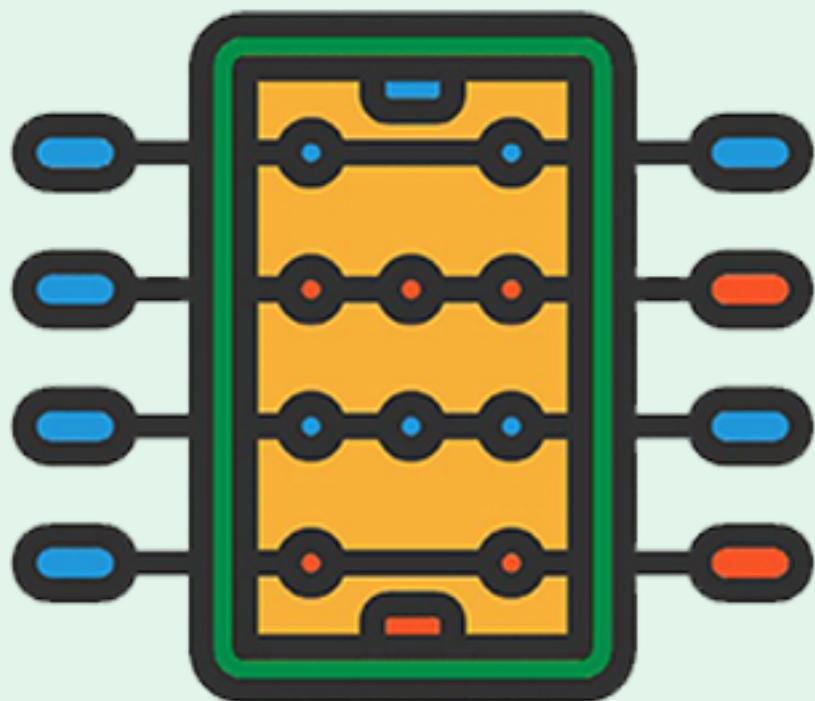




Universidade de Brasília

Autoosball



Faculdade do Gama - FGA



Grupo Autonomous Foosball

Autonomous Foosball

Relatório do Ponto de Controle 3 da disciplina de Projeto Integrador de Engenharias 2.

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

Brasília, DF
2020

Lista de ilustrações

Figura 1 – Gráfico Torque x RPM (SERVOTRONIX, 2014)	16
Figura 2 – Gráfico Torque x RPM (SERVOTRONIX, 2014)	17
Figura 3 – Motores de deslocamento linear	19
Figura 4 – Motores de chute	19
Figura 5 – Drivers dos motores e placa controladora	19
Figura 6 – Diagrama Lógico do Circuito Eletrônico	22
Figura 7 – Fluxograma do SESC	24
Figura 8 – Iniciando o SESC	24
Figura 9 – Mapeamento eletrônico da mesa, ponto máximo	25
Figura 10 – Mapeamento eletrônico da mesa, ponto mínimo	25
Figura 11 – Centralização eletrônica da mesa	25
Figura 12 – fluxograma do antitravamento da bola	27
Figura 13 – Campo de jogo no plano cartesiano (mm)	28
Figura 14 – Configuração atual da mesa	29
Figura 15 – Campo	30
Figura 16 – Apoio da câmera, esboço	31
Figura 17 – Suporte do display	32
Figura 18 – Mesa Deformada	33
Figura 19 – Condições de contorno	33
Figura 20 – Tensão na Treliça	34
Figura 21 – Ponto de máxima tensão von mises	34
Figura 22 – Otimização	35
Figura 23 – Passo 1	38
Figura 24 – Passo 2	39
Figura 25 – Passo 3	39
Figura 26 – Passo 4	40
Figura 27 – Passo 5	41
Figura 28 – Passo 6	42
Figura 29 – Passo 7	42
Figura 30 – Passo 8	43
Figura 31 – Passo 9	43
Figura 32 – Passo 10	44
Figura 33 – Passo 11	45
Figura 34 – Passo 12	46
Figura 35 – Passo 13	46
Figura 36 – Passo 14	47

Figura 37 – Estrutura da Solução de Software	48
Figura 38 – Diagrama de Atividades da Solução de Software	50
Figura 39 – Caso de Uso da Solução de Software	51
Figura 40 – Fluxograma do PUDM	53
Figura 41 – Fluxo do Servidor de Decisão	53
Figura 42 – Diagrama de Sequência do Servidor de Decisão	54
Figura 43 – Diagrama de Classe do Servidor de Decisão	56
Figura 44 – Tela de Apresentação	58
Figura 45 – Tela para selecionar a dificuldade de jogo	58
Figura 46 – Tela responsável por calibrar a câmera da mesa	59
Figura 47 – Tela do placar de jogo	59
Figura 48 – Tela do placar de jogo sendo executada	60
Figura 49 – Imagem do campo de jogo gerada pela simulação	61
Figura 50 – Imagem do campo de jogo após a troca do sistema de cor	61
Figura 51 – Máscara da imagem do campo de jogo para marcar a posição da bola .	62
Figura 52 – Imagem do campo de jogo com a bola marcada	62
Figura 53 – Tempo médio (em ms) de execução do algoritmo de detecção da bola de pebolim	63
Figura 54 – Tempo médio (em ms) de execução do algoritmo de tomada de decisão .	64
Figura 55 – Verificação das funcionalidades do Servidor de Decisão	66
Figura 56 – Diagrama de estados do servidor de decisão	67
Figura 57 – Diagrama de sequencia da parte de envio de cliente PUDM	68
Figura 58 – Diagrama de sequencia da parte de recebimento de uma decisão gerada pelo Servidor de Decisão	69
Figura 59 – Caso de Uso da Simulação na Engine de Jogos	70
Figura 60 – Imagens da Simulação da Mesa de Pebolim	71
Figura 61 – Movimento da Bola na Simulação	72
Figura 62 – Chute de um Jogador na Simulação	73
Figura 63 – Demonstração da Simulação executada com dois Servidores de Decisão	74
Figura 64 – Diagrama de classe da simulação	75
Figura 65 – Diagrama de membros	91
Figura 66 – Análise SWOT	94
Figura 67 – Diagrama de Ishikawa	95
Figura 68 – EAP - Estrutura Analítica do Projeto Autonomous Foosball	106
Figura 69 – Gráfico Gantt	107
Figura 70 – Cronograma Parte 1	107
Figura 71 – Cronograma Parte 2	108
Figura 72 – Cronograma Parte 3	109
Figura 73 – Características do motor no sistema	121

Figura 74 – Microprocessador Raspberry Pi 4 Model B Anatel.(FILIPE FLOP, 2020b)	122
Figura 75 – Microcontrolador Mega 2560 R3	123
Figura 76 – Câmera Compatível Raspberry Pi 5MP.(FILIPE FLOP, 2020a)	124
Figura 77 – Tela Lcd Hdmi 10.1 1024x800 Ips Touch Screen Raspberry Pi.(MERCADO LIVRE, 2020)	125
Figura 78 – Driver Para Motor de Passo 5A WD-TB6600 - Wotiom.(BAÚ DA ELETRÔNICA, 2020)	126
Figura 79 – Fonte Meanwell 1000-24(AMERICANAS, 2020)	127
Figura 80 – Fonte Raspberry Oficial (ROBOCORE, 2020)	129

Lista de tabelas

Tabela 1 – Motores NEMA 23s e 34s	16
Tabela 2 – Posições e deslocamento linear dos jogadores (mm)	28
Tabela 3 – Detalhamento dos componentes	31
Tabela 4 – Frequências modais	33
Tabela 5 – Tabela de componentes	37
Tabela 6 – Tempo médio de execução do algoritmo de detecção da bola de pebolim	64
Tabela 7 – Tempo médio de execução do algoritmo de tomada de decisão	65
Tabela 8 – Matriz de Probabilidade	81
Tabela 9 – Matriz de Impacto	81
Tabela 10 – Matriz de Prioridade	81
Tabela 11 – Níveis de Prioridades	82
Tabela 12 – Lista É/Não é	82
Tabela 13 – Riscos Iniciais do Projeto	85
Tabela 14 – Equipe do Projeto	86
Tabela 15 – Cronograma e Marcos	87
Tabela 16 – Diretores e Técnicos	89
Tabela 17 – Equipes e Responsabilidades	90
Tabela 18 – Ferramentas de comunicação	91
Tabela 19 – Riscos Gerais	96
Tabela 20 – Resposta aos riscos gerais	97
Tabela 21 – Riscos de estruturas	98
Tabela 22 – Resposta aos riscos de estruturas	99
Tabela 23 – Riscos equipe de motores	100
Tabela 24 – Resposta aos riscos equipe de motores	101
Tabela 25 – Riscos da Equipe de Software	102
Tabela 26 – Continuação dos Riscos Negativos da Engenharia de Software	103
Tabela 27 – Resposta aos Riscos da Equipe de Software	104
Tabela 28 – Continuação da Resposta aos Riscos da Equipe de Software	105
Tabela 29 – Tabela de Requisitos da equipe de motores.	110
Tabela 30 – Tabela de Requisitos Estruturais	111
Tabela 31 – Temáticas de Software	112
Tabela 32 – Features de Software	112
Tabela 33 – Histórias de Usuário	113
Tabela 34 – Histórias de Usuário	114
Tabela 35 – Custos com componentes eletrônicos/alimentação	115
Tabela 36 – Custos dos componentes estruturais/mecânicos	116

Tabela 37 – Motores NEMA 23s e 34s (SERVOTRONIX, 2014)	130
Tabela 38 – Descrição dos eventos do PUDM no repositório	167
Tabela 39 – Descrição das funcionalidades do Servidor de Decisão no repositório	167
Tabela 40 – Descrição das funcionalidades da Unity no repositório	168

Lista de Símbolos, Nomenclaturas e Abreviações

- AC - Alternate Current
- AES - Advanced Encryption Standard
- CC - Continuous Current
- CEC - Consumer Electronics Control
- CRC - Cyclic Redundancy Check
- CSI - Camera Serial Interface
- DDR4 - Double Data Rate Fourth Generation
- DIP - Dual in-line Package
- DSI - Display Serial Interface
- DSI - Display Serial Interface
- EAP - Estrutura Analítica do Projeto
- EEPROM - Electrically-Erasable Programmable Read-Only Memory
- eMMC - Embedded Multimedia Card
- ESCS - Eletronic Stability Control System
- FGA - Faculdade Gama
- FIT - Feira de Inovação e Tecnologia
- FMM - Força Magnetomotriz
- GND - Ground
- GPIO - General Purpose Input/Output
- GPU - Graphics Processing Unit
- GSFK - Gaussian Frequency-Shift Keying
- HD - High Definition

- HDMI - High-Definition Multimedia Interface
- I2C - Inter-Integrated Circuit
- I2S - Integrated Interchip Sound
- IA - Inteligência artificial
- IDE - Integrated Development Environment
- IOT - Internet of Thinks
- ITSF - International Table Soceer Federation
- LAN - Local Area Network
- LCD - Liquid Cristal Display
- LPDDR3 - Low Power Double Data Rate 3
- MIPI - Mobile Industry Processor Interface
- OTG - On the Go
- PC - Ponto de Controle
- PCI - Peripheral Component Interconnect
- PIC - Peripheral Interface Controller
- PUDM - Pebolim Unified Data Model
- PWM - Pulse Width Modulation
- RAM - Random Acess Memory
- RF - Radio Frequency
- ROM - Read Only Memory
- SD - Secure Digital Card
- SDRAM - Synchronous dynamic random-access memory
- SESC - Sistema Eletronico de Simulação e Controle
- SPI - Serial Peripheral Interface
- SRAM - Static Random Access Memory
- TAP - Termo de Abertura de Projeto

- TKIP - Temporal Key Integrity Protocol
- UART - Universal Asynchronous Receiver/Transmitter
- UNB - Universidade de Brasília
- USB - Universal Serial Bus
- VCC - Common Collector Voltage
- VDC - Volts Direct Current
- WEP - Wired Equivalent Privacy
- WPA - Wi-Fi Protected Access

Sumário

1	APRESENTAÇÃO	14
1.1	Contextualização	14
2	SOLUÇÃO	15
2.1	Solução Geral	15
2.2	Soluções da equipe de motores	15
2.2.1	Sistema de motores	15
2.2.2	Dimensionamento dos motores	15
2.2.3	Rotação e Movimentação linear	17
2.2.4	Alimentação do projeto	17
2.2.5	Sistema de acoplamento dos motores	18
2.3	Soluções da Engenharia Eletrônica	20
2.3.1	Microcontroladores	20
2.3.2	Drivers	20
2.3.3	Câmera	21
2.3.4	Tela Touch Screen	21
2.3.5	Sensores de mapeamento e segurança	21
2.3.6	Diagrama Lógico e de Conexão Eletrônica	21
2.3.7	Comunicação Serial via USB	23
2.3.8	SESC - Sistema Eletrônico de Simulação e Controle	23
2.3.9	Dados recebidos e enviados ao servidor	26
2.3.10	Sistema antitravamento da bola	26
2.3.11	Mapeamento da mesa	27
2.4	Solução de Estrutura	29
2.4.1	Suporte para câmera	30
2.4.2	Sistema dos motores	31
2.4.3	Suporte do display	31
2.4.4	Simulações estruturais	32
2.4.4.1	Mesa Pebolim	32
2.4.4.2	Mecanismo Pebolim	33
2.4.5	Processo de montagem da mesa	35
2.4.5.0.1	Componentes gerais	35
2.4.5.0.2	Procedimento de montagem	38
2.5	Solução de Software	48
2.5.1	Estrutura da Solução de Software	48

2.5.2	Software para Mesa de Pebolim	50
2.5.2.1	PUDM	51
2.5.2.2	Servidor de Decisão	53
2.5.2.2.1	Organização	55
2.5.2.2.2	Interface de Usuário	57
2.5.2.2.3	Restrições	60
2.5.2.2.4	Inteligência Artificial	65
2.5.2.3	Controlador de Máquina	68
2.5.2.3.1	Simulação	69
3	CONSIDERAÇÕES FINAIS	77
	REFERÊNCIAS	78
	APÊNDICES	80
	APÊNDICE A – GERENCIAMENTO GERAL	81
A.1	Análise Quantitativa	81
A.1.1	Matriz de Probabilidade	81
A.1.2	Matriz de Impacto	81
A.1.3	Matriz de Prioridade	81
A.2	Escopo	82
A.2.1	Descrição do Produto	82
A.2.2	Lista É/Não é	82
A.2.3	Premissas e Restrições	82
A.2.4	Termo de Abertura de Projeto	83
A.2.4.1	Introdução	83
A.2.5	Propósito e Justificativa	83
A.2.5.1	Vantagens e desvantagens da substituição do homem por uma máquina	83
A.2.6	Objetivos do Produto	84
A.2.6.1	Objetivo Geral	84
A.2.6.2	Objetivos Específicos	84
A.2.7	Requisitos de Alto Nível	84
A.2.8	Riscos Iniciais	84
A.2.9	Stakeholders	86
A.2.9.1	Equipe do Projeto	86
A.2.9.2	Professores	86
A.2.9.3	Público Alvo	86
A.2.10	Cronograma e Marcos	87

A.3	Gerência	88
A.3.1	Metodologia	88
A.3.2	Microsoft Teams	88
A.3.3	Organização da Equipe	88
A.3.4	Integrantes e Suas Responsabilidades	89
A.3.4.1	Diretores e Técnicos	89
A.3.4.2	Equipes e Responsabilidades	90
A.3.5	Gerenciamento de Comunicação	91
A.3.6	Atas	91
A.3.6.1	Reunião 1	91
A.3.6.2	Reunião 2	92
A.3.6.3	Reunião 3	92
A.3.6.4	Reunião 4	92
A.3.7	Gerenciamento de Qualidade do Projeto	93
A.3.8	Identificação dos Riscos	94
A.3.8.1	Análise SWOT	94
A.3.9	Análise Qualitativa	95
A.3.9.1	Diagrama de Ishikawa	95
A.3.9.2	Descrição do diagrama de Ishikawa	95
A.3.10	Registro dos Riscos	96
A.3.10.1	Risco Gerais	96
A.3.10.2	Resposta aos riscos gerais	97
A.3.10.3	Riscos da Estrutura	98
A.3.10.4	Resposta aos riscos de estruturas	99
A.3.10.5	Riscos Equipe de Motores - Engenharias de Energia e Eletrônica	100
A.3.10.6	Resposta aos Riscos Equipe de Motores	101
A.3.10.7	Riscos da Equipe de Software	102
A.3.10.8	Resposta aos Riscos da Equipe de Software	104
A.4	Tempo	105
A.4.1	Estrutura Analítica do Projeto	105
A.4.2	Cronograma	106
A.5	Requisitos	110
A.6	Requisitos Gerais	110
A.7	Requisitos de motores	110
A.8	Requisitos de Estrutura	111
A.9	Requisitos de Software	111
A.9.1	Técnica de Elicitação	111
A.9.2	Backlog do Produto	112
A.9.2.1	Temáticas	112

A.9.2.2	Features	112
A.9.2.3	Tarefas	113
A.10	Custo	114
	APÊNDICE B – MOTORES/ELETRÔNICA	117
B.1	Cálculos associados aos motores e alimentação	117
B.1.1	Cálculos associados aos motores de rotação	117
B.1.1.1	Dados Iniciais	117
B.1.1.1.1	Equações Utilizadas	117
B.1.1.1.2	Determinando a velocidade final da bola	118
B.1.1.1.3	Determinando o torque	118
B.1.1.2	Cálculos associados aos motores de Deslocamento	119
B.1.1.2.1	Dados Iniciais	119
B.1.1.2.2	Equações Utilizadas	119
B.1.1.2.3	Determinando Aceleração	120
B.1.1.2.4	Determinando Forças	120
B.1.1.2.5	Determinando Torque	120
B.1.1.3	Cálculos associados à Movimentação linear do NEMA 23	121
B.1.1.4	Cálculos associados à alimentação	121
B.2	Características dos dispositivos eletrônicos e energéticos do projeto	122
B.2.1	Elementos em eletrônica	122
B.2.1.1	Microcontrolador principal	122
B.2.1.2	Microcontrolador secundário	123
B.2.1.3	Câmera	124
B.2.1.4	Display	125
B.2.1.5	Driver	126
B.2.2	Alimentação do projeto	127
B.2.2.1	Alimentação dos Motores e Drivers	127
B.2.2.2	Alimentação Microprocessador	128
B.2.3	Motores do projeto	129
B.3	Código Fonte da Solução de Eletrônica	129
	APÊNDICE C – DRAFTINGS	136
	APÊNDICE D – SOFTWARE	167
D.1	Código Fonte da Solução de Sofware	167
D.2	Detalhes da Modelagem PUDM	168
D.3	Distribuição	171

1 Apresentação

1.1 Contextualização

O pebolim foi criado na primeira metade do século XX, tendo chegado ao Brasil na década de 1950, através de imigrantes espanhóis. No começo da década de 60, começou a ser tratado como esporte com federações e regulamentações nacionais, mas, apenas em 2002 foi criada a International Table Soccer Federation (ITSF), a comissão responsável pela unificação das regras para campeonatos internacionais.

Segundo as regras internacionais, o principal objetivo do jogo é que a bola entre no gol adversário, vencendo o jogador que ganhar três de cinco jogos, sendo cada um composto por 5 gols. O último jogo da partida deve ser composto de 5 gols, mas deve ser vencido por 2 gols de vantagem, com um máximo de 8 gols. Na prática do pebolim, os principais componentes que compõe o jogo são a mesa, a bola e as manoplas ([ITSF, 2008](#)).

A chegada de novas tecnologias afeta todas as área da vida moderna, incluindo o esporte e o lazer. Segundo Júnior ([JUNIOR, 2010](#)) , a introdução de aparelhos tecnológicos traz mais racionalidade e precisão aos movimentos. Exemplo disso é o uso de tecnologia para adaptar a altura das tabelas de basquete ou para criar formas de interação entre os praticantes de alguma modalidade. Outro exemplo ocorre no surf através do uso de pranchas com maior flutuabilidade e equipamentos de segurança para iniciantes.

Mesmo com grande popularidade, algumas mesas de pebolim permanecem sem uso. A proposta deste trabalho é utilizar tecnologia moderna para proporcionar a experiência de jogo ao usuário quando um oponente humano não estiver disponível. A mesa contará com um sistema automatizado que moverá independentemente o time adversário e um monitor que mostrará o placar do jogo.

2 Solução

2.1 Solução Geral

A solução para o problema da automatização da mesa de pebolim é feita através de um projeto de mesa de pebolim próprio, partindo do padrão estabelecido e realizando as alterações necessárias para que os componentes dos subsistemas sejam acomodados na própria mesa. A aquisição da imagem é feita com um sistema de visão e a partir do processamento da imagem obtida, a melhor decisão é calculada e transmitida para os motores, que são responsáveis pela movimentação dos eixos da mesa. Para o usuário do produto a experiência de jogo não deve ser diferente do que seria em uma mesa normal.

2.2 Soluções da equipe de motores

2.2.1 Sistema de motores

A solução da equipe de motores partiu dos detalhes base de uma mesa comum de pebolim:

- A mesa possui 4 hastas de cada lado, sendo cada haste acoplada aos jogadores de cada posição de jogo;
- Em um ambiente de jogo, são detectados dois movimentos dos jogadores: Linear e rotacional, consistindo nos movimentos horizontal e de chute, respectivamente.

Dito isto, a solução utilizada consiste na utilização de 2 motores para cada haste, sendo um responsável pelo movimento linear(horizontal), auxiliado por um trilho, e outro responsável pelo movimento rotacional.

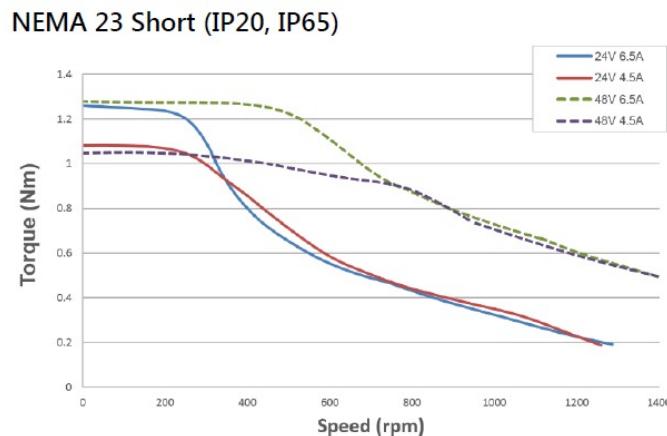
2.2.2 Dimensionamento dos motores

Segundo dimensionamento feito pelo grupo, os motores escolhidos foram: O motor NEMA 23 para o movimento linear das hastas e o motor NEMA 34 para o movimento rotacional. Abaixo, estão as informações relacionadas a cada motor, sendo os dados obtidos pelo *datasheet* apresentado por ([SERVOTRONIX, 2014](#)).

Model	Unit	Value	
NEMA	-	23 Short	34 Long
Input Power, Nominal($\pm 10\%$)	VDC	14-48	14-48
Auxiliary Input Power, Nominal($\pm 10\%$)	VDC	6-24	6-24
Auxiliary Input Power, Maximum	W	1	1
Detent Torque	mNm	40	350
Thrust Load Limit	kg	0.6	3.8
Overhung Load Limit(From shaft end)	N	50	260
Rotor Inertia	$g \cdot cm^2$	260	2750
Holding torque at continuous current	Nm	1.1	5.5
Holding torque at peak current	Nm	1.3	7
Continuous Output Current	A	4.5	7
Peak Output Current(application dependent)	A	6.5	11.5
Step Angle	deg	1.8	1.8
Magnetic Encoder, Resolution	ppr	4096	4096
Circuit Loss	W	6	6
Weight	kg	0.80	4.30
Connection Hardware, Screw size/torque	Nm	3	5.2
Under-Voltage Trip, Nominal	VDC	Logic	Logic
Over-Voltage Trip	VDC	Logic	Logic

Tabela 1 – Motores NEMA 23s e 34s

Levantados os requisitos, concluiu-se que o motor **NEMA 23 em seu modelo Short(IP20) (24 VDC e 4.5A)** é o mais adequado para o deslocamento linear, enquanto o **motor NEMA 34 em seu modelo Long(IP20) (24 VDC e 4.5A)** é o ideal para o movimento rotacional das hastas. Abaixo são apresentados os gráficos Torque x RPM dos dois motores escolhidos, nas figuras 1 e 2 e, logo após, os valores obtidos através dos cálculos do dimensionamento dos motores e da alimentação, encontrados no apêndice B deste documento B.1.

