this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
         this->definirLinhasColunas(linhas, colunas);
         this->definirMensagem(2, "REFERENCIANDO Z ", "AGUARDE...");
      }
```

```
        void updateTela(String botaoPrecionado) override {
            this->imprimir();
            delay(500);
            this->notificar("REFERENCIAR Z");
            this->notificar("REFERENCIACAO COMPLETA");
            this->notificar("FORCAR UPDATE");
        }
};

#endif
```

Arquivo ReferenciandoGarra.hpp

```
#ifndef TELAREFERENCIANDOGARRA
#define TELAREFERENCIANDOGARRA

#include "./Interface/Tela.hpp"

class TelaReferenciandoGarra : public Tela {
    public:
        TelaReferenciandoGarra(int colunas, int linhas, int RS, int E, int DR4, int DR5,
int DR6, int DR7) {
            this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
            this->definirLinhasColunas(linhas, colunas);
            this->definirMensagem(2, "REFERENCIANDO", "GARRA AGUARDE..");
        }

        void updateTela(String botaoPrecionado) override {
            this->imprimir();
            delay(500);
            this->notificar("REFERENCIAR GARRA");
            this->notificar("REFERENCIANDO X");
            this->notificar("FORCAR UPDATE");
```

```
    }
};

#endif
```

Arquivo Referenciar.hpp

```
#ifndef TELAREFERENCIAR
#define TELAREFERENCIAR

#include "./Interface/Tela.hpp"

class TelaReferenciar : public Tela {
  public:
    TelaReferenciar(int colunas, int linhas, int RS, int E, int DR4, int DR5, int DR6,
int DR7) {
        this->definirPinos(RS, E, DR4, DR5, DR6, DR7);
        this->definirLinhasColunas(linhas, colunas);
        this->definirMensagem(2, "PRECIONE ENTER", "P/ REFERENCIAR");
    }

    void updateTela(String botaoPrecionado) override {
        if (botaoPrecionado == "ENTER") {
            this->notificar("REFERENCIANDO GARRA");
            this->notificar("FORCAR UPDATE");
        } else if (botaoPrecionado == "VOLTAR") {
            this->notificar("MODO");
        }
    }
};

#endif
```