Figura 1 – Gráfico Torque x RPM ([SERVOTRONIX, 2014](#))

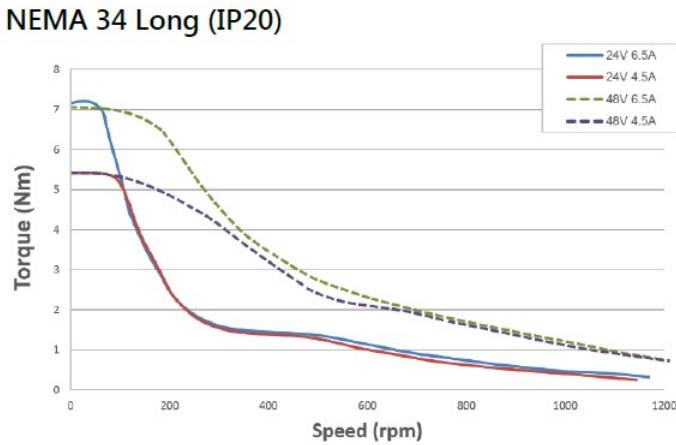


Figura 2 – Gráfico Torque x RPM ([SERVOTRONIX, 2014](#))

2.2.3 Rotação e Movimentação linear

Os resultados abaixo dizem respeito ao motor responsável pelo movimento de chute dos jogadores controlados pelo sistema, NEMA 34.

- Passos necessários para o chute: são necessários 42 passos, pois cada passo do motor é de $1,8^\circ$ e para o chute é necessário um ângulo de 75° ;
- Torque: 1,92 Nm;
- RPM: 350;
- Velocidade da bola: > 5 m/s.

Os resultados a seguir dizem respeito ao motor responsável pelo movimento linear das hastes controladas pelo sistema, NEMA 23.

- Torque: 0,92 Nm;
- RPM: de 90 a 250;
- Velocidade máxima de movimentação das hastes: 1,00 m/s;
- Resolução do motor = 0,53 mm deslocados por passo.

2.2.4 Alimentação do projeto

- A potência máxima demandada na fonte é de 960W em 5A e 24VDC para os drivers;
- A corrente de operação da fonte é de 5A, pois é a capacidade máxima dos drivers;

- Tensão de entrada 110 a 220VAC (bivolt) ; tensão de saída 24VDC para os drivers;
- Tensão de entrada 110 a 220VAC (bivolt) ; tensão de saída 5VDC para o microcontrolador;
- O microprocessador será responsável por alimentar o microcontrolador via USB - Serial.

Uma vez que os motores NEMA operam em 4,5 A, uma fonte comercial de 900W (ou mais) bivolt com saída de 24VDC é ideal. Logo, a fonte utilizada é a **SE-1000-24 da MeanWell (1000W Saída 24V)**, descrita e apresentada no apêndice [B.2.1](#). Ressalta-se que uma vez que a fonte esteja dimensionada de maneira adequada em termos de carga e tensão de saída, a mesma deve ser instalada em um local com boa ventilação, já que o superaquecimento a desligará. Para a alimentação do microprocessador, a **fonte oficial da Raspberry pi 4** é suficiente.

A primeira fonte citada, é uma fonte chaveada, dentro de suas características está a de possibilitar uma melhor conversão sem a necessidade de dispositivos robustos, ou de escolha manual para tensões de entrada em 110Vac ou 220Vac. Já a segunda fonte, é padrão para alimentação do microprocessador e dos periféricos associados.

A alimentação deve ser feita associando as duas partes em uma única fonte de tensão, isso pode ser feito através de uma associação de terminais das fontes, semelhante a um 'T' comumente encontrado em tomadas. A estrutura resultante do acoplamento, já com ambas as partes unidas, deve ser alocada em um local que não tenha riscos de acidentes ao usuário, uma vez que este elemento poderá chegar a elevadas temperaturas.

2.2.5 Sistema de acoplamento dos motores

A mesa de pebolim projetada fará as adaptações apenas para um lado de jogo, de forma que um jogador humano possa jogar contra um jogador robô. Dito isto, o sistema de acoplamento com a estrutura está representado nas figuras [3](#); [4](#) e [5](#):



Figura 3 – Motores de deslocamento linear



Figura 4 – Motores de chute



Figura 5 – Drivers dos motores e placa controladora

Este sistema é constituído pelos seguintes componentes:

- Trilho simples;

- Motor NEMA 23;
- Motor NEMA 34;
- Componente de fim de curso;
- Cremalheira;
- Engrenagem;
- Adaptador de eixo motor/haste;
- Drivers dos motores;
- Fonte bivolt 24V com potência superior a 900W;

2.3 Soluções da Engenharia Eletrônica

As soluções em engenharia eletrônica visaram dimensionar os dispositivos eletrônicos, assim como o funcionamento eletrônico da mesa integrado à software e à estrutura.

2.3.1 Microcontroladores

A mesa contará com 2 microcontroladores: **O microcontrolador principal** e **o microprocessador**. O **microcontrolador principal** que será embarcado à mesa processará as imagens da câmera a 60fps (*frames por segundo*) com a mínima qualidade de 360p (*pixel*) colorida, que serão utilizadas para calcular a posição e a velocidade da bola com a IA (Inteligência Artificial). Além disso, se comunicará via USB com um **microprocessador** para transmitir a movimentação para os drivers, responsáveis por movimentar os motores e coletar informações sobre a posição dos jogadores. Considerando que o sistema de inteligência artificial de software requer um processamento rápido, a opção escolhida para o microcontrolador principal foi a **Raspberry Pi 4**, figura 74, manuseada através do sistema operacional Ubuntu Core e utilizando a biblioteca *pyFirmata*.

O **microprocessador** servirá para intermediar a comunicação entre o microcontrolador principal, motores e mapeamento dos jogadores. Por conta da alta quantidade de motores, utilizaremos o **Arduíno Mega 2560**, que conta com 54 pinos I/O, como microprocessador.

2.3.2 Drivers

Cada um dos 8 drivers deverão fornecer a potência necessária para os 8 motores respectivos do conjunto das hastas, sendo que cada uma das hastas possuirá um par de motor e drive. Um motor servirá para a movimentação lateral dos jogadores e outro

motor para o movimento de chute da bola. Isso demanda uma potência maior do que a que é fornecida pelo microcontrolador, fazendo necessário o uso do driver. Considerando as especificações dos motores NEMA 23 e NEMA 34, o modelo de driver escolhido foi o **WD-TB2404**.

2.3.3 Câmera

A câmera, além de ter que funcionar a 60fps (*frames* por segundo) e 360p (*pixel*), deve possuir abertura angular proporcional às medidas da mesa. Com isso, utilizou-se a **Raspberry Pi 5MP**.

2.3.4 Tela Touch Screen

A tela touch screen deverá conter um menu de informações de configurações do jogo para que o usuário possa selecionar as características do jogo antes de começar a partida. A tela escolhida foi a **IPS Touch Screen Raspberry Pi**, que possui uma resolução de 1024x800 com 10.1 polegada.

2.3.5 Sensores de mapeamento e segurança

Na figura 6, observa-se os sensores de mapeamento, que servem para mapear a posição dos jogadores na mesa e os sensores de segurança que servem para manter a integridade dos motores (M1, M2, M3, M4, M5, M6, M7, M8) e da mesa. 8 sensores de fins de curso mecânico (FC_M min M1, FC_M max M1, FC_M min M2, FC_M max M2, FC_M min M3, FC_M max M3, FC_M min M4, FC_M max M4) serão designados para o eixo de locomoção, eles definirão os pontos de máximo e mínimo que o eixo poderá se locomover, sendo 2 sensores para cada eixo localizados nos trilhos embaixo da mesa. Outros 4 sensores de reflexão de IR (Infravermelho) com uma faixa de sinalização (FC_E M5, FC_E M6, FC_E M7, FC_E M8) no eixo de rotação dos eixos do zagueiro, meio-campo e atacante, mapearão o ângulo que os jogadores estão.

2.3.6 Diagrama Lógico e de Conexão Eletrônica

As portas utilizadas no diagrama da figura 6 são explicadas abaixo:

1. USB (Universal Serial Bus) : É uma tecnologia onde há transmissão e armazenamento de dados e ainda faz conexão de periféricos (aparelhos ou placas que enviam e recebem informação) sem a necessidade de desligar o computador;
2. CSI (Camera Serial Interface): É uma especificação que define uma interface entre uma câmera e um processador host;

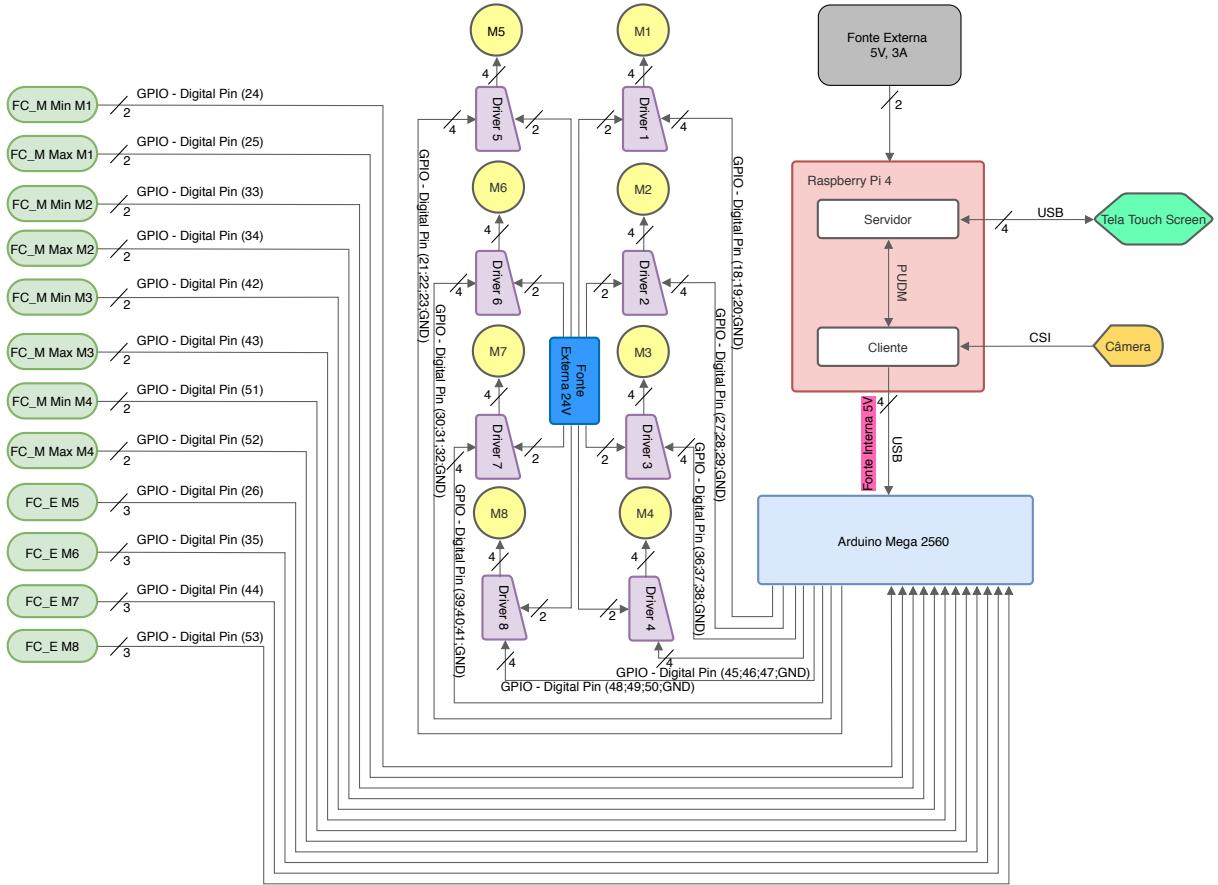


Figura 6 – Diagrama Lógico do Circuito Eletrônico

3. GPIO (General Purpose Input/Output): São portas programáveis de entrada e saída de dados usadas para fazer uma interface entre os periféricos e os microcontroladores.

O diagrama da conexão eletrônica mostra as portas e barramentos dos componentes interligados, bem como sua alimentação. Os fins de curso mecânicos estão nomeados como "FC_M" e os digitais são apresentados como "FC_E". Ambos estão conectados ao Arduíno Mega por meio das portas lógicas (Digital Pin). Os fins de curso mecânicos possuem 2 barramentos, ou seja, dois fios saindo do mesmo, já os eletrônicos têm 3 barramentos. Tais componentes são alimentados pelo Arduíno. Os 8 drivers estão conectados ao Arduíno Mega através de portas digitais e possuem 4 barramentos cada um, no qual estes são alimentados por uma fonte externa de 24 Volts. Um cabo *USB* conecta o Arduíno à Raspberry Pi 4, que o alimenta com 5 Volts. Além disso, Raspberry possui outros dois componentes acoplados ao mesmo: A tela *touch screen* e a câmera, em que o primeiro é conectado também por um cabo *USB* e o segundo por um cabo *flat*.

2.3.7 Comunicação Serial via USB

A Raspberry Pi 4 controla e alimenta o Arduíno via serial *USB*, utilizando a própria linguagem *Python* através da biblioteca *pyFirmata*. Três passos são necessários para que isso seja possível, são eles:

- Primeiro - O usuário deverá conectar a placa Arduino Mega em seu computador e em seguida configurar na *IDE* do Arduino, fazendo o *upload* do *sketch StandardFirmata* para a placa;
- Segundo - Instalar a biblioteca *pyFirmata* executando o comando `$ pip3 install pyfirmata` no terminal da Raspberry;
- Terceiro - O usuário deverá conectar seu Arduino na Raspberry via cabo *USB* e a partir daí o seu código feito em *Python*, utilizando a biblioteca *piFirmata*, irá controlar o Arduino.

2.3.8 SESC - Sistema Eletrônico de Simulação e Controle

A figura 7 mostra o comportamento geral do SESC (Sistema Eletrônico de Simulação e Controle), nela está incluída a integração com a IA, motores e o sistema de validação da eletrônica da mesa. O software foi escrito em *Python* e é composto por 4 arquivos, que serão executado pela Raspberry Pi 4, onde se comunicará com a IA e o Arduíno. Como explicado anteriormente, a Raspberry Pi 4 irá controlar o Arduíno com a biblioteca do *pyFirmata*.

O arquivo *SESCevent.py* é a parte do sistema que é integrada à IA, o qual é responsável por receber e enviar os eventos via PUDM e processar a foto da mesa. O SESC é iniciado por meio do evento *ActionEvent*, onde contém a posição futura de determinado jogador e o comando para chutar ou não. Esse evento é enviado para o arquivo *SESCcontrol.py* que retornará os estados atuais de cada jogador e enviará o evento *StatusUpdateEvent*.

No arquivo *SESCcontrol.py*, é recebido o *ActionEvent* e é verificado se a mesa foi inicializada, caso não, o programa habilita as entradas e saídas do Arduíno, seleciona as configurações iniciais da mesa, coletadas previamente, e envia *RegisterEvent* para a IA. Para iniciar a simulação, dados aleatórios sobre a posição de cada jogador são gerados e depois sincronizados com a mesa física a partir do mapeamento. Depois de inicializada, é preciso mostrar o campo do jogo com os dados iniciados, figura 8.

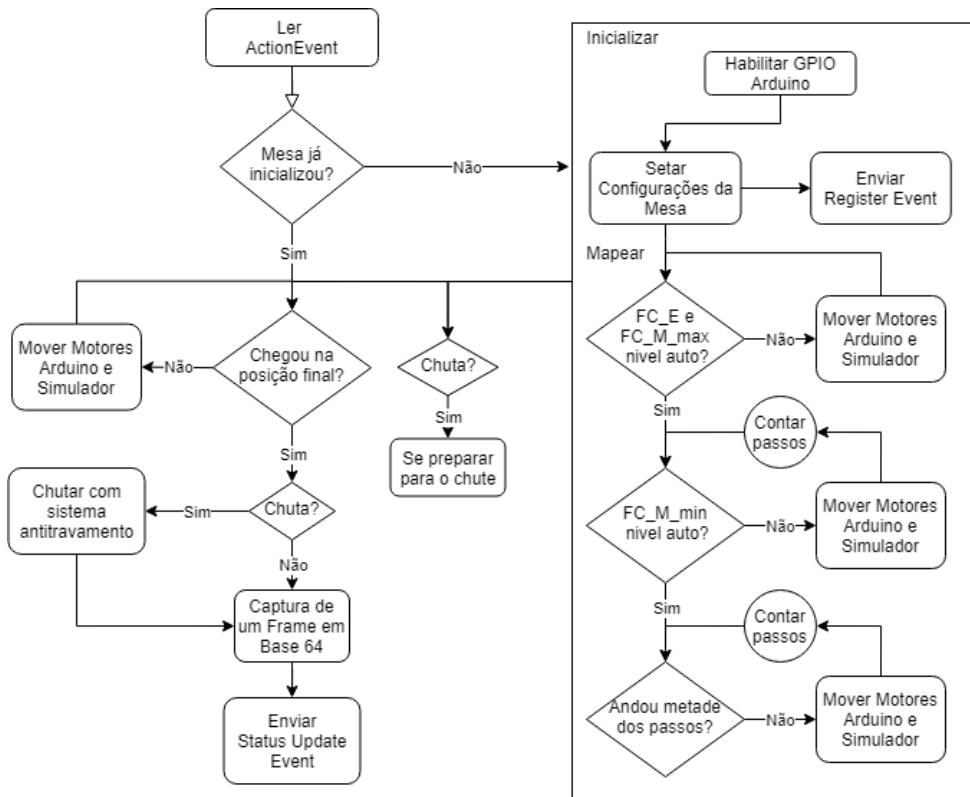


Figura 7 – Fluxograma do SESC

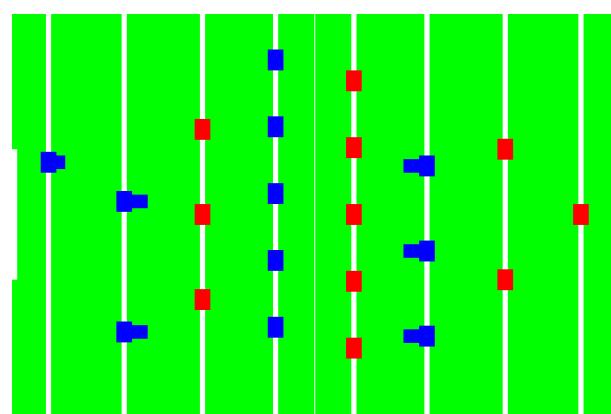


Figura 8 – Iniciando o SESC

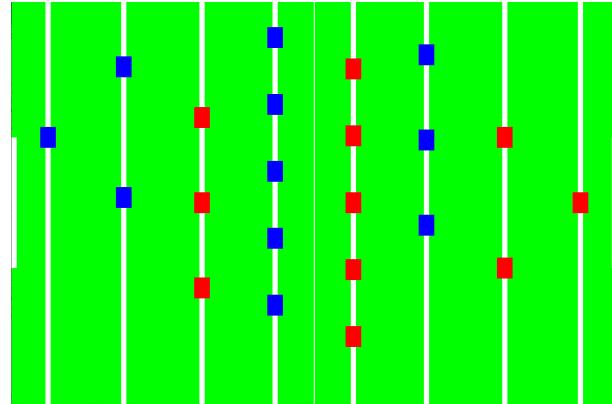


Figura 9 – Mapeamento eletrônico da mesa, ponto máximo

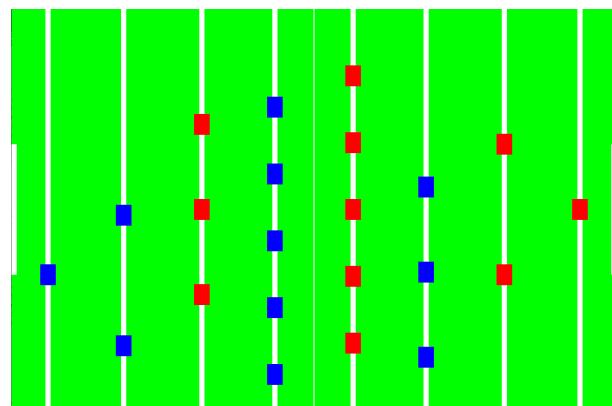


Figura 10 – Mapeamento eletrônico da mesa, ponto mínimo

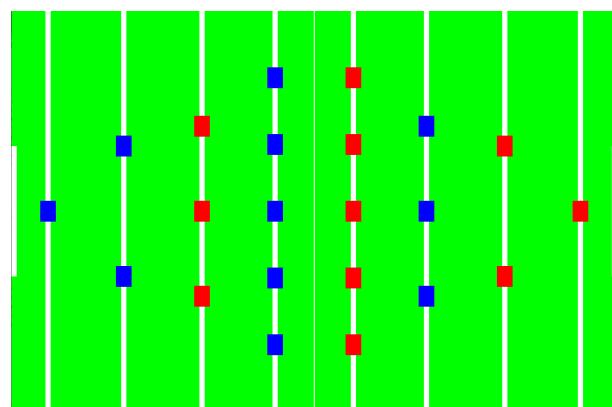


Figura 11 – Centralização eletrônica da mesa

Ainda na inicialização, o mapeamento eletrônico da mesa é feito, e para isso, é necessário fazer com que todos os jogadores estejam em pé e indo para os pontos máximos da mesa, até que os sensores de fim de curso sejam ativados, figura 9. Após isso, o programa coloca todos os jogadores nos pontos mínimos contanto a quantidade de passos que foram dados, figura 10.

Os jogadores são centralizados usando a quantidade de passos que deram, figura 11 e por fim é feito o primeiro evento enviado pela IA. A locomoção dos motores físicos e simulados é realizado no mesmo momento.

O arquivo *SESCdraw.py* é o programa responsável por virtualizar o campo e os jogadores e ele é chamado toda vez que acontece um movimento nos motores. No Apêndice B.3 é possível ver algumas funções chave do SESC comentadas.

Os códigos fonte para a solução de eletrônica é organizado por meio do software de controle de versão Git. Os repositórios Git do projeto são disponibilizado de forma aberta na organização no GitHub¹ criada para este projeto.

Nesta organização o repositório utilizado é:

- **SESC**: Contém a documentação relacionada ao SESC;

2.3.9 Dados recebidos e enviados ao servidor

No inicio de cada partida, todos os jogadores estarão perpendiculares ao campo e centrados, isso será feito a partir do mapeamento pré-configurado pela Raspberry. O SESC (Sistema Eletrônico de Simulação e Controle) coletará e enviará constantemente o estado dos jogadores para a IA (Inteligencia Artificial), ambas estão na Raspberry. Este estado será anexado junto a um *frame* da câmera. Depois de processar essa informação, A IA irá enviar para o cliente um comando do estado futuro de um jogador, para que O SESC, por sua vez, determine uma configuração para que o jogador esteja nesse estado. O estado futuro conterá informações sobre a linha movimentada, movimento feito linearmente (eixo y) e o chute.

- Linha : Esse dado é referente a um dos quatro eixos, que são as hastas;
- Movimentação: A posição final em relação ao ponto zero;
- Chute: Se haverá ou não um chute.

2.3.10 Sistema antitravamento da bola

No projeto foi verificado a necessidade de um sistema para que não haja a condição de travamento da bola embaixo dos pés de algum jogador indeterminadamente, pois poderia gerar um colapso dos componentes eletrônicos e mecânicos da mesa. A solução gerada para esse problema foi: caso o cliente não receba nenhuma informação de detecção da bola após uma fração de segundos, o sistema considerará que a bola está travada e enviará um comando para que as hastas se movam para o lado com maior liberdade de

¹ <https://github.com/pi2-2020-1-pebolim>

movimento e se desloque até seu ponto final. Assim, a bola irá se mover para o lado, liberando o jogador e o sistema. O algoritmo criado para essa solução foi ilustrado abaixo, figura 12.

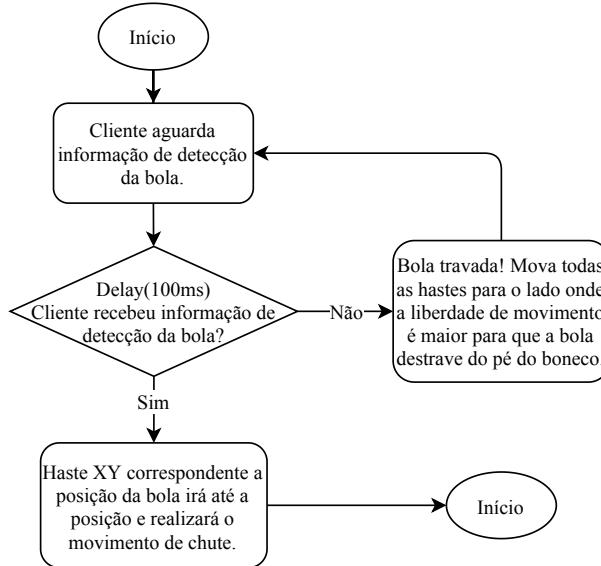


Figura 12 – fluxograma do antitravamento da bola

2.3.11 Mapeamento da mesa

Dimensões da mesa

- Tamanho do campo de jogo : 1160 x 770 mm

Espaçamento entre jogadores

- Defesa : 300 mm
- Meio de campo: 128 mm
- Ataque : 192,5 mm

Espaçamento entre hastes

- Primeira haste para o gol : 700 mm
- Entre hastes: 145 mm

O quanto cada jogador anda para os lados

- Goleiro: 125 mm

- Defesa : 135 mm
- Meio de campo: 60 mm
- Ataque : 120 mm

Para identificação de cada jogador, atribuiu-se um letra e um número, como mostra a tabela abaixo.

ID	Jogador	Posição Fixa (X)	Posição Inicial (Y)	Y max	Y min
A1	Goleiro	70	385	510	260
B2	Defesa	215	535	670	400
B3	Defesa	215	235	370	100
C4	Meio de campo	505	641	701	581
C5	Meio de campo	505	513	573	453
C6	Meio de campo	505	385	445	325
C7	Meio de campo	505	257	317	197
C8	Meio de campo	505	129	189	69
D9	Ataque	795	577,5	697,5	457,5
D10	Ataque	795	385	505	265
D11	Ataque	795	192,5	312,5	72,5

Tabela 2 – Posições e deslocamento linear dos jogadores (mm)

A tabela 2 mostra as posições lineares de cada jogador, sendo uma posição fixa x e uma posição inicial y indicando que os jogadores estão parados. Além disso, há também o Ymax e Ymin, que representam os deslocamentos máximos e mínimos de cada jogador.

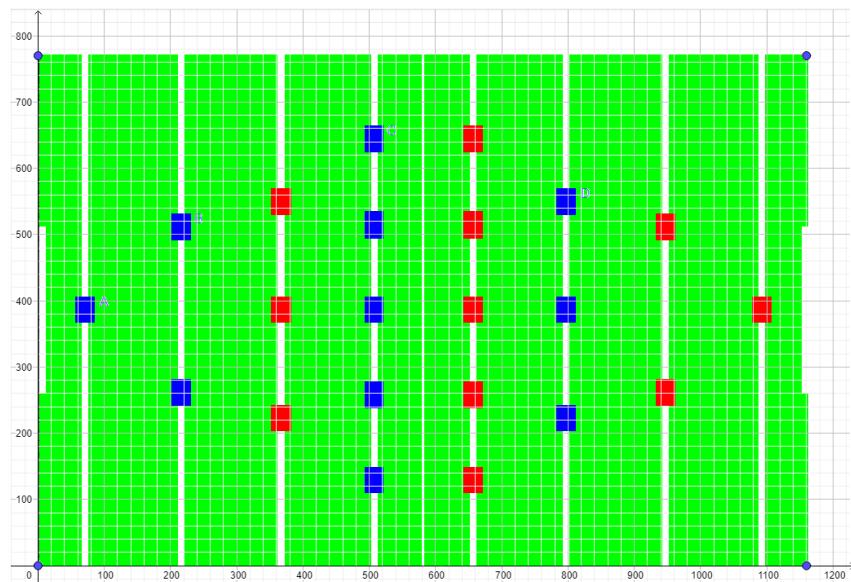


Figura 13 – Campo de jogo no plano cartesiano (mm)

A figura 13 mostra como o campo foi mapeado em centímetros e como é feito o reconhecimento da posição de cada jogador controlado pelo sistema.

2.4 Solução de Estrutura

Desde o estágio anterior a arquitetura básica do projeto se manteve a mesma, porém algumas mudanças pontuais ocorreram. Tais mudanças de design foram motivadas pela otimização e detalhamento, natural do carácter recursivo e iterativo do ciclo de vida do desenvolvimento do projeto. Simulações estruturais foram realizadas como parte do processo de verificação e validação das escolhas de solução para o design. A configuração atual pode ser vista na figura 14.



Figura 14 – Configuração atual da mesa

O design teve inspiração em mesas comerciais. A mesa tem cerca de 1955 mm de comprimento, 1567 mm de altura e 1167 mm de largura. O tablado do campo tem cerca de 1160 mm de comprimento por 770 mm de largura. O campo pode ser visto na figura 15. Os tablados laterais da mesa são feitos de madeira, assim como o campo. Os pés da mesa possuem uma sapata rosada regulável, feita de borracha, e pesos, feitos de aço, para garantir estabilidade.

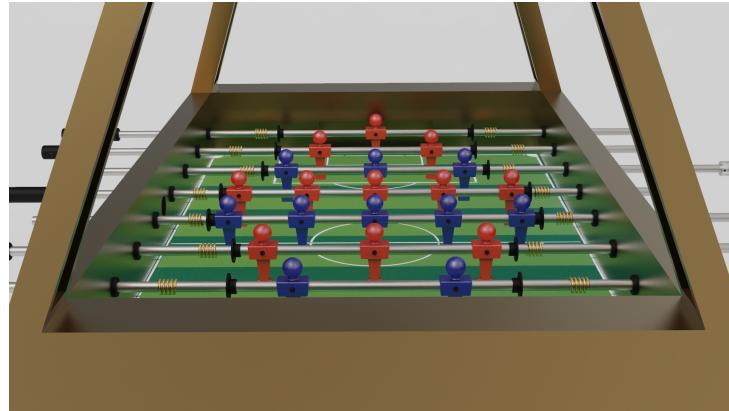


Figura 15 – Campo

2.4.1 Suporte para câmera

Para o posicionamento adequado da câmera sobre a superfície de jogo, um suporte para a câmera foi projetado com fibra de vidro e com uma altura predefinida. O suporte tem cerca de 1353 mm de comprimento e 866 mm de largura.

Uma tampa de acrílico é montada na parte inferior do suporte, de modo a proteger a câmera e a eletrônica associada, mas com uma abertura para que a lente da câmera capture o campo. Para garantir que a altura escolhida é suficiente para que a câmera consiga pegar todo o campo, é necessário saber o ângulo de abertura da câmera escolhida e relacioná-lo com o comprimento do campo.

Essa altura mínima é determinada através da seguinte relação:

$$H = \frac{CO}{\tan(\alpha/2)}$$

Onde:

α = ângulo de abertura da câmera

CO = Metade do comprimento do campo

H = Altura desejada

Sabendo que o comprimento do campo é igual a 1160 mm e o ângulo de abertura da câmera escolhida é de 150°, a altura mínima necessária é 15,58 cm. A altura do suporte é de cerca de 70 cm, mais que suficiente para que a câmera consiga capturar o campo todo.

A figura 16 mostra como a solução foi apresentada.



Figura 16 – Apoio da câmera, esboço

2.4.2 Sistema dos motores

Para a movimentação dos eixos da mesa, os motores devem ser alocados de forma que consigam realizar os movimentos longitudinal e rotacional. Para a movimentação linear foi pensado na fixação de trilhos e de motores na superfície inferior da mesa, conforme visto na figura 3. Para a movimentação rotacional foi pensado no acoplamento de motores nos próprios eixos da mesa, conforme visto na figura 4.

O sistema de motores conta com os seguintes componentes:

Engrenagem de eixo	Raio menor = 13.5 mm e raio maior = 18 mm
Suporte para o motor	Peça de encaixe quadrada de 61x71 mm
Trilho fixo	comprimento de 82.55mm
Trilho móvel	comprimento de 556.22 mm
Cremalheira	comprimento de 439 mm e 3 parafusos de 2,5mm
Treliça	representação diagonal com 251.47mm de comprimento
Manopla	30 mm de raio maior para encaixe da mão

Tabela 3 – Detalhamento dos componentes

Seu funcionamento se baseia na informação recebida pelo sistema de software/eletrônica, podendo movimentar os trilhos para o deslocamento linear dos jogadores ou podendo apenas criar um torque no eixo com os motores de chute. No caso do deslocamento linear, os trilhos acompanharão o movimento assistido pela engrenagem e cremalheira. já no caso do movimento rotacional, os motores de chute funcionarão em função do giro no eixo dos jogadores, não movimentando o sistema de trilhos.

2.4.3 Suporte do display

O uso do display é fundamental para o acesso às configurações de jogo, e para garantir que a tela fosse acessível para ambos os lados, caso optem pelo jogo humano x

humano, foi criado um suporte rotacional visto pelos dois lados e suspenso sobre o campo, montado no suporte para a câmera, como mostrado na figura 17



Figura 17 – Suporte do display

2.4.4 Simulações estruturais

2.4.4.1 Mesa Pebolim

A mesa de pebolim desenvolvida sob moldes especificados para jogo é sujeita a vibrações contínuas em condições de jogo. Estas, se avaliadas conforme as frequências naturais da mesa, não podem entrar em sobreposição a fim de evitar efeitos de ressonância que deformariam a ponto de danificar sua estrutura. Desta forma, foram obtidas as frequências e avaliado o comportamento da mesa através da análise modal, desta forma, observando a Tabela 4, as três primeiras frequências baixas caracterizam os movimentos de corpo rígido, dois de translação e um de rotação, o que evidencia a condição da mesa apoiada no solo, enquanto que a partir destas, as frequências já caracterizam deformações estimadas da estrutura sob ressonância e que devem ser evitadas. A Figura 18 apresenta o valor mais baixo de frequência que deve ser evitado no mecanismo, e assim considerado como limitante, uma vez que, sob condições adequadas de uso, as fontes de excitação fornecem frequências bem abaixo da faixa de 50 Hz, principalmente devido à proporção da mesa e às baixas frequências de excitação dos motores e das movimentações de mecanismos durante o jogo seja por humano ou por máquina.

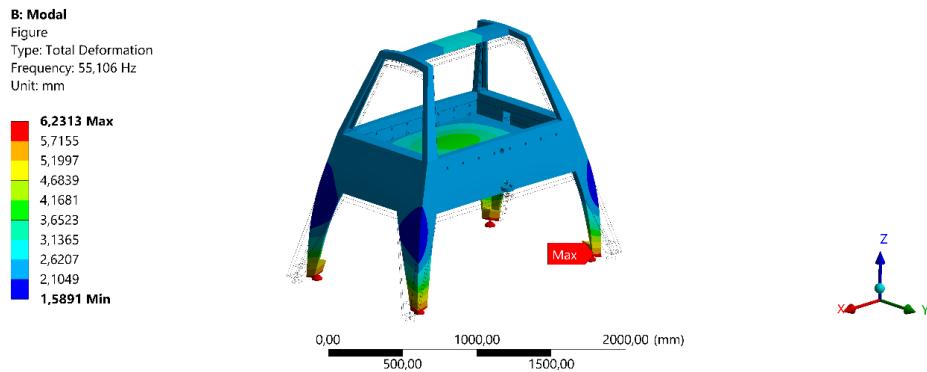


Figura 18 – Mesa Deformada

Modo	Frequência (Hz)
1	1,8294e-004
2	3,2124e-004
3	3,553
4	55,106
5	60,197
6	76,044

Tabela 4 – Frequências modais

2.4.4.2 Mecanismo Pebolim

A resistência dos mecanismos automatizados do pebolim é dimensionada conforme suas condições críticas de operação, que, apesar de prevenidas via software, são consideradas para caracterização da robustez do sistema.

Uma vez com os dois motores em funcionamento atuando com a intensidade máxima de seus respectivos torques, o travamento repentino do boneco nas condições especificadas na Figura 19 caracteriza a situação mais crítica.

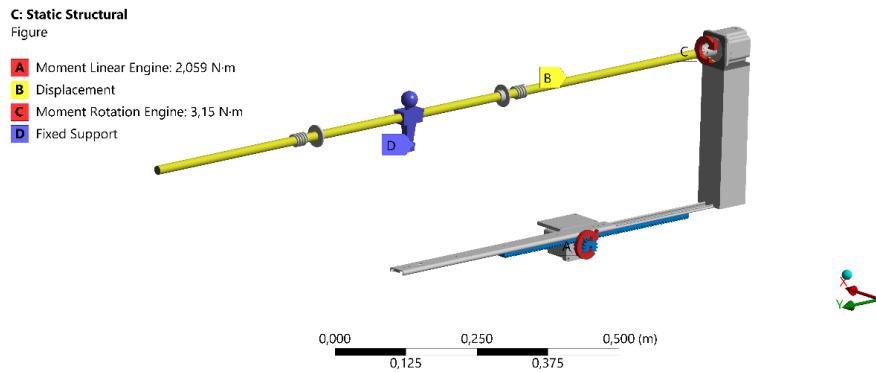


Figura 19 – Condições de contorno

Tal modelagem é capaz de fornecer o resultado nas figuras 20 e 21 evidenciando que a tensão máxima conforme critério de von-Mises se encontra nos contatos entre os dentes da engrenagem e cremalheira, e que esta é inferior à tensão de escoamento de seu material correspondente, o Nylon-6 (38 ± 1 MPa), caracterizando um fator de segurança satisfatório de 8,5.

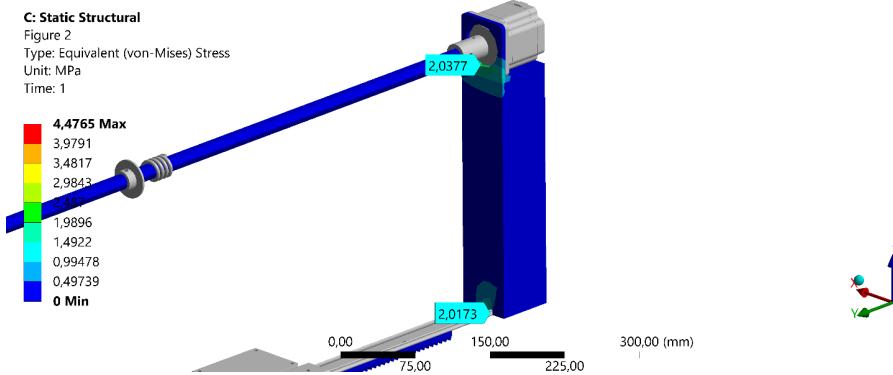


Figura 20 – Tensão na Treliça

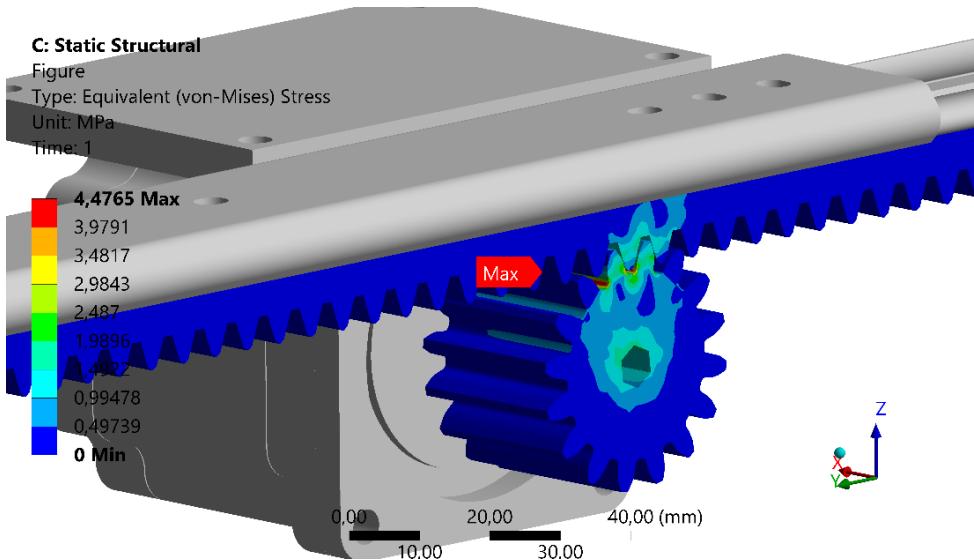


Figura 21 – Ponto de máxima tensão von mises

Ainda evidenciado o efeito dessas condições no suporte do motor em alumínio, nota-se o superdimensionamento da geometria, sendo assim proposto um modelo otimizado de modo a reduzir aproximadamente 60% da massa da estrutura e ainda assim manter a rigidez e a integridade do mecanismo nas condições críticas especificadas. A Figura 22 reflete os resultados da topologia aliviada conforme o método de elementos finitos, uma vez que os elementos de regiões consideradas desnecessárias são removidos da estrutura. Para melhor agrado visual e adequação eficiente de parafusos que serão acoplados ao suporte.

dos à peça, a nova modelagem do sistema baseou-se nos resultados reduzindo os alívios sugeridos em locais estratégicos.

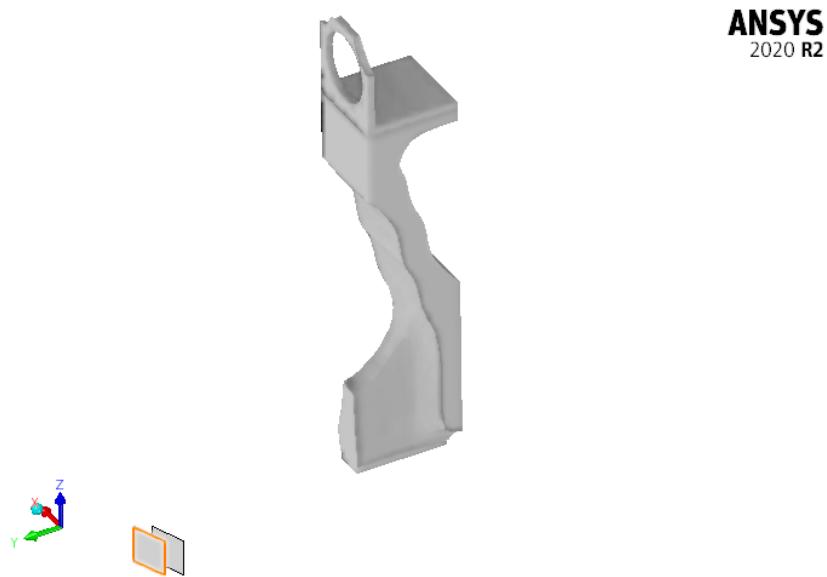


Figura 22 – Otimização

2.4.5 Processo de montagem da mesa

O processo de montagem da mesa será realizado apenas uma vez na compra do produto. Para a montagem, certifique-se que todos os componentes abaixo estejam presentes na sua mesa:

2.4.5.0.1 Componentes gerais

- **Partes distintas:** 78
- **Partes total:** 562

Nº	Nome	Material	Qtd.
1	Tablado do campo	Madeira	1
2	Adesivo	Plástico	1
3	Tablado superior	Madeira	1
4	Rampas guias para bola	Madeira	1
5	Tablado frontal	Madeira	1
6	Tablado esquerdo	Madeira	1
7	Tablado traseiro	Madeira	1
8	Tablado direito	Madeira	1
9	Bucha do eixo	Nilon	1
10	Tablado do fundo	Madeira	16
11	Suportes da cobertura	Fiba de vidro	1
12	Tampa da cobertura	Acrílico	1
13	Peso pé direito	Aço SAE 1008	2
14	Sapata do pé roscada	Borracha	4
15	Peso pé esquerdo	Aço SAE 1008	2
16	Microcontrolador Arduino Mega	OEM	1
17	Parafuso M3X6	Aço parafuso	16
18	Corpo led driver motor de passo	OEM	8
19	Placa de montagem driver motor de passo	OEM	8
20	Chave driver motor de passo	OEM	8
21	Terminal euro 4 pinos driver motor de passo	OEM	8
22	Carcaça de metal driver motor de passo	OEM	8
23	Terminal euro 6 pinos driver motor de passo	OEM	8
24	Terminal euro 2 pinos driver motor de passo	OEM	8
25	Bloco câmera	OEM	1
26	Placa de circuito impresso	OEM	1
27	Resistor Câmera	OEM	4
28	Bloco pinos câmera	OEM	1
29	Presilha do bloco câmera	OEM	1
30	Lente câmera	OEM	1
31	Saída câmera	OEM	1
32	Display 10.1" Raspberry Pi	OEM	1
33	Raspberry Pi 4 Model B	OEM	1
34	Módulo display Raspberry Pi	OEM	1
35	Porca M3X12	Aço parafuso	4
36	Parafuso M2,5X3	Aço parafuso	4

37	Suporte display	PLA/ABS	1
38	Engrenagem eixo motor	Aço SAE 1020	4
39	Suporte motor de passo NEMA 23	Aço SAE 1008	4
40	Parafuso Allen M5X16	Aço parafuso	16
41	Parafuso Philips M5X16	Aço parafuso	32
42	Motor de passo NEMA 23	OEM	4
43	Sensor de fim de curso	OEM	8
44	Porca M5	Aço parafuso	16
45	Parafuso Fenda M3X14	Aço parafuso	8
46	Parafuso Fenda M3X16	Aço parafuso	8
47	Parafuso Fenda M4X16	Aço parafuso	17
48	Parafuso Fenda M3X8	Aço parafuso	4
49	Fonte de energia	OEM	1
50	Suporte da fonte	Aço SAE 1008	4
51	Parafuso Fenda M3.5X5	Aço parafuso	8
52	Parafuso sextavado M6X12	Aço parafuso	4
53	Trilho deslizante móvel	OEM	4
54	Trilho deslizante rolamento	OEM	4
55	Parafuso Allen cabeça chata M6X25	Aço parafuso	34
56	Parafuso Allen M10X30	Aço parafuso	8
57	Parafuso Fenda M6X16	Aço parafuso	10
58	Parafuso Fenda M5X12	Aço parafuso	16
59	Eixo dos jogadores	Aço SAE 1020	8
60	Boneco jogador	Plástico	22
61	Bucha do eixo móvel	Náilon	16
62	Mola do eixo	Aço mola	16
63	Parafuso Allen M4x8	Aço parafuso	44
64	Parafuso Fenda M6X20	Aço parafuso	16
65	Trilho deslizante móvel	OEM	4
66	Cremalheira	Aço SAE 1020	4
67	Parafuso Allen M5X6	Aço parafuso	16
68	Acoplamento flexível motor/eixo	OEM	4
69	Parafuso Allen sem cabeça M4X24	Aço parafuso	4
70	Parafuso Allen M6X16	Aço parafuso	8
71	Estrutura móvel do suporte do motor	Alumínio SAE 6061 - T6	4
72	Motor NEMA 34	OEM	4
73	Suporte motor NEMA 34	OEM	4
74	Parafuso Allen M6X25	Aço parafuso	16
75	Porca M6	Aço parafuso	16
76	Parafuso Allen sem cabeça M5X25	Aço parafuso	4
77	Manopla	Plástico	4
78	Parafuso Fenda M6X25	Aço parafuso	4

Tabela 5 – Tabela de componentes

2.4.5.0.2 Procedimento de montagem

É importante ressaltar a necessidade de seguir os comandos na mesma ordem descrita abaixo para não haver problemas de montagem ou afins. Para o passo a passo de montagem, identificaremos os itens referentes a cada etapa com a tabela acima através do uso de () e o número identificado.

No início da montagem, utiliza-se o tablado superior (3) como base para o encaixe dos componentes estruturais, figura 23. A partir dele, deve-se encaixar o tablado do campo (1), que é o campo de jogo, com o adesivo (2). Logo após, as rampas guias para a bola (4), que serve como suporte para o retorno da bola após o gol, deve ser acoplado com o mesmo adesivo plástico, figura 24. Feito isto, agora finaliza-se a base da mesa parafusando o tablado frontal (5), tablado esquerdo (6), tablado traseiro (7) e tablado direito (8) na base já estruturada da mesa, utilizando 10 parafusos Allen cabeça chata M6X25 (39) em cada lateral, totalizando 20 parafusos, figura 25. Após isso, a base completa da mesa está montada, necessitando agora acoplar o restante dos componentes estruturais adicionais.

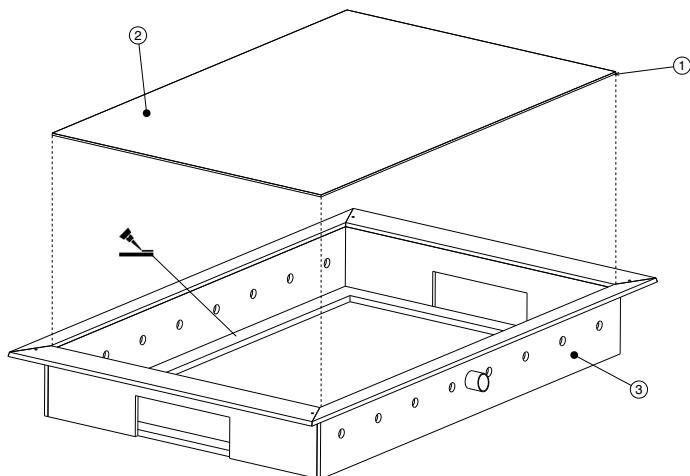


Figura 23 – Passo 1

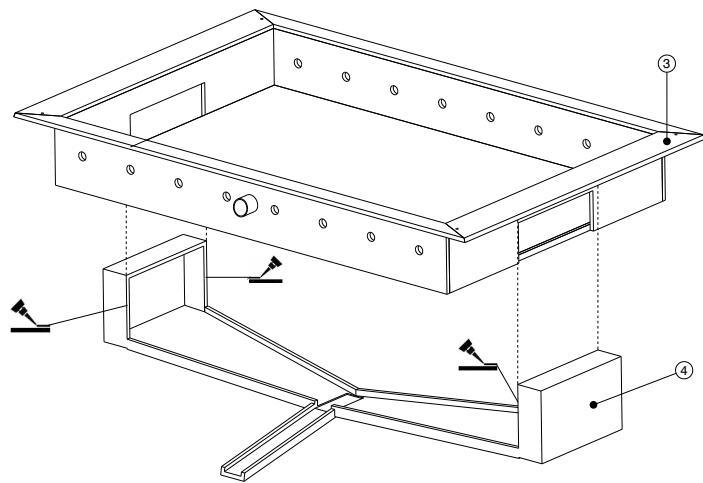


Figura 24 – Passo 2

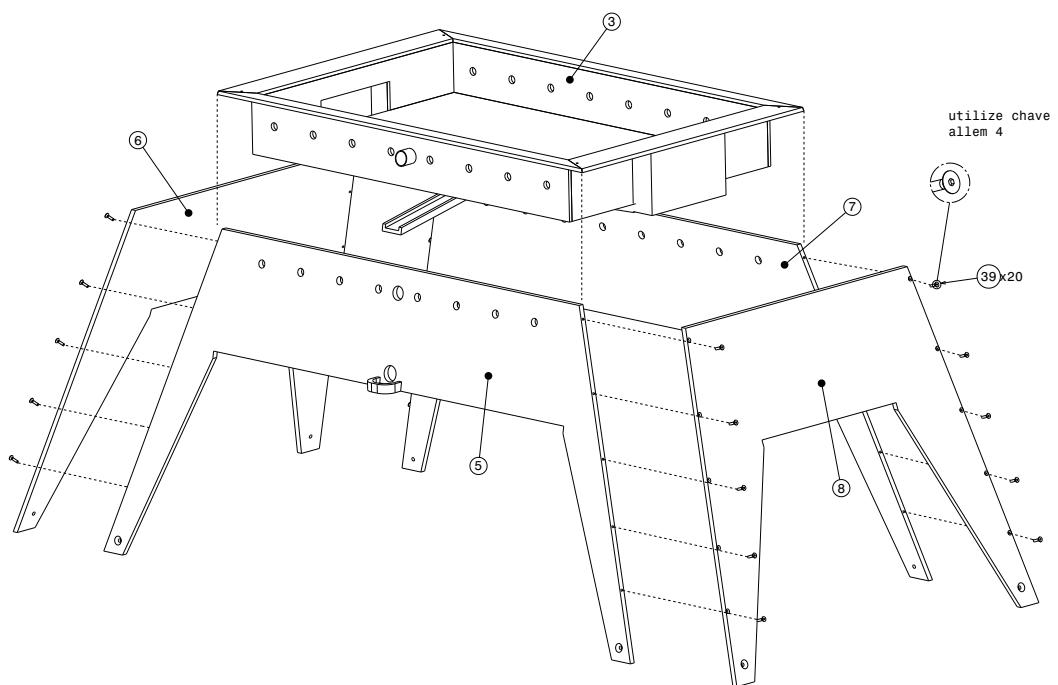


Figura 25 – Passo 3

Com a base estruturada, É necessário encaixar as buchas de eixo (9) para o aco-

plamento futuro das hastas dos jogadores, figura 26. Após isso, podem ser colocados os 4 pés de nivelamento, peça já montada pelos itens (13,14 e 15), figura 27. É importante ressaltar que os pés de nivelamento são diferentes para ambos os lados da mesa, logo os pés direitos não podem ser encaixados no lado esquerdo e vice-versa. Os 4 pés utilizam 2 parafusos Allen M10X30 (40) cada, totalizando 8 parafusos no total.

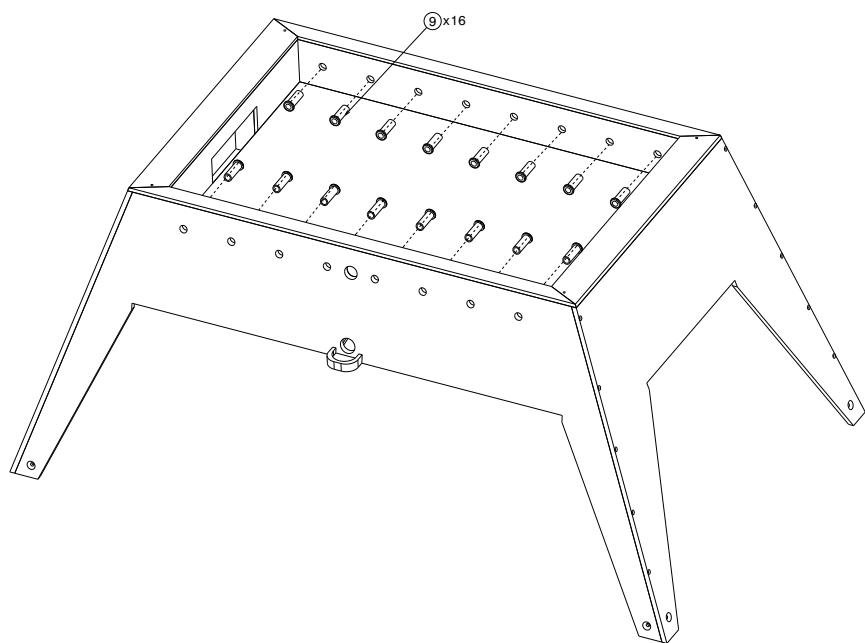


Figura 26 – Passo 4

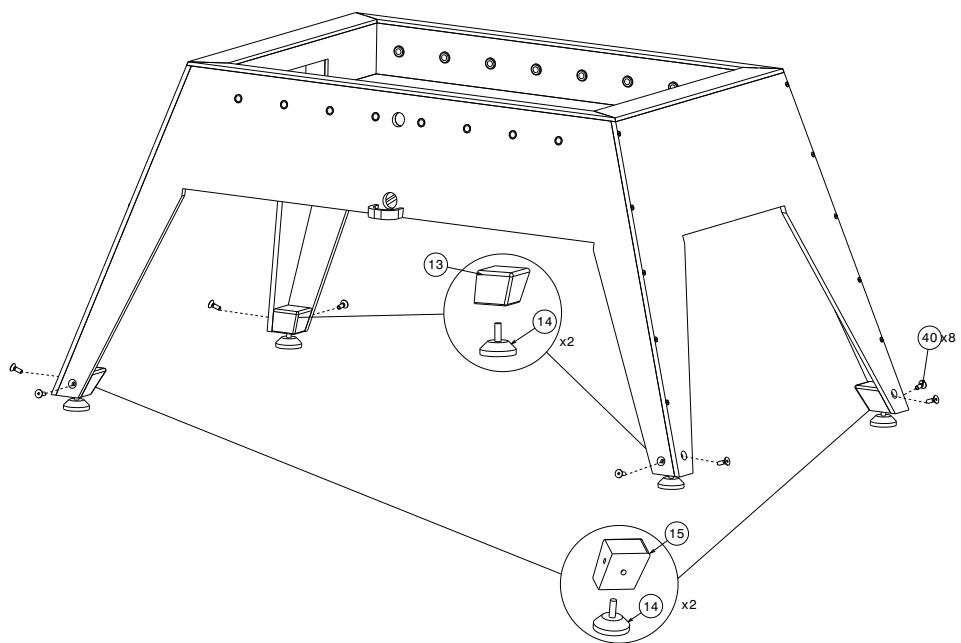


Figura 27 – Passo 5

O próximo passo é encaixar as partes superiores da estrutura, começando pelo suporte da cobertura (11), que é fixado na parte de cima do tablado superior (3) e parafusado com 4 parafusos Allen cabeça chata M6X25 (39), figura 28. Terminado, encaixe a câmera (20) e a tela(19),finalizando com a tampa da cobertura (12) parafusada com 10 parafusos Fenda M6X16 (41), figura 29.

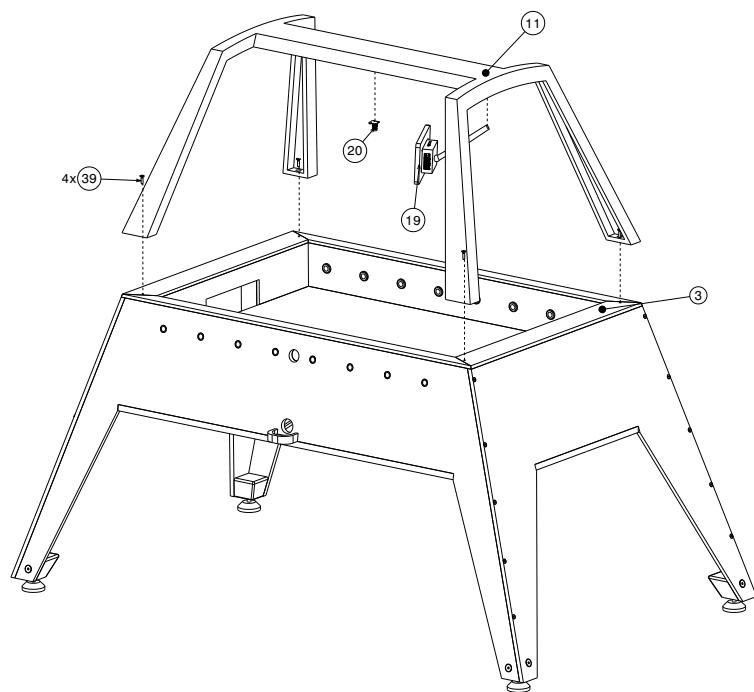


Figura 28 – Passo 6

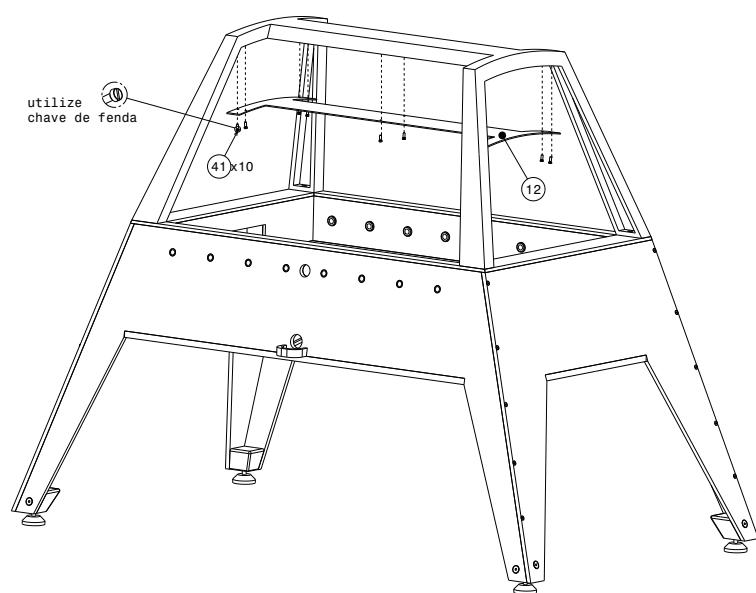


Figura 29 – Passo 7

Indo agora para a parte de baixo, ainda aberta, da mesa, os 8 drivers (18) vão

com 2 parafusos Fenda M4X16 (32) cada, totalizando 16 parafusos, figura 30. Ainda na parte de baixo, teremos o arduino (16) parafusado com 4 parafusos Fenda M3X8 (33). Para fechar, utiliza-se o tablado do fundo (10) parafusado com 10 parafusos Allen M6X25 (57) para encaixar com as laterais da mesa, figura 31.

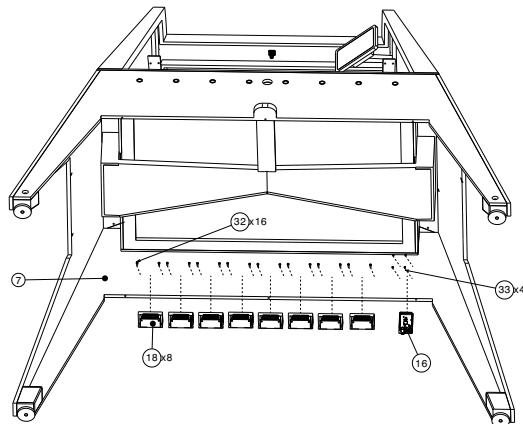


Figura 30 – Passo 8

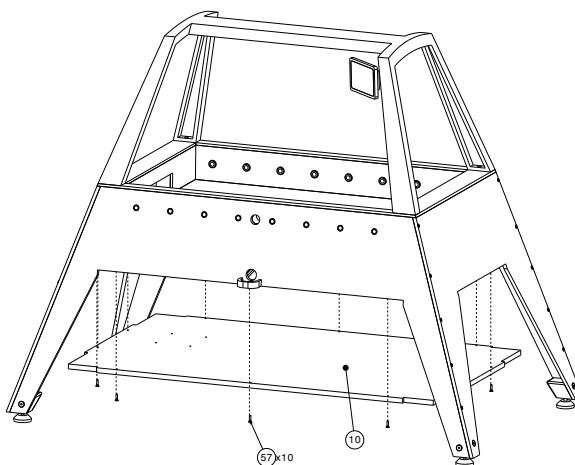


Figura 31 – Passo 9

ATENÇÃO: Preste atenção às pontas das hastes, um dos lados é de encaixe da manopla e o outro o lado de encaixe do suporte do motor de rotação. O lado de encaixe da manopla deve estar no lado **contrário** ao lado em que foram parafusados os motores de eixo, não importando qual lado seja.

Vamos agora trabalhar no campo dos jogadores e no motor de rotação das hastes. Já colocadas as buchas de eixo, encaixa-se o eixo dos jogadores **até a metade do caminho**, coloca-se uma mola (46) e uma bucha de eixo móvel (45), sendo a bucha de eixo móvel parafusada com um parafuso Fenda M6X20 (48) em cada uma. Antes de passar a haste, agora serão colocados a quantidade de jogadores respectivos de cada haste, sendo 1 goleiro, 2 zagueiros, 5 meios de campo e 3 atacantes, e cada jogador parafusado com 2 parafusos Allen M4X8 (47). Depois de colocados os jogadores, parafusa-se uma bucha de eixo e depois encaixa-se uma mola, nesta ordem, podendo agora passar a haste pelo restante da mesa, figura 32.

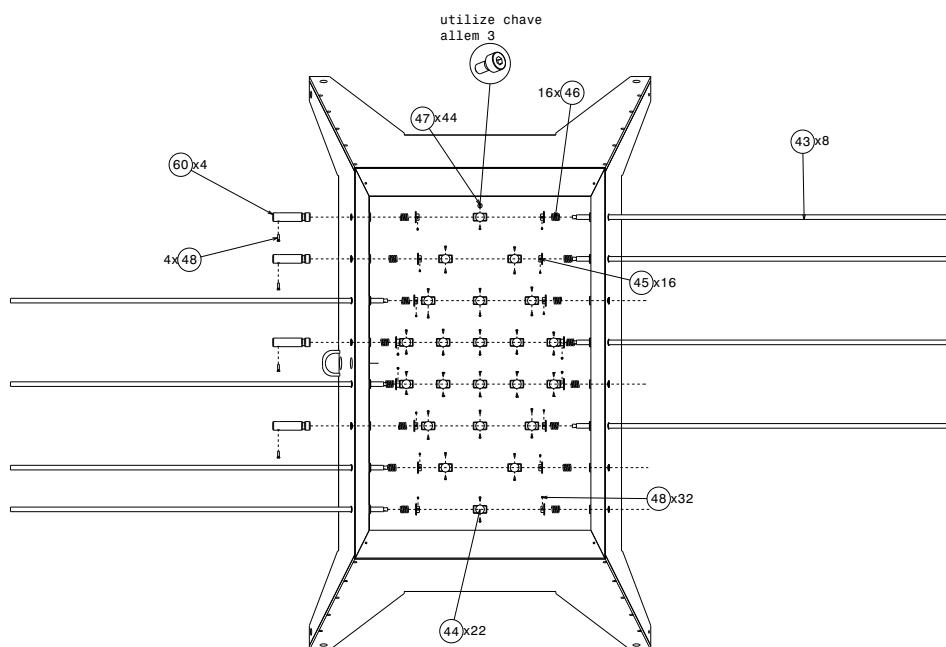


Figura 32 – Passo 10

ATENÇÃO: Não passe a haste antes de passar a mola, pois a mola deve estar no lado de dentro de campo. Além disso, parafuse os jogadores corretamente para que eles fiquem fixos e rotacionem apenas com o giro da haste.

Do lado de fora do campo, no lado do jogador humano (lado contrário ao lado dos motores parafusados), coloca-se as manoplas (60) com 1 parafuso fenda M6X20 (48) para cada manopla, figura 32.

No lado do jogador robô, primeiro montaremos o suporte do motor e depois encaixaremos à estrutura. Com a estrutura móvel de suporte do motor em mãos, parafusa-se o suporte do motor NEMA 34 (56) usando 4 parafusos Fenda M3.5x5(36) para cada suporte. Depois disso, parafusa-se o motor NEMA 34 (55) usando 4 parafusos Allen M6X25(57) e

uma porca M6 (58) para cada parafuso. Para fixar o conjunto na estrutura, parafusa-se a parte de baixo da treliça no trilho deslizante móvel (38) com 2 parafusos Allen M6X16 (53). Por fim, o acoplamento (51) será usado como intermédio entre o motor, já encaixado no suporte, e as hastes do jogador robô. Para encaixar o motor no acoplamento, usa-se 1 parafuso Allen sem cabeça M4X24 (52) para cada motor e para encaixar o acoplamento na ponta da haste, usa-se 1 parafuso Allen sem cabeça M5X25(59), figura 33.

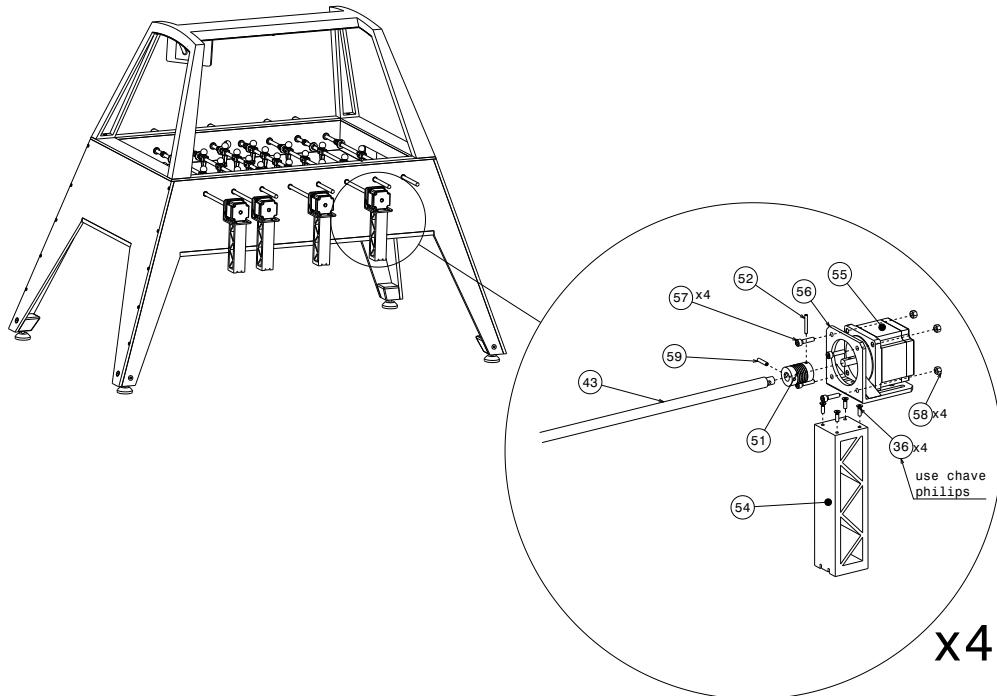


Figura 33 – Passo 11

Com a mesa fechada e os componentes eletrônicos internos já instalados, iniciaremos o processo de montagem dos motores de eixo, motor localizado na parte de baixo, que é o motor de deslocamento horizontal. Parafusa-se os trilhos fixos (38) no tablado fundo (10) com 4 parafusos Fenda M5x12 (42) em cada trilho, sendo 4 trilhos e 16 parafusos no total. Parafusa-se os 4 motores NEMA 23 (27) e os fixa nos suportes do motor (24), sendo necessários 4 parafusos Allen M5X16 (25) e 4 porcas M5 (29) para cada motor, figura 34. Ainda nos motores de baixo, são utilizados 2 sensores de fim de curso (28) em cada motor, sendo parafusados com 2 parafusos M3X14 (30) cada, sendo 16 no total. Terminados todos os passos, parafusa todo o suporte com 16 parafusos Philips M5X16 (26) na parte de baixo da mesa, tablado fundo (10), figura 35. Para finalizar os procedimentos na parte de baixo da mesa, a fonte de energia (34) e o suporte da fonte (35) serão parafusados um no outros com 8 parafusos Fenda M3.5x5 (36), e o conjunto parafusa-se com 4 parafusos sextavado M6X12 (37) no tablado fundo (10), figura 36.

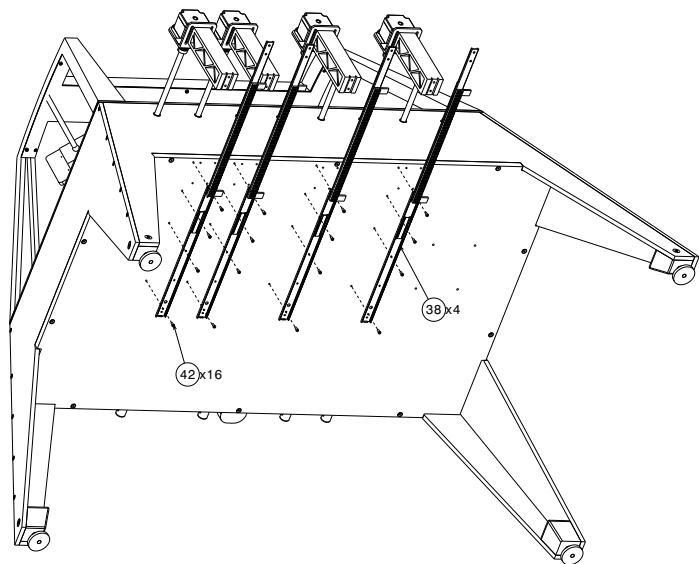


Figura 34 – Passo 12

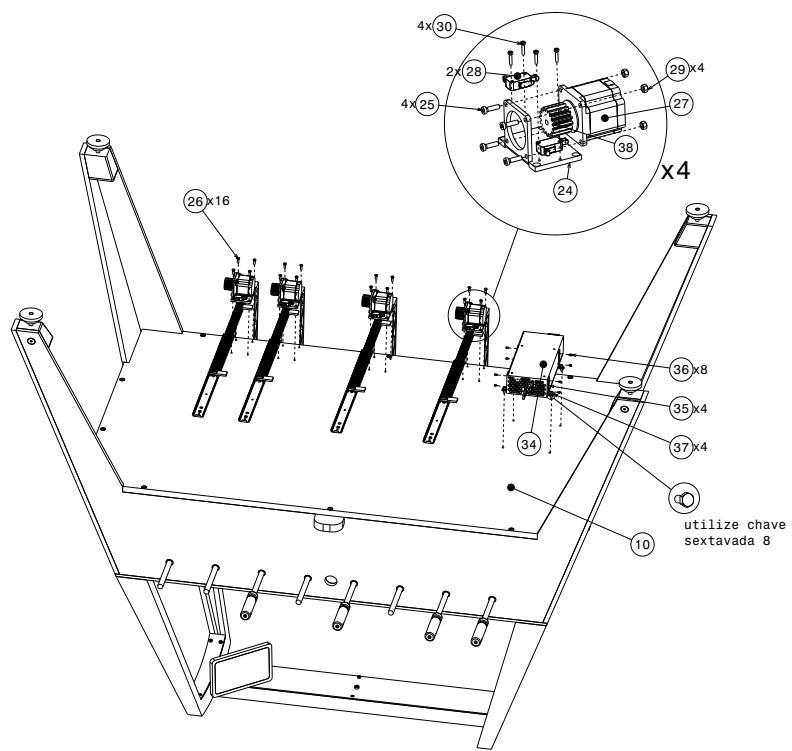


Figura 35 – Passo 13

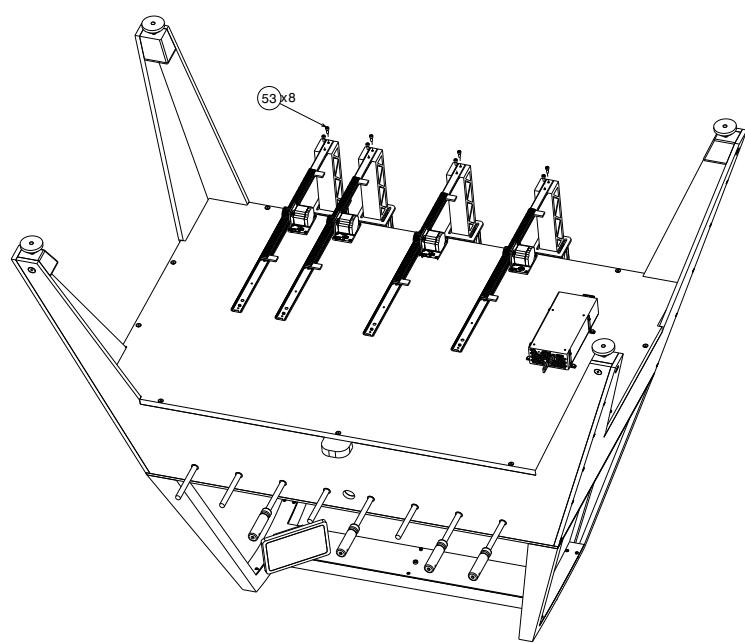


Figura 36 – Passo 14

2.5 Solução de Software

2.5.1 Estrutura da Solução de Software

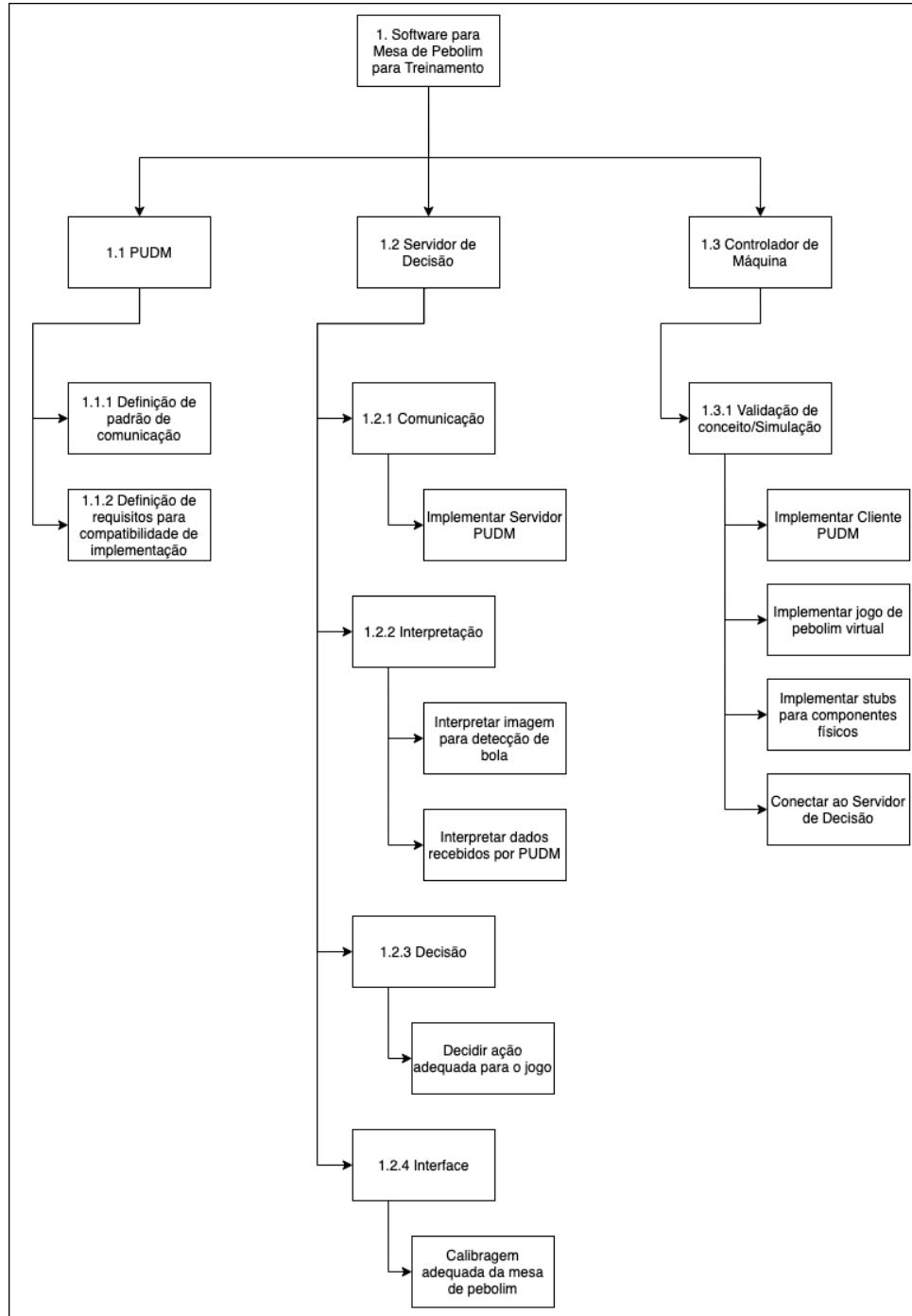


Figura 37 – Estrutura da Solução de Software

Conforme a figura 37, a estrutura da solução de Software para a mesa de pebolim possui três principais componentes: o PUDM, modelagem de dados criada especificamente para o projeto; o Servidor de Decisão, responsável por interpretar os dados recebidos, processá-los e decidir qual a ação a ser tomada; e o Controlador da Máquina, responsável

por obter os dados da câmera e enviá-los para o Servidor de Decisão, bem como receber os dados do Servidor de Decisão para tomar a ação designada por ele.

Em termos gerais, a proposta de solução de software é a construção de um software nominado de Servidor de Decisão que recebe dados da mesa de pebolim e realiza as operações necessárias para gerar uma ação adequada para o estado do jogo de pebolim. Para tanto, tal proposta envolve a utilização de tecnologias de processamento de imagens e de inteligência artificial.

O software desenvolvido deverá ser executado no microprocessador embarcado na mesa de pebolim, conforme mostrado na Seção 2.3.1, fornecendo os controles para o usuário por meio de uma tela acoplada à mesa, conforme mostrado na Seção ??.

Por conta do contexto da pandemia da Covid-19 e a necessidade de realizar o projeto remotamente, não foi possível realizar a construção do protótipo físico da mesa de pebolim com os seus componentes instalados. Desta forma, foi necessária a criação de uma simulação de uma mesa de pebolim através de uma chamada *Engine*, utilizada para a criação de jogos eletrônicos, para que pudéssemos fazer o teste da solução de inteligência artificial.

A *Engine* de jogos é um software ou conjunto de bibliotecas feito para facilitar o desenvolvimento de jogos de computador. Esta opção foi escolhida por conta das facilidades relacionadas à simulação de objetos com características físicas e também pela possibilidade de renderizar a mesa em tempo real em uma imagem, permitindo uma integração com o Servidor de Decisão. Isso será utilizado para auxiliar no desenvolvimento da inteligência artificial e para, no final, validá-la.

O Servidor de Decisão, software responsável pela tomada de decisões, é desacoplado do sistema de gerenciamento dos motores, representado na área de software como Controlador de Máquina. A comunicação entre os dois é realizada por meio da modelagem de dados PUDM, a ser especificada na Seção 2.5.2.1. A figura 40 mostra que qualquer outra espécie de controlador de máquina pode ser, em teoria, compatível com o Servidor de Decisão desenvolvido desde que a modelagem de dados seja obedecida.

Um ponto a se destacar é que embora a construção do protótipo físico no projeto e a implementação do Controlador de Máquina sejam referenciadas na solução de software eles são melhor definidos em outras partes da solução, esses itens serão discutidos nesta solução a partir de um ponto de vista de software. Isso acontece já que para podermos validar o Servidor de Decisão e a modelagem de dados desenvolvidos, houve a necessidade de criar a simulação da mesa de pebolim na forma de um jogo eletrônico para poder agir de forma a simular a mesa de pebolim física.

2.5.2 Software para Mesa de Pebolim

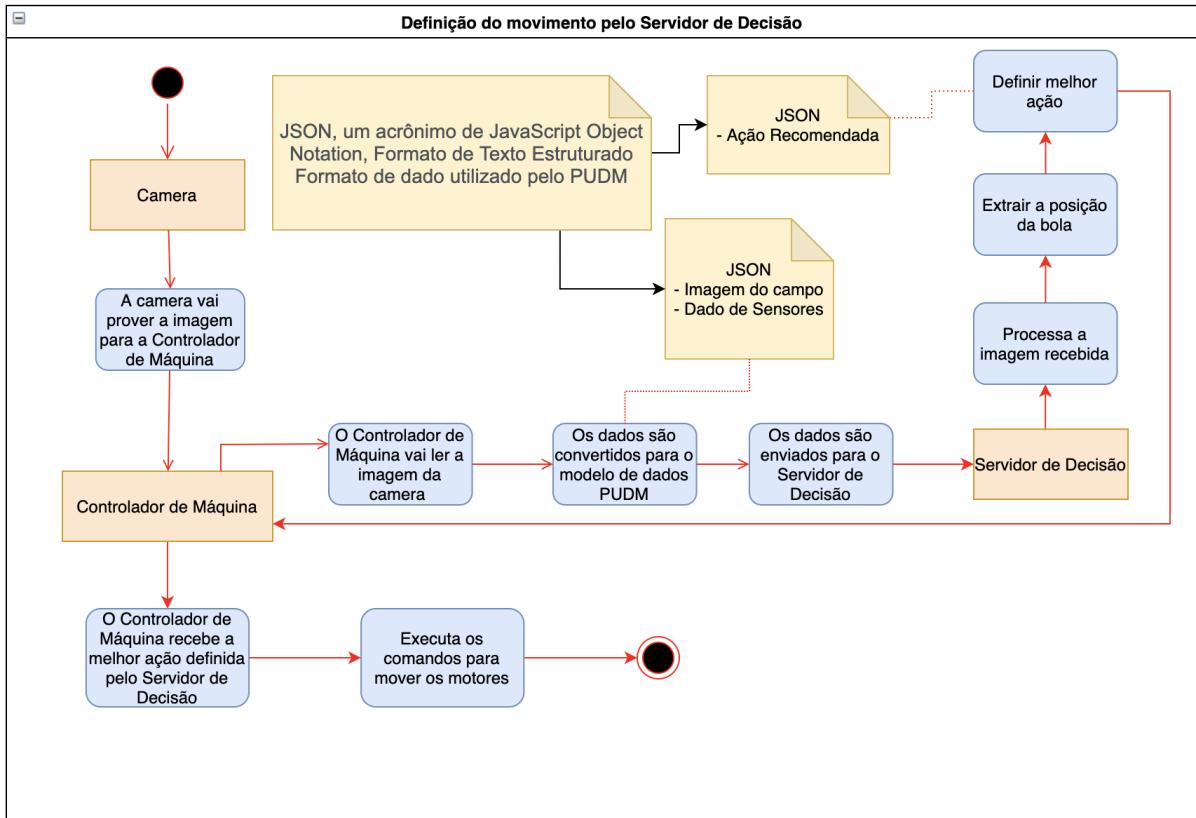


Figura 38 – Diagrama de Atividades da Solução de Software

A figura 38 mostra o comportamento geral da solução de software, incluindo o que se espera receber do Controlador de Máquina da mesa de pebolim e mostrando as etapas gerais que devem ocorrer.

O software que operacionaliza a mesa de pebolim recebe as imagens capturadas do campo pela câmera através do Controlador de Máquina, que realiza a leitura e converte seus dados para a modelagem PUDM, unificando os dados e as imagens. Após isso, os envia para o Servidor de Decisão. Este, por sua vez, processará as imagens para extrair a posição da bola e definir a melhor decisão a ser tomada na jogada, que então é passada para o Controlador de Máquina que tem o papel de comunicar os comandos necessários para que os motores realizem a ação definida pelo Servidor de Decisão.

A figura 39 exibe como a solução de software pode ser vista a partir do caso de uso do usuário que deseja treinar jogando contra a inteligência artificial da mesa de pebolim.

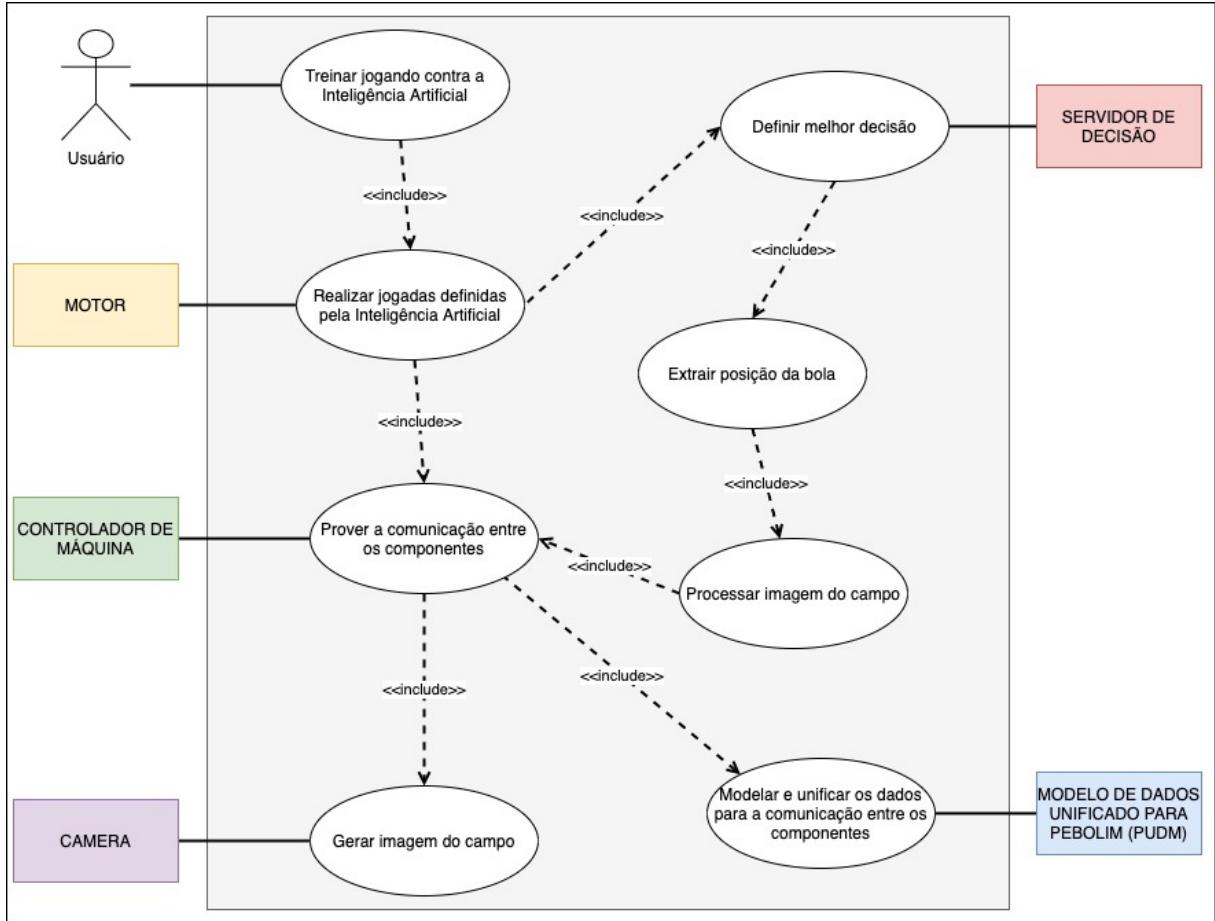


Figura 39 – Caso de Uso da Solução de Software

2.5.2.1 PUDM

O Pebolim Unified Data Model (PUDM) é uma modelo de dados criado para permitir a utilização de diversos controladores de máquina na arquitetura da mesa de pebolim.

A ideia deste modelo de dados surge a partir do conceito de *Interface* nas linguagem de programação orientadas a objetos, em que esta representa uma estrutura que funciona como um contrato, garantindo que as classes que o implementem sejam obrigadas a ter determinados métodos. O PUDM faz este papel de contrato, porém, para os componentes envolvidos na mesa de pebolim automatizada, ele orienta como deve ser feita a comunicação entre o controlador da mesa de pebolim e o Servidor de Decisão e, assim, garantir a compatibilidade entre os sistemas.

O modelo PUDM é feito para que seus eventos sejam criados em formato de *JavaScript Object Notation* (JSON). O JSON é um formato leve de troca de dados entre sistemas que pode ser manipulado como um texto comum, trazendo simplicidade ao manipular esses dados, aumentando a sua velocidade de transporte e diminuindo o tempo de

execução do software de um sistema que o utiliza.

O modelo PUDM é definido com base em eventos onde cada um deles possui atributos próprios. Abaixo encontram-se listados os eventos definidos no PUDM, acompanhados de uma breve explicação. A descrição detalhada do modelo de dados pode ser encontrada no Apêndice D.2.

- RegisterEvent: é enviado do cliente para o servidor por meio de uma requisição *HTTP POST* no início de sua execução. Serve para identificar a mesa e suas características no servidor;
- StatusUpdateEvent: é enviado do cliente para o servidor para informar do estado atual da mesa, junto da imagem mais recente da câmera;
- ActionEvent: é enviado a partir do servidor para o cliente com o resultado do processamento da inteligência artificial.

Os eventos *RegisterEvent* e *StatusUpdateEvent* são gerados no lado cliente e enviados por meio de requisições *HTTP POST*. Já o evento *ActionEvent* é gerado no lado servidor e é enviado por meio de uma conexão *websocket*, a qual o cliente deve solicitar a criação.

O evento *StatusUpdateEvent* faz referência a codificar a imagem em formato *Base64*. Isso é feito para possibilitar a transmissão dos *bytes* da imagem dentro do texto no formato JSON usado pelo PUDM. Este processo de codificação em *Base64* consiste em converter uma quantidade de *bytes* em um texto escrito apenas com caracteres de um conjunto de 64 caracteres que representem estes *bytes*. Esta codificação obedece um padrão pré-estabelecido, permitindo que outros componentes façam a decodificação do texto de volta para os *bytes* originais.

É importante ressaltar que todas as tecnologias relacionadas ao PUDM (JSON, codificação *Base64*, HTTP POST, Websockets) são comuns na indústria de software e possuem diversas aplicações. Além disso, eles possuem compatibilidade com diversos sistemas operacionais, arquiteturas de computador e linguagens de programação, tais como C, C++, C#, Java, Python, entre outras.

A figura 40 exibe visualmente como o PUDM é utilizado dentro da solução de software.

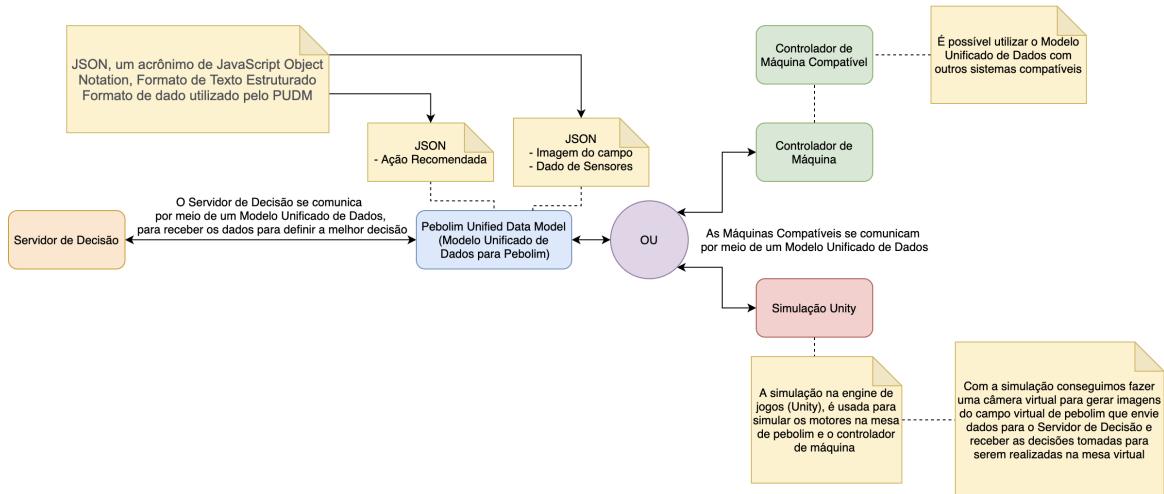


Figura 40 – Fluxograma do PUDM

2.5.2.2 Servidor de Decisão

O Servidor de Decisão do Pebolim é a camada responsável por receber os dados encaminhados pelo Controlador da Máquina e realizar inferências de controle a partir dos estados disponibilizados. A ideia de criar um serviço independente para fazer decisão e usar uma interface para interagir diretamente com comandos de controle é tirar a responsabilidade de formatação e adaptação de dados da regra de negócios e garantir uma responsabilidade única por camada dentro do projeto. Neste caso, o Servidor de Decisão seguiria as etapas descritas na figura 41.

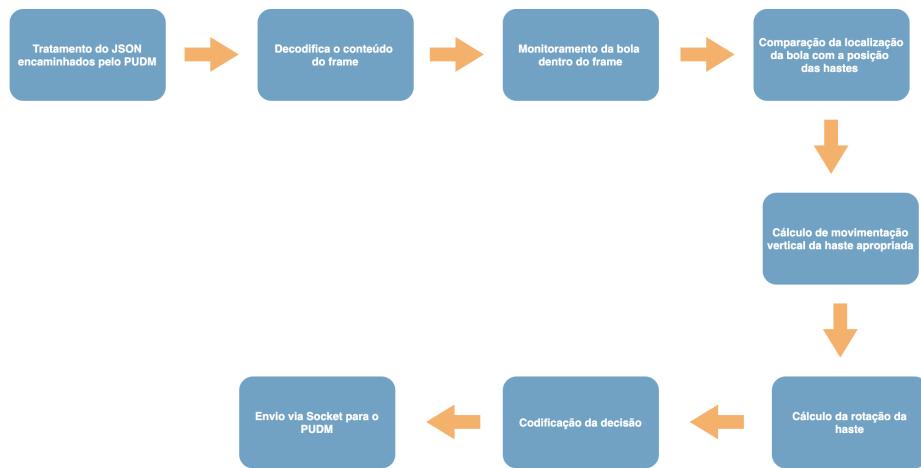


Figura 41 – Fluxo do Servidor de Decisão

Como mostrado na figura 41, o servidor começa tratando os dados encaminhados pela interface para o padrão adequado ao processamento. Como estamos lidando com dados gerados por imagens, os dados são encaminhados através de uma mensagem que

contém a posição das hastes e a imagem do estado atual da mesa. Quando é feita a decodificação da imagem, é utilizado um algoritmo de rastreamento usando técnicas de visão computacional. Primeiramente é detectado a posição da bola em cada frame e a partir de um *Deque*, estrutura de dados responsável por salvar o histórico com as imagens encaminhadas durante o jogo de pebolim, é feito o rastreamento do objeto, coletando métricas de direção e velocidade da bola no campo.

Com a posição da bola e os dados das posições das hastes, a distância entre eles é analisada e feita uma estimativa de trajetória da bola. Com a trajetória estimada, é possível designar uma movimentação para cada haste e é averiguada a possibilidade de rotação da haste caso um jogador daquela haste possa interceptar a bola e realizar um contra-ataque. Após a execução de todo o fluxo, é feito o agrupamento de todos os dados recolhidos nas etapas anteriormente descritas e um novo evento é encaminhada de volta para a interface de controle com os novos comandos a serem designados aos motores. O diagrama de sequência da figura 42 mostra o fluxo descrito anteriormente em um formato baixo nível, onde é possível visualizar a comunicação entre as camadas e os principais eventos recebidos e enviados pelo Servidor de Decisão.

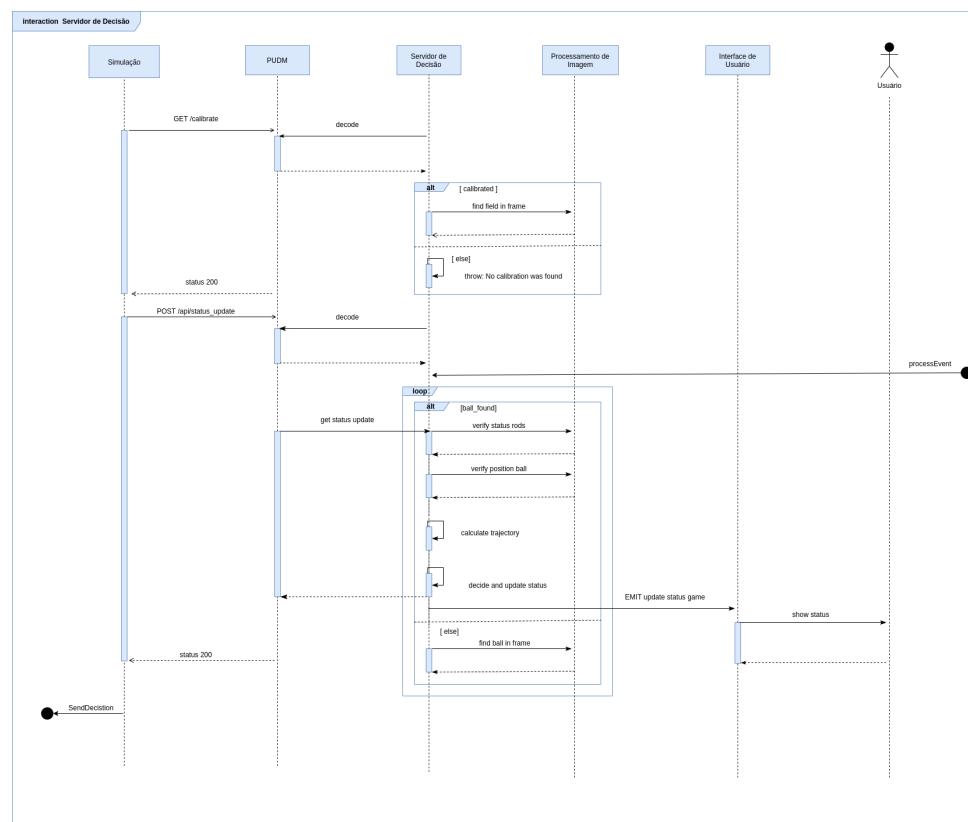


Figura 42 – Diagrama de Sequência do Servidor de Decisão

2.5.2.2.1 Organização

Para desenvolver o Servidor de Decisão foi usado o framework Flask, que é baseado na linguagem Python. O Flask tem a funcionalidade de criar um servidor Rest, pelo qual a simulação irá se conectar com o servidor para saber qual ação deve tomar, comunicando-se via requisições HTTP. Além disso o Flask possui a funcionalidade de agir como servidor Websocket, o que também será usado para a comunicação com o cliente. Como o projeto desenvolvido usa em sua grande parte algoritmos de visão computacional para fazer detecção e rastreamento, a decisão de usar ferramentas baseadas em Python foi de grande ajuda, pois esta linguagem tem um universo consolidado de bibliotecas para lidar com algoritmos de processamento de imagem e vídeo. A escolha da utilização do Flask também foi influenciada por uma preocupação relacionada ao desempenho do software, o Flask é um framework construído de forma a causar o menor impacto, pois é modelado de forma a ser uma *micro-framework*, ou seja, fazer somente o trabalho necessário.

A biblioteca OpenCV do Python foi usada para realizar as detecções e o rastreamento da bola dentro das imagens recebidas em formato PUDM no projeto e assim realizar as decisões posteriormente. Os principais algoritmos já estão implementados dentro da biblioteca e ela conta com uma comunidade movimentada, além de documentação madura dentro do ecossistema de visão computacional.

Para manter uma estrutura de código manutenível e legível, foram usados os princípios do SOLID, que é um acrônimo para um conjunto de princípios usado para tornar o design de software do projeto mais entendíveis. O significado de cada uma das letras do princípios está listado abaixo.

- O "S" remete a responsabilidade única para uma classe, em que cada uma delas devem ter uma única responsabilidade na qual remete ao nome designado a ela.
- O "O" vem do Princípio Aberto/Fechado, em que as entidades de software devem ser abertas para extensões, mas fechadas para modificações.
- O "L" vem do Princípio da Substituição de Liskov, em que as classes base devem ser substituíveis por suas classes derivadas.
- A letra "I" vem do Princípio da Segregação da Interface, em que uma classe não deve ser forçada a implementar interfaces e métodos que não irão utilizar.
- O "D" remete ao Princípio de Inversão de Dependências, em que as instâncias deverão depender das abstrações e não das implementações.

Com o uso da ferramenta Flask e o mínimo acoplamento proposto pelo framework, a adaptação para o padrão SOLID foi bem mais fácil, comparado aos outros frameworks da mesma linguagem, como é mostrado no diagrama de classe presente na figura 43.

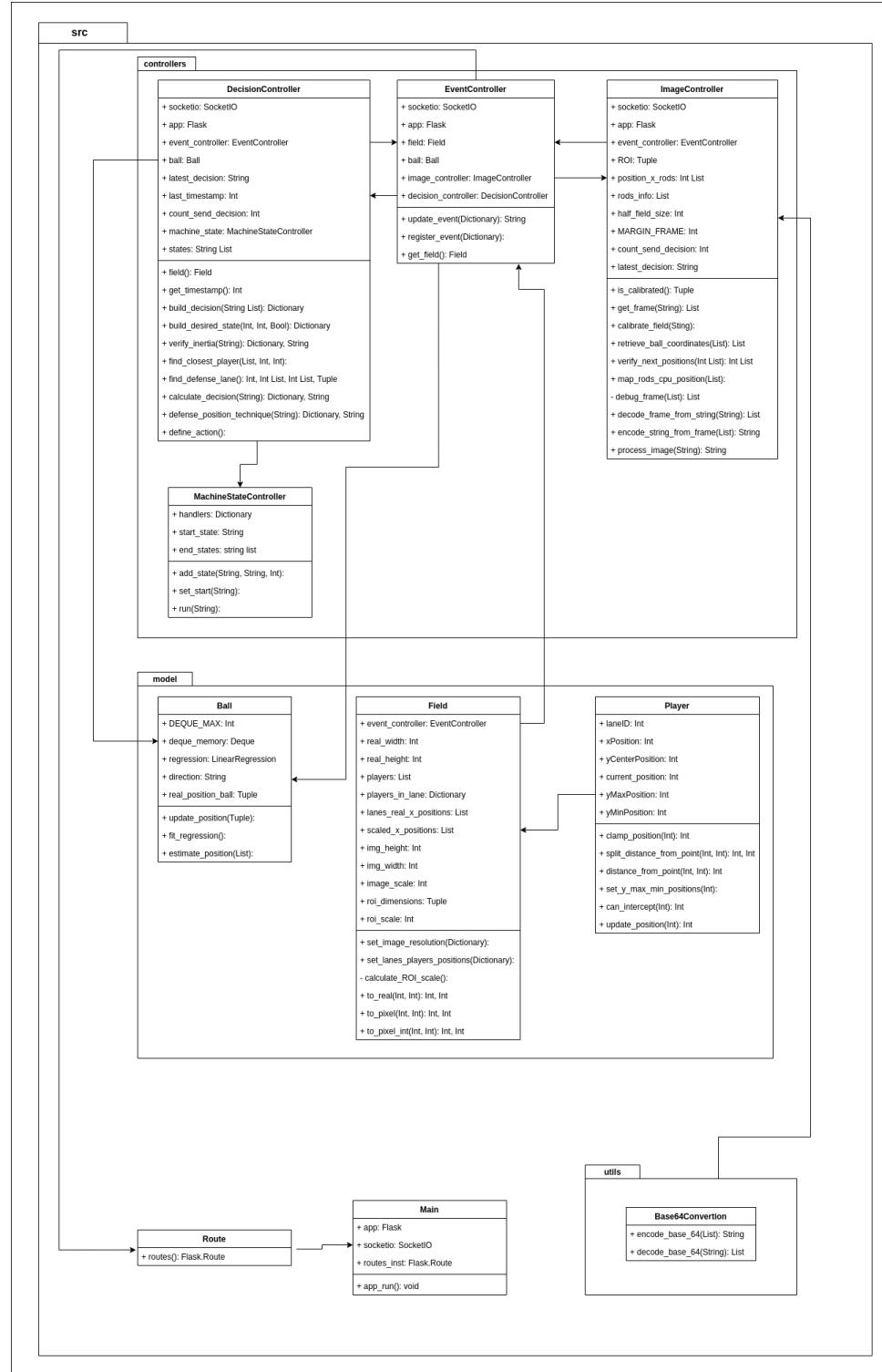


Figura 43 – Diagrama de Classe do Servidor de Decisão

No servidor de decisão há duas classes principais, onde é realizado toda a regra de negócio em relação a decisão e atualização das ações para o controlador de máquina, são elas a classe Calibration, responsável por verificar o primeiro frame vindo da simulação e analisar qual é a parte importante dentro do frame (campo), de modo que não haja nenhum ruído no momento de decisão das ações de jogo. A segunda classe importante é

a Decision, no qual usa os parâmetros de calibração para localizar a bola dentro do frame e em seguida analisar quais ações devem ser tomadas para que a bola seja bloqueada.

Além do mais há classes intermediarias que servem para ajudar as classes principais na ação de decisão. As classes de Location e State são classes abstratas, onde seu objetivo é apenas formatar os dados que implementem elas e evitar a duplicação de código como o paradigma de Orientação a Objetos e o SOLID propõem. Há também a classe Utils que contém métodos para lidar com a codificação Base64 usada nos dados da imagem evitadas via PUDM. A classe Main é responsável por executar o servidor de Decisão e a de Routes tem o objetivo de organizar os *endpoints* do Servidor de Decisão e disponibilizar estes para o Controlador de Maquina/Simulação.

2.5.2.2.2 Interface de Usuário

Um dos requisitos citados na Seção ?? é a presença de uma tela sensível ao toque para a interação do usuário com a mesa de Pebolim de uma forma mais amigável. Para atender os requisitos mencionados, foram desenvolvidas interfaces para a tela especificada, onde englobasse a customização das principais funcionalidades da mesa automatizada, como: a calibração da câmera da mesa que disponibiliza as imagens para o servidor de decisão, a escolha da dificuldade de jogo contra a máquina e o acompanhamento do placar da partida em tempo real do jogo.

Para que a interação do usuário ocorresse de forma fluída, as telas foram desenvolvidas para ter uma interface amigável e intuitiva, com botões grandes devido a natureza *touch screen* da tela, de forma que o usuário que está interessado em usar a mesa encontre a funcionalidade requisitada facilmente e em poucos toques de tela.

A apresentação é a primeira tela do fluxo de interface, tem o intuito de apenas apresentar o nome do produto e ser uma tela intermediária enquanto o usuário não começa o jogo.

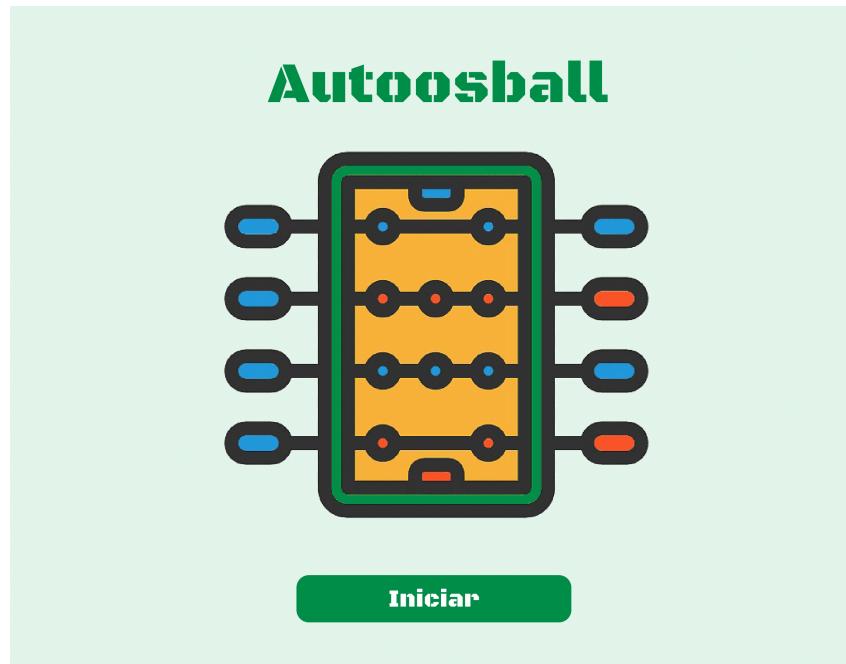


Figura 44 – Tela de Apresentação

Na tela abaixo o usuário consegue selecionar o nível de dificuldade que o jogadores controlados pelo servidor de decisão jogará na partida.



Figura 45 – Tela para selecionar a dificuldade de jogo

O usuário poderá também calibrar a câmera da mesa de Pebolim usando a tela mostrada abaixo. Com a interação da calibração por meio do usuário, o servidor de decisão será mais preciso durante o reconhecimento dos objetos, pois o campo de visão calibrado

incluirá apenas os objetos do campo, excluindo ruídos externos que podem atrapalhar no reconhecimento da bola ou das hastas.



Figura 46 – Tela responsável por calibrar a câmera da mesa

Após os passos de escolha nível de dificuldade e calibração da mesa, o painel do placar é mostrado ao usuário para que este configure a duração de jogo e acompanhe os pontos marcados durante a partida.



Figura 47 – Tela do placar de jogo



Figura 48 – Tela do placar de jogo sendo executada

²

2.5.2.2.3 Restrições

Uma das fases de desenvolvimento do Servidor de Decisão é a detecção dos objetos na mesa de pebolim. Como o Servidor lida com recursos visuais, é necessário usar algoritmos de visão computacional para fazer as predições de objetos nos quadros dos vídeos. Dentro da seção de algoritmos de rastreamento, existem um conjunto vasto de opções com pequenas particularidades para predição. A quantidade de quadros processados por segundo e a acurácia em rastrear os objetos são as principais variáveis para a escolha de um algorítimo desse porte (LIU et al., 1998).

Como citado na Seção 2.5.2.2.1, a biblioteca OpenCV foi utilizada para realizar o rastreamento do campo de jogo e da bola de pebolim em cada um dos quadros enviados para o Servidor de Decisão. Durante o processo de calibragem, o algoritmo primeiramente detecta o campo de jogo, conforme mostrado na figura 49 abaixo.

² <https://www.figma.com/file/0WLka9l7QeDjOwg4p35LFT/Pi2-Final>

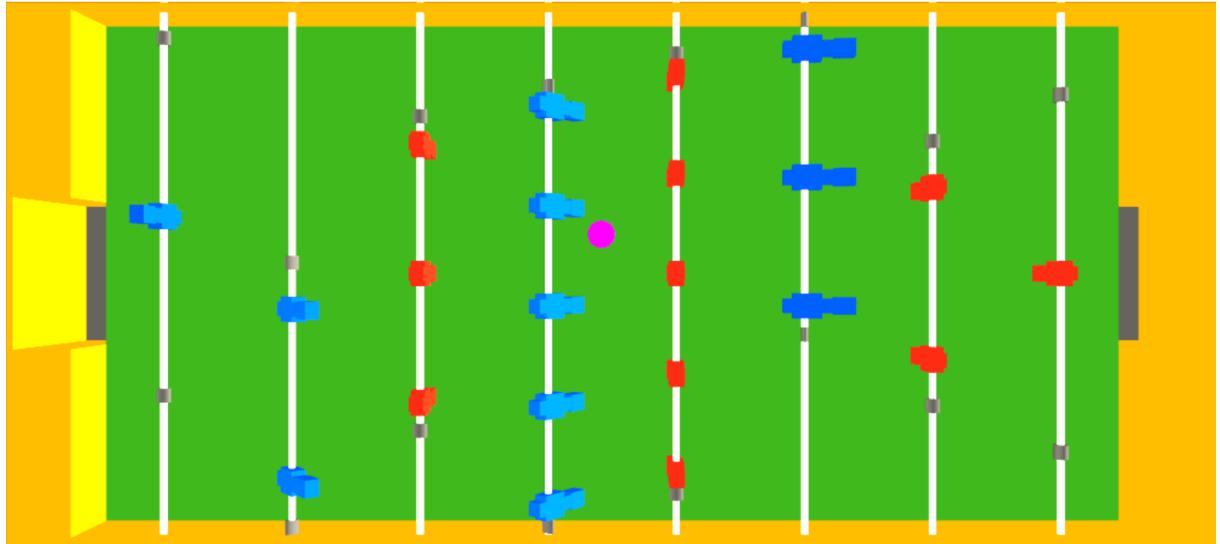


Figura 49 – Imagem do campo de jogo gerada pela simulação

Depois de ter detectado o campo de jogo, a imagem passa pela mudança do sistema de cor RGB (*red*, *green*, e *blue*) para o sistema de cor HSV (*hue*, *saturation* e *value*) para facilitar a criação de uma máscara que vai permitir extrair os contornos dos objetos dessa imagem. A figura 50 mostra a imagem depois de ter trocado de sistema de cor.

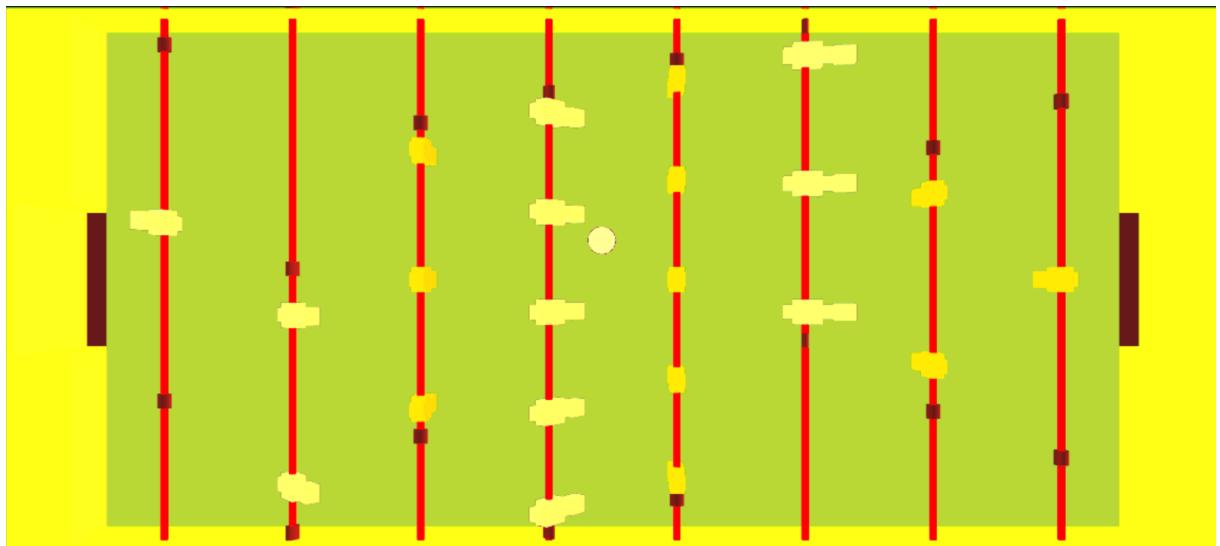


Figura 50 – Imagem do campo de jogo após a troca do sistema de cor

Feita a troca do sistema de cor, a imagem passa por um processo de mascaramento para extrair os contornos evidenciados no frame e assim procurar por um contorno com formato redondo. A figura 51 mostra a máscara criada a partir da imagem com a posição da bola de pebolim em destaque.

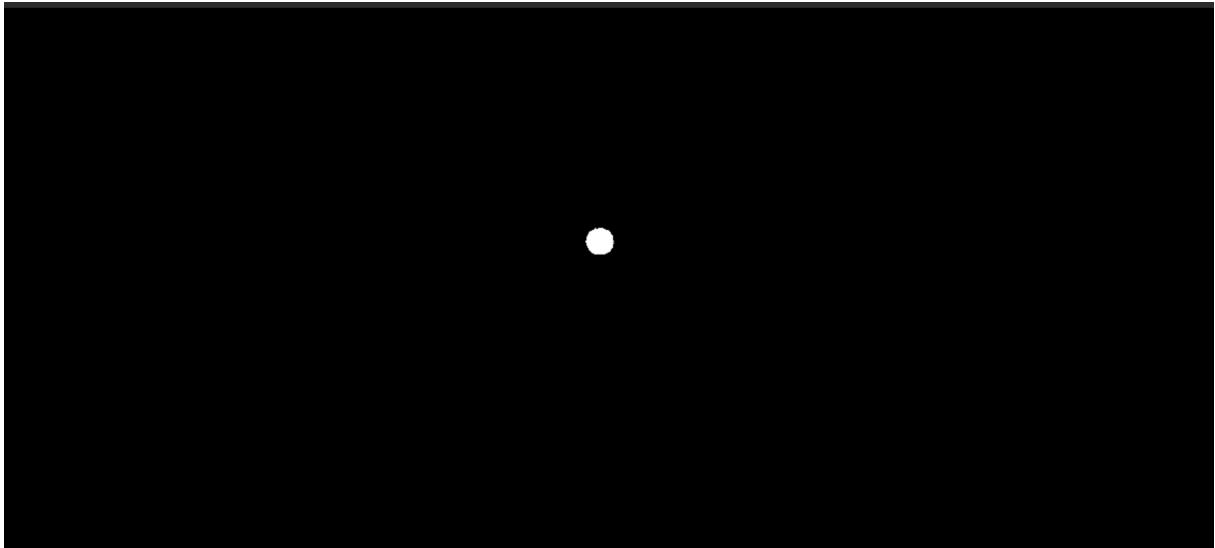


Figura 51 – Máscara da imagem do campo de jogo para marcar a posição da bola

Por fim, com a posição do centro e o tamanho do raio da bola de pebolim, o algoritmo marca o seu centro e cria um círculo circunscrito na bola da imagem original, conforme mostrado na figura 52, e encaminha essas informações, tanto a posição quanto a nova imagem com a bola destacada, para o Servidor de Decisão realizar os cálculos com base na posição atual e nas posições já armazenadas no **Deque** (histórico com as posições anteriores da bola) e assim tomar uma decisão sobre o evento analisado a enviar para o Controlador da Máquina.

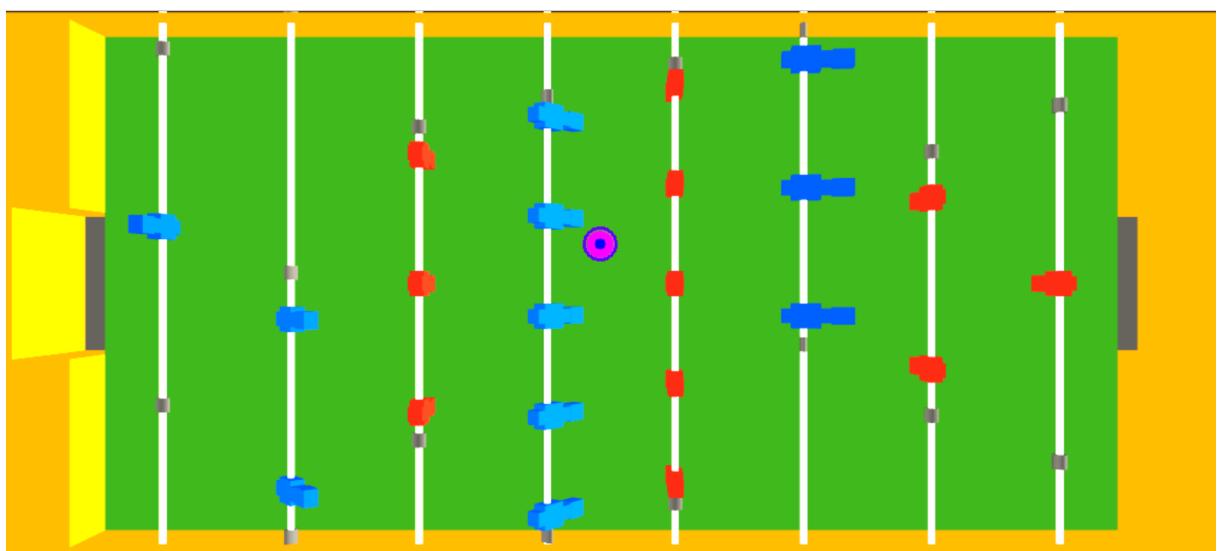


Figura 52 – Imagem do campo de jogo com a bola marcada

Por conta do processamento necessário para coletar essas informações das imagens recebidas pelo Servidor de Decisão, as duas restrições mais importantes do projeto são

o tempo de resposta do algoritmo durante o processo de detecção da posição da bola de pebolim e o tempo de resposta do algoritmo para tomada de decisão. Por conta disso, foi necessário realizar medições do tempo de execução do algoritmo de detecção e de tomada de decisão e seus respectivos cálculos de tempo médio de execução para estimar esse tempo de resposta.

Para realizar a estimativa do tempo médio de execução do algoritmo de rastreamento da bola no campo de pebolim, realizamos a medição desse tempo em dez oportunidades, medindo o tempo de execução cem vezes em cada uma delas. A figura 53 mostra a distribuição dos tempos medidos em cada uma das dez medições realizadas e o comportamento da média deles ao longo delas.

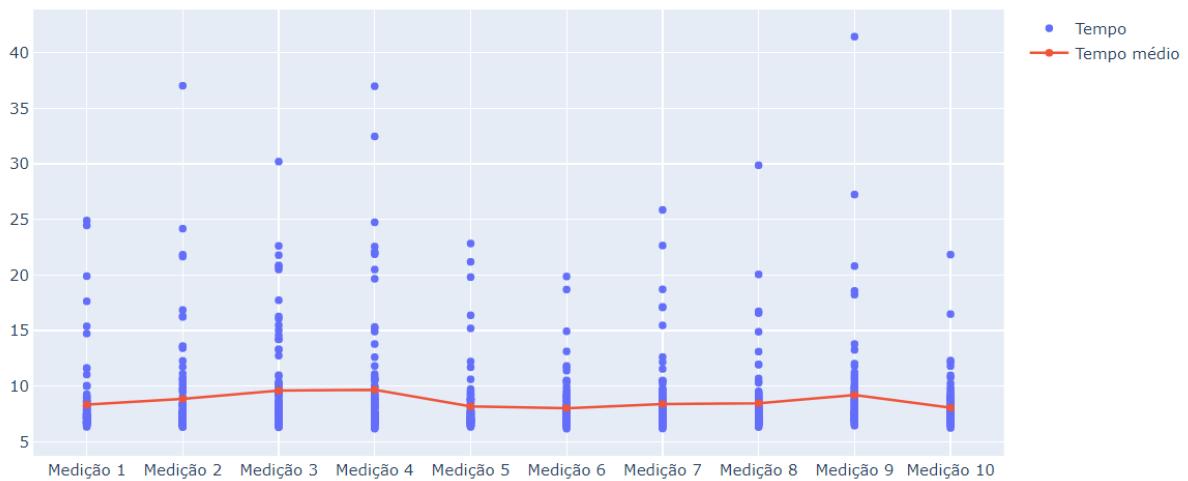


Figura 53 – Tempo médio (em ms) de execução do algoritmo de detecção da bola de pebolim

Na Tabela 6 encontram-se os tempos médios de execução do algoritmo de detecção da bola que foram calculados a partir das cem execuções do algoritmo em cada uma das dez medições realizadas.

Tabela 6 – Tempo médio de execução do algoritmo de detecção da bola de pebolim

Medição	Tempo Médio (ms)
1	8,324828147888184
2	8,85556697845459
3	9,585514068603516
4	9,672987461090088
5	8,166718482971191
6	8,01135778427124
7	8,385565280914307
8	8,450908660888672
9	9,191820621490479
10	8,053457736968994

Calculando a média dos tempos médios da Tabela 6 obtemos o valor aproximado de 8,669872522 ms.

Assim como realizado para a detecção da bola, para realizar a estimativa do tempo médio de execução do algoritmo de tomada de decisão realizamos a medição desse tempo em dez oportunidades, medindo o tempo de execução cem vezes em cada uma delas. A figura 54 mostra a distribuição dos tempos medidos em cada uma das dez medições realizadas e o comportamento da média deles ao longo delas.

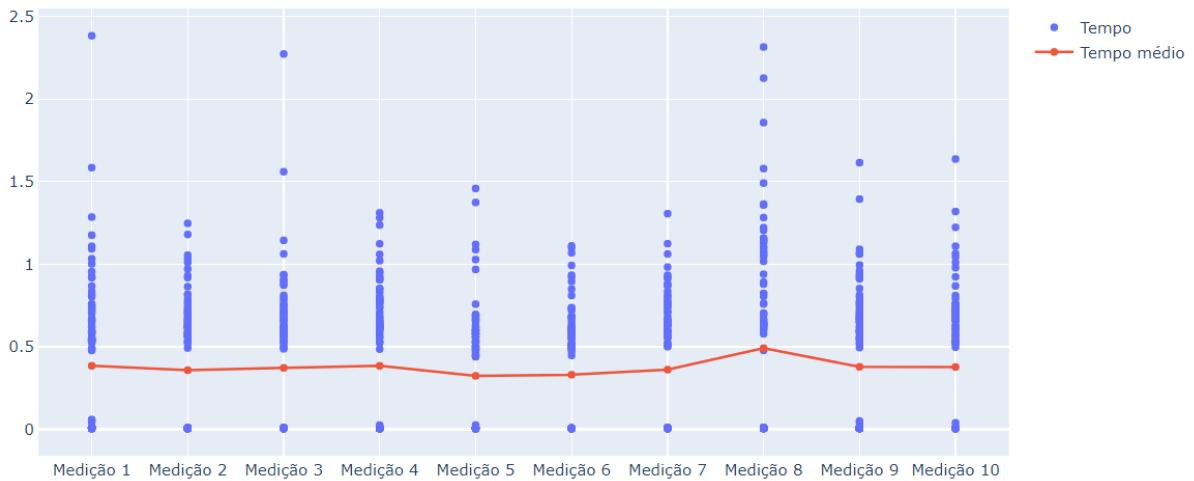


Figura 54 – Tempo médio (em ms) de execução do algoritmo de tomada de decisão

Na Tabela 7 encontram-se os tempos médios de execução do algoritmo que foram calculados a partir das cem execuções do algoritmo em cada uma das dez medições realizadas.

Tabela 7 – Tempo médio de execução do algoritmo de tomada de decisão

Medição	Tempo Médio (ms)
1	0,38437604904174805
2	0,35786867141723633
3	0,3715825080871582
4	0,38407087326049805
5	0,3234219551086426
6	0,33017873764038086
7	0,3609490394592285
8	0,49103498458862305
9	0,37793874740600586
10	0,3771066665649414

Calculando a média dos tempos médios da Tabela 7 obtemos o valor aproximado de 0,3758528232574463 ms.

2.5.2.2.4 Inteligência Artificial

Ao realizar a sua inicialização, a mesa de pebolim envia o *RegisterEvent* para o Servidor de Decisão com as informações sobre a mesa onde a solução foi instalada. Entre tais informações encontram-se as dimensões reais da mesa em centímetros, as dimensões, em pixels, das imagens capturadas pela câmera e a quantidade de imagens por segundo a serem enviadas pela câmera, a identificação e a posição real de cada uma das hastas, a quantidade de jogadores em cada uma delas, a distância entre eles e o limite de movimentação que estes jogadores obedecerão. Tais informações são fundamentais para realizar os ajustes da inteligência artificial do Servidor de Decisão.

Com essas informações iniciais registradas calcula-se a proporção entre as dimensões reais da mesa de pebolim e a imagem retirada pela câmera para que a posição da bola extraída das imagens possa ser convertida para a posição real dela na mesa tendo em vista que o rastreamento da bola está sendo feito a partir da imagem recebida da câmera, conforme descrito na Seção 2.5.2.2.3.

Feito isso, calcula-se a posição inicial de cada um dos jogadores em cada uma das hastas a partir da posição central de cada uma delas, bem como o intervalo de posições possíveis para as quais cada um desses jogadores pode se mover e assumir uma posição defensiva durante a partida ou, até mesmo, realizar a ação de chutar a bola para contra-atacar.

Utilizando as posições da bola que foram armazenadas em um *buffer* identifica-se se a bola está parada ou em movimento, e, se estiver em movimento, qual é a direção que a bola está tomando para saber se os jogadores devem retornar à posição inicial ou assumir posições visando a defesa do gol.

Para verificar se os jogadores devem assumir posições de defesa do gol, calcula-se qual dos jogadores de cada uma das hastas está melhor colocado e qual a posição que ele deve assumir para defender o gol. Com essa informação, calcula-se qual o movimento cada haste deve realizar com relação à sua posição inicial e, se possível, realizar o movimento de chute para contra-atacar.

A figura 55 identifica os passos realizados pelo Servidor de Decisão da seguinte maneira: acima encontram-se as coordenadas da posição da bola no tamanho real (em centímetros) e na imagem (em pixels) e a direção que a bola está tomando ("left"); os círculos na cor azul claro indicam a posição inicial de cada um dos jogadores, os círculos menores indicam a posição atual do jogador e os círculos maiores na cor preta indicam as posições que eles devem assumir para defender o gol, caso um circulo maior esteja colorido em azul, significa que há o comando de chute; e as linhas em azul indicam qual jogador de cada haste está mais próximo da bola, já a linha vermelha um histórico das posições da bola. Nem sempre o desenho destas informações se posiciona exatamente no ponto do objeto físico na imagem, pois há uma distorção gerada pela lente da câmera.



Figura 55 – Verificação das funcionalidades do Servidor de Decisão

Com o intuito de melhorar a experiência de decisão do servidor, foi usado o conceito de *Máquina de Estado Finita*, definida como um modelo matemático computacional ou representação de um fluxo bem definido, onde há um número finito de estados selecionados que a aplicação pode se encontrar em um determinado espaço de tempo.

Há diversas formas de se representar uma máquina de estado, a mais comum se assemelha a estrutura de dados Grafo, onde para cada Nô dentro da estrutura é atribuído um estado selecionado dentro do contexto desenvolvido. Cada Nô pode se conectar com vários ou nenhum Nô de outro estado dependendo do fluxo definido pelo domínio execu-

tado. A ideia geral desse tipo de estrutura é que para cada estado listado, há um tipo de ação bem definida para o estado selecionado em um determinado momento, de forma que os fluxos padrões dentro do domínio se comportem da forma esperada e sejam alto explicativos.

A *Máquina de Estado* desenvolvida para o Servidor de Decisão encontra-se na figura 56.

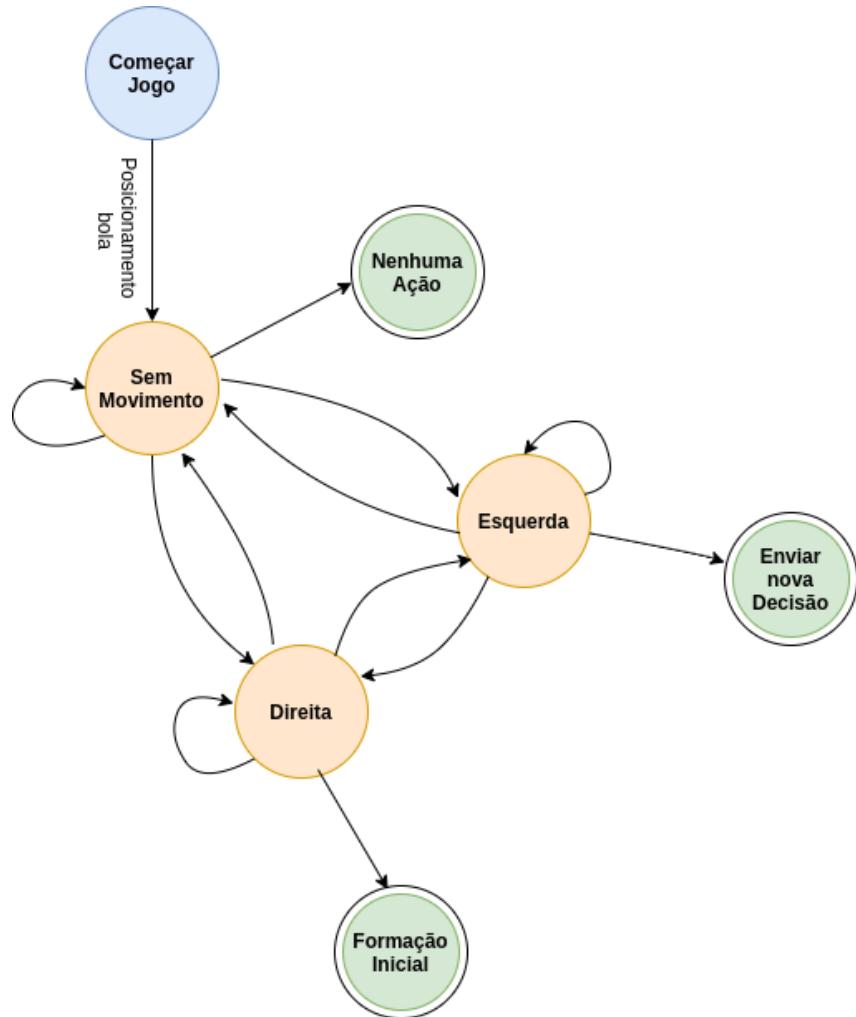


Figura 56 – Diagrama de estados do servidor de decisão

Ao determinar a decisão final, a bola pode estar em um dos três estados diferentes dependendo de qual direção está sendo aplicada. São elas: sem movimento, quando a bola permanece na mesma posição desde a ultima decisão feita; Esquerda, quando a bola está indo em direção ao campo do time controlado pelo servidor de decisão; Direita, quando a bola está indo na direção do gol do campo inimigo. Dependendo da direção que a bola se encontra o estado é mudado, e uma ação diferente é aplicada.

Na figura 56, é possível visualizar três etapas durante a execução da máquina. O nó em azul é o estado inicial, no caso o começo de jogo. Os nós em amarelo são estados

intermediários onde a bola do jogo pode se encontrar, ou seja, sem movimento, esquerda ou direita. Os Nós em verde são as ações acarretadas por cada estado intermediário.

Quando não há nenhuma movimentação da bola, é comandado que a linha imediatamente atrás da bola se alinhe a bola e tente executar um chute, já quando a bola está indo em direção ao campo da máquina controlada pelo servidor, o servidor calcula a trajetória da bola e verifica se há a possibilidade de interceptação e chute para a situação de jogo. Caso a bola vá em direção ao campo inimigo, os jogadores à frente da bola são movidos para fora do caminho ou postos de forma a alterar lateralmente a trajetória da bola enquanto isso os jogadores atrás da bola se posicionam de forma a se defender de um possível contra-ataque dos jogadores inimigos, ficando na posição inversa da trajetória.

2.5.2.3 Controlador de Máquina

O Controlador de Maquina representa o componente que interage diretamente com os motores na mesa de pebolim. É responsável por converter os comandos recebidos do Servidor de Decisão em movimentos na mesa. Além disso, é responsável por coletar os dados físicos e enviar para o Servidor de Decisão. Implementa o lado cliente do PUDM.

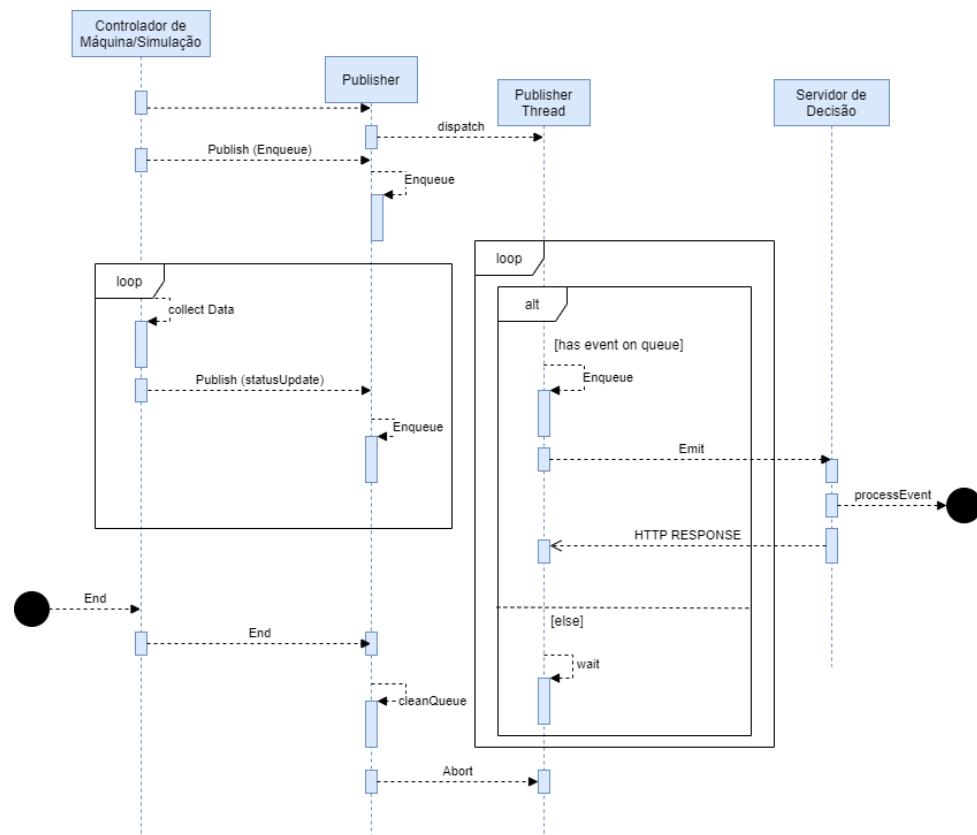


Figura 57 – Diagrama de sequencia da parte de envio de cliente PUDM

É importante observar que a forma como o cliente PUDM é implementado fisicamente não é relevante, desde que os seus padrões de comunicação para entrada e saída de

dados adotem o PUDM. E que sua implementação física não será abordada na área de software, já que sua importância nesta área é apenas relacionada à simulação.

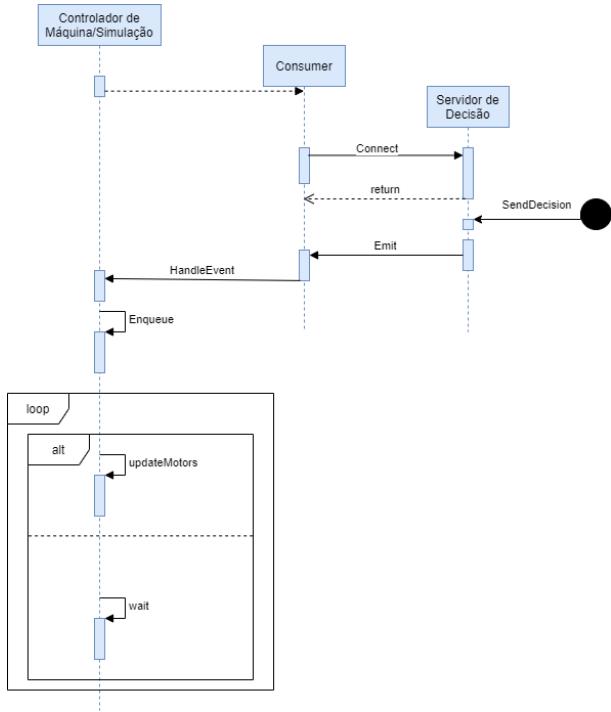


Figura 58 – Diagrama de sequencia da parte de recebimento de uma decisão gerada pelo Servidor de Decisão

Na figura 57 é exibida, no formato de um Diagrama de Sequencia, uma representação generalizada dos eventos para enviar um dado, em formato PUDM, a partir do Controlador de Máquina ou Simulação, para o Servidor de Decisão. Já a figura 58, no mesmo formato, representa uma generalização do processo de recebimento de um evento gerado pelo Servidor de Decisão.

2.5.2.3.1 Simulação

Como explicado na Seção 2.5, uma simulação em formato de jogo eletrônico da mesa de pebolim será usada para auxiliar o desenvolvimento da Inteligência Artificial do Servidor de Decisão. Na figura 59 é demonstrado o caso de uso a partir do ponto de vista do desenvolvedor.

A simulação contém uma mesa de pebolim desenhada em um modelo tridimensional com dimensões reais.

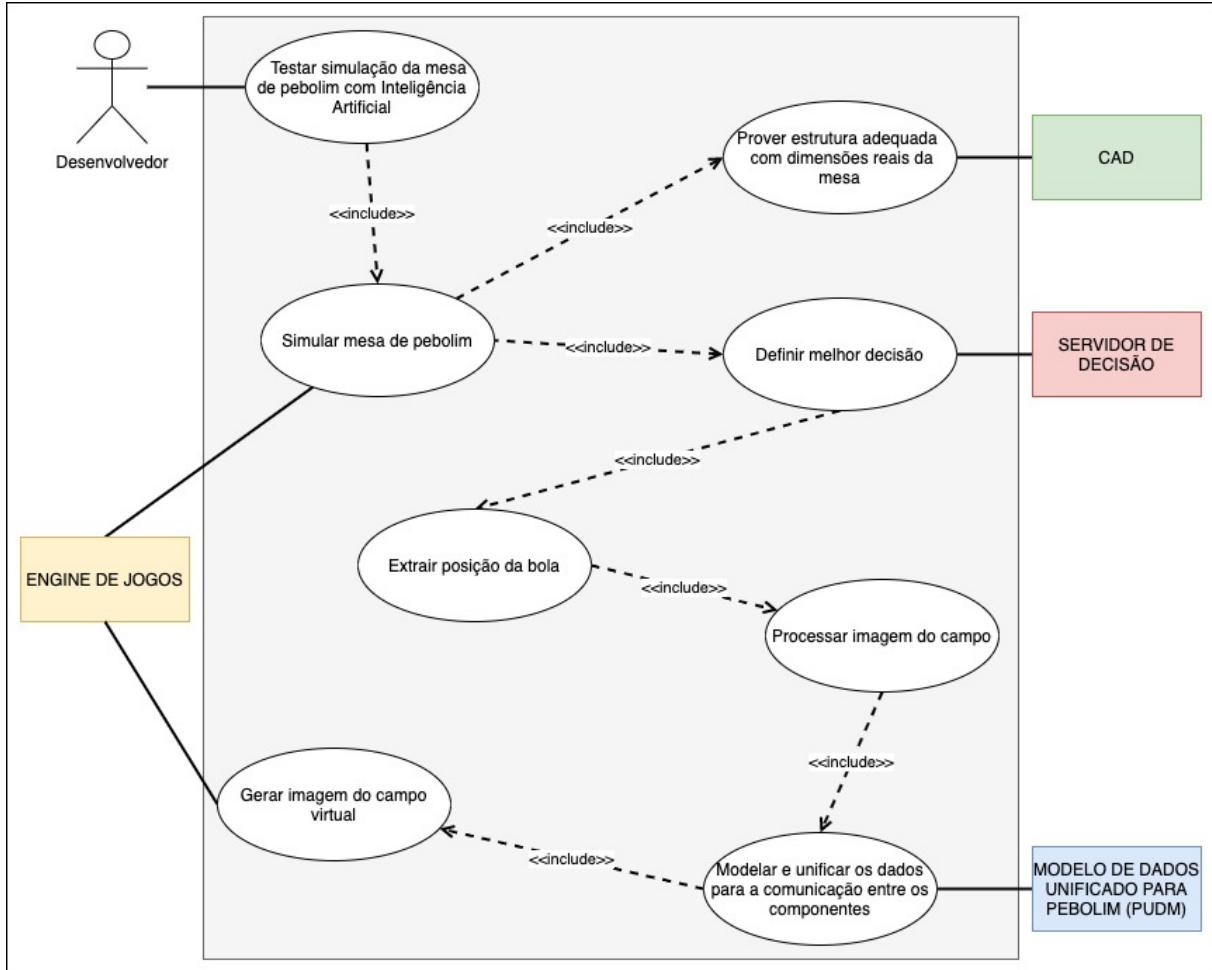


Figura 59 – Caso de Uso da Simulação na Engine de Jogos

A simulação da mesa de pebolim virtual foi desenhada na *engine* de jogos seguindo as medidas de uma mesa profissional. Por meio de *scripts* criados para controlar os objetos da mesa virtual, foram configurados os movimentos das hastes e jogadores, e também adequação da física dos objetos para o que se espera, subjetivamente, de uma mesa de pebolim.

A *engine* de jogos escolhida é a Unity. Esta é uma *engine* comercial de jogos, com uma versão gratuita que atende as necessidades do projeto. A Unity permite a execução de *scripts* em linguagem de programação C# para o controle dos objetos virtuais que estão dentro do jogo.

A integração da simulação da mesa de pebolim na *engine* de jogos com o Servidor de Decisão, se dá por da implementação do modelo de dados PUDM, em que os dados são enviado a partir de um método de requisição POST do protocolo HTTP, e os dados são recebidos por meio de WebSockets que permite uma comunicação interativa com o modelo PUDM.

As imagens são geradas por meio de uma câmera virtual, que captura uma imagem do jogo de acordo com as especificações da câmera real do projeto descrita na Seção ???. A câmera virtual implementa um *timer* para se adequar a taxa de quadros e converte a imagem para o tamanho correto. Um exemplo de imagem gerada pode ser visto na figura 49.

A simulação de física é uma das limitações enfrentadas durante o desenvolvimento. Não foi tomado como objetivo replicar a física do mundo real de forma perfeita, apenas simular o suficiente para permitir o desenvolvimento do servidor de decisão gerando suas imagens e simulando comportamentos de forma aproximada.

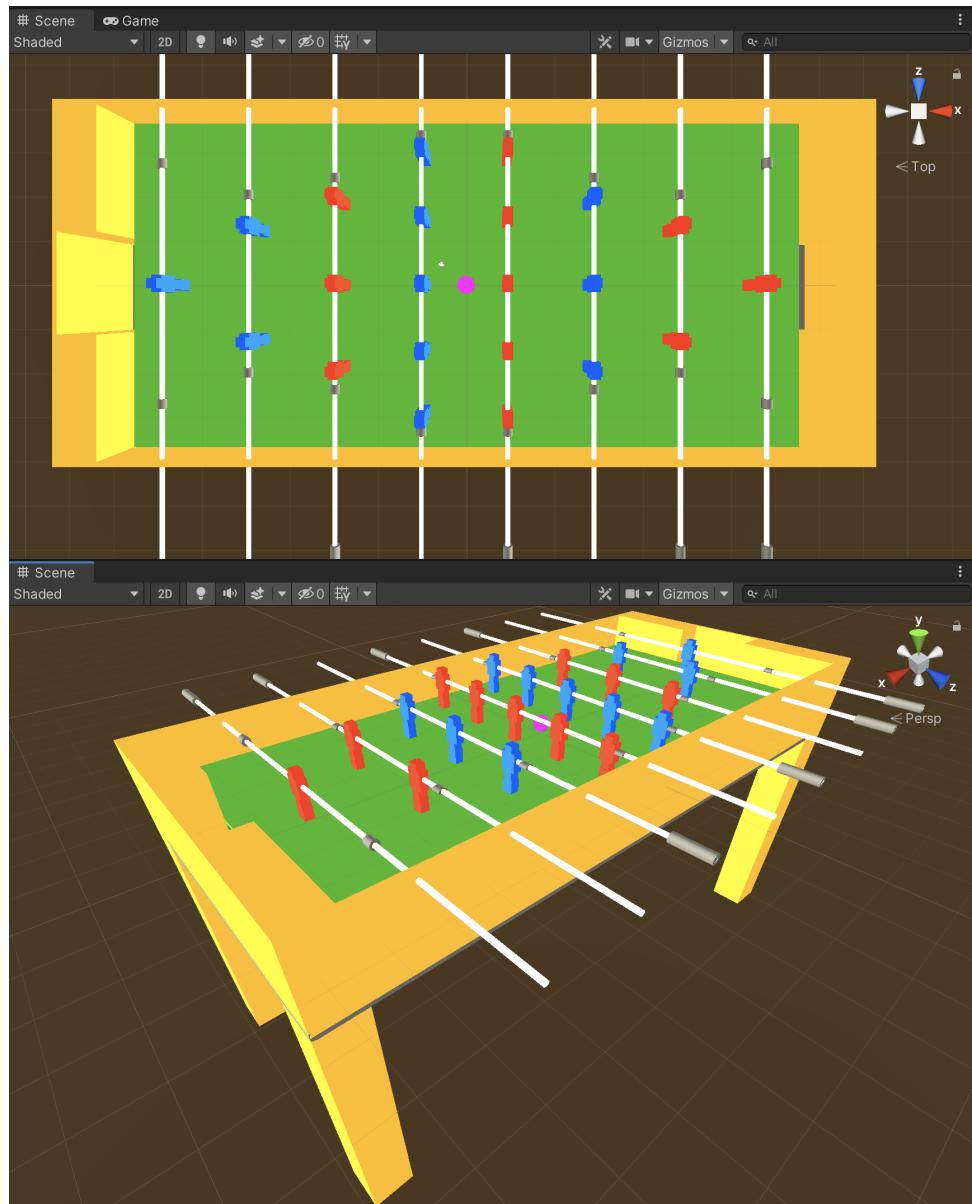


Figura 60 – Imagens da Simulação da Mesa de Pebolim

Na figura 60 é possível visualizar a simulação da mesa virtual desenhada na engine

de jogos Unity, em que os jogadores azuis estão configurados para receber movimentos no modelo PUDM e realizar os movimentos.

Como exemplo da utilização da física, temos na figura 61 a visualização do vetor velocidade atribuído ao objeto da bola, representado por uma linha rosa saindo da bola. Uma força é aplicada na bola movendo ela em direção ao centro da mesa, para evitar que a bola fique presa ou parada em algum canto.

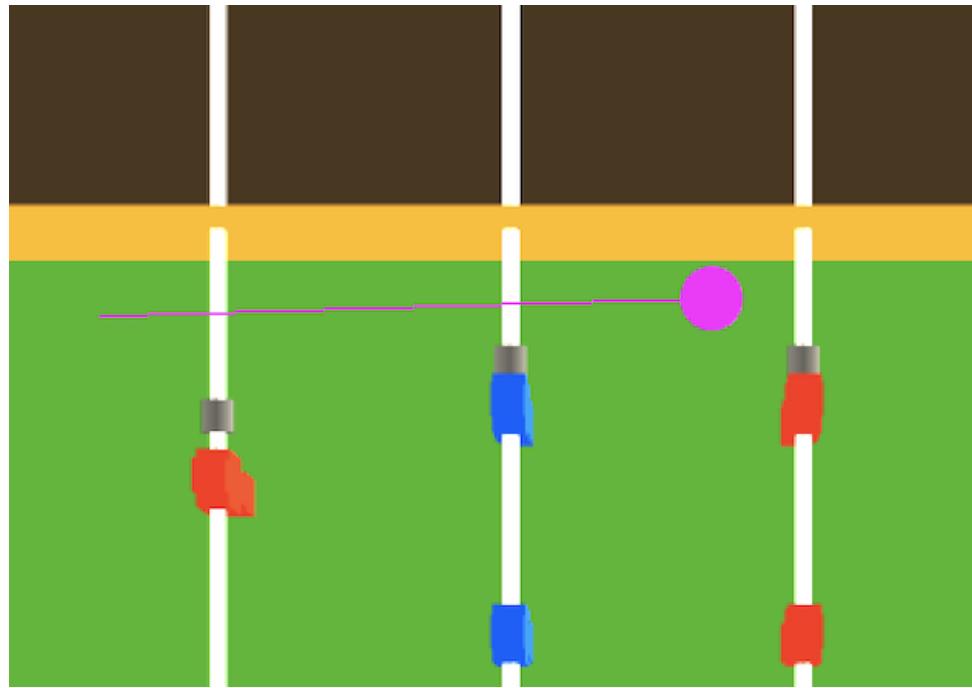


Figura 61 – Movimento da Bola na Simulação

Já na figura 62 o comportamento de chute é exibido, adicionando uma força na direção desejada do chute. Essa força adicionada é representada pela linha amarela. A linha azul conecta o centro do objeto da bola e o centro do objeto do jogador no momento em que o chute ocorreu. E a linha rosa é a mesma citada anteriormente. Observe que a linha permanece por mais tempo do que um *frame* para que possa ser observada, por isso a bola não se encontra na posição marcada pela linha azul.

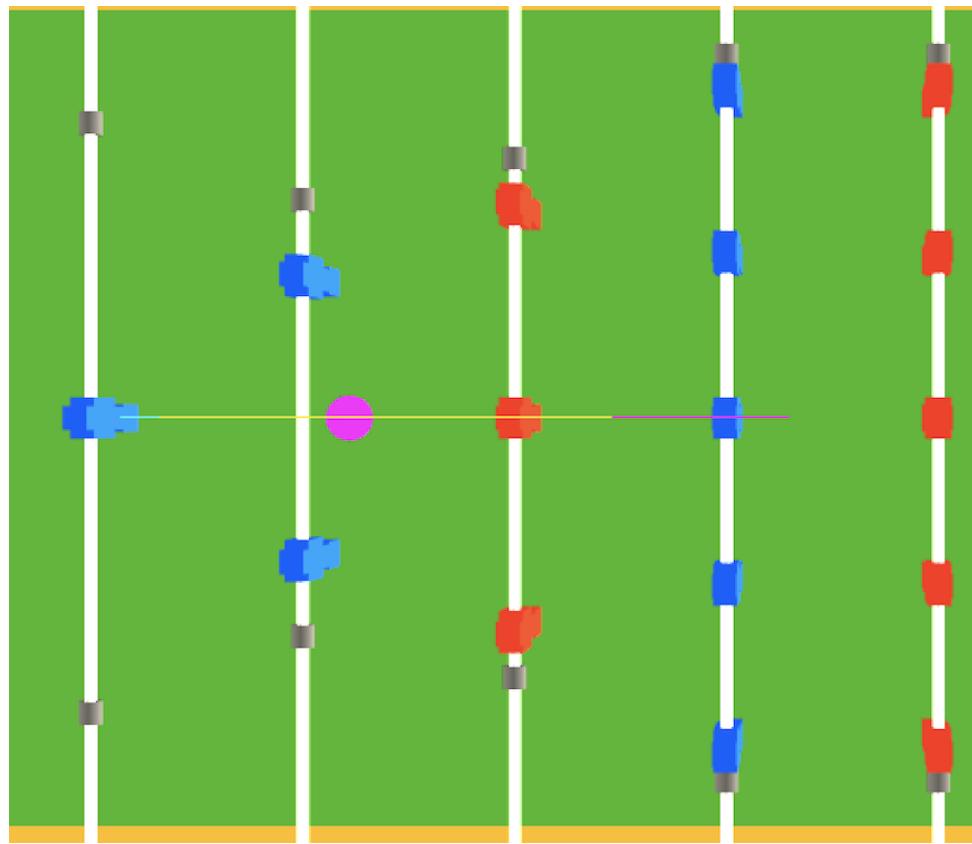


Figura 62 – Chute de um Jogador na Simulação

A simulação é capaz de ser executada conectada à dois Servidores de Decisão, cada um controlando um lado da mesa. Isso foi feito para podermos testar o comportamento dos algorítimos de decisão desenvolvidos, como capturado na figura 63.

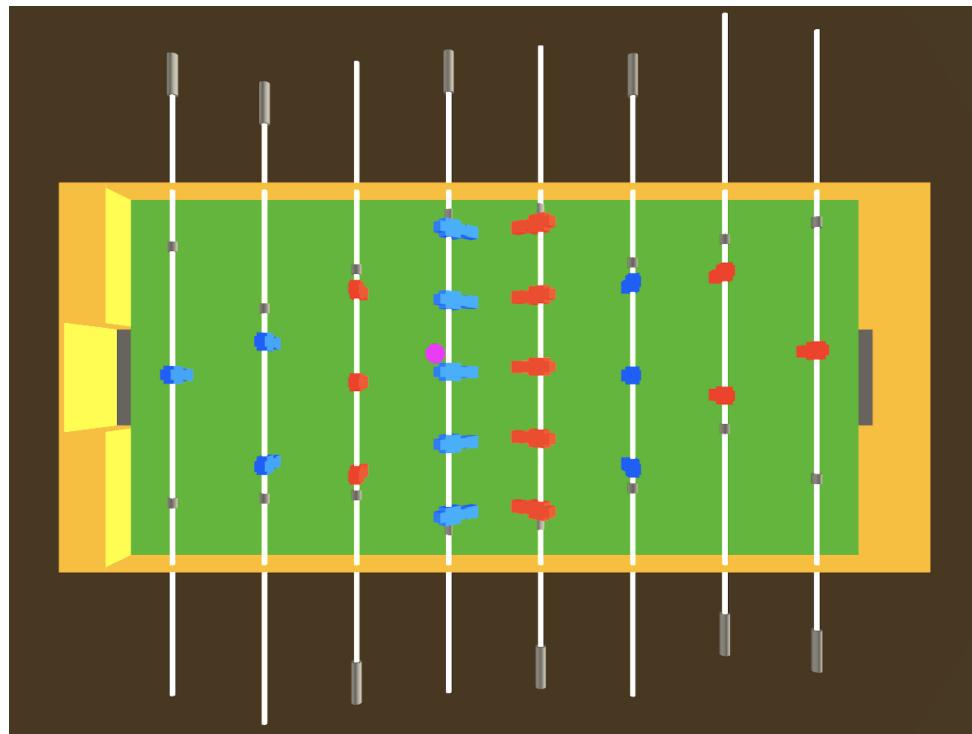


Figura 63 – Demonstração da Simulação executada com dois Servidores de Decisão

Além disso, toda a comunicação feita entre a simulação e o servidor de decisão é realizada utilizando a modelagem de dados PUDM, explicada na Seção 2.5.2.1.

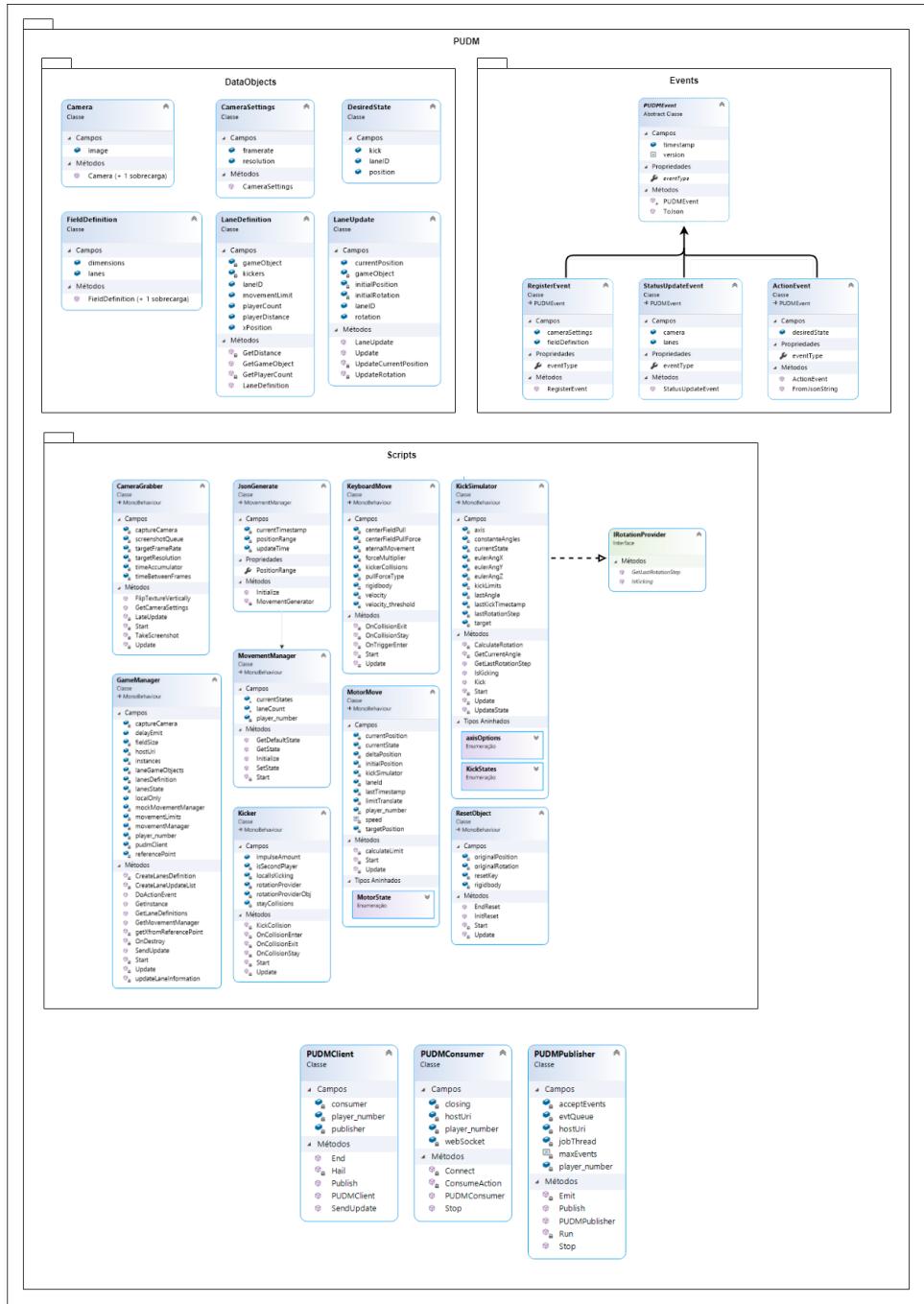


Figura 64 – Diagrama de classe da simulação

O PUDM é implementado através de 2 pacotes, *Data Objects* e *Events*. O pacote *Data Objects* possui as classes relacionadas aos objetos criados na simulação da mesa virtual de pebolim, que por sua vez estes objetos serão utilizados para gerar os eventos implementados no pacote *Events*. O Pacote *Events* possui as classes que configuram o PUDM definido na Seção 2.5.2.1.

Para a parte da simulação foi necessário criar o pacote *Scripts*, que possui as classes que foram usadas para gerenciar os comportamentos dos objetos da mesa de pebolim

virtual, e simular o comportamento dos motores. Esse pacote é essencial para o funcionamento da simulação e contém as classes: KeyboardMove, permite usar o teclado para mover a bola na simulação e realizar chutes; MotorMove, serve para simular o movimento dos motores ao receber o pacote PUDM; CameraGrabber, responsável por capturar os frames da camera, e controla a atualização da imagem do campo virtual; GameManager, responsável por gerenciar o funcionamento do jogo; ResetObject, para reiniciar a posição inicial dos objetos no jogo; Kicker, para configuração da física e realização do chute dos jogadores e KickSimulator, para simular um chute e conseguir realizar testes.

3 Considerações Finais

A ideia do projeto possui potencial comercial e tecnológico. Alguns protótipos foram desenvolvidos por universidade ao redor do mundo e com a implementação de novas tecnologias é uma boa fonte de desenvolvimento científico na área acadêmica.

O mercado de mesas autônomas ainda é extremamente pequeno e portanto existe pouca concorrência. Um protótipo bem sucedido pode ser usado para desenvolver um modelo de produção industrial. Um produto comercial poderá ser usado por jogadores amadores e proprietários de bares e casas de jogos.

Referências

AMERICANAS. *Fonte Alimentação MeanWell SE-1000-24*. 2020. Disponível em: <[BAÚ DA ELETRONICA. *Driver Para Motor de Passo 5A WD-TB6600 - Wotiom*. 2020. Disponível em: <<https://www.baudaelectronica.com.br/driver-para-motor-de-passo-5a-wd-tb6600-wotiom.html>>. Acesso em: 23 Ago. 2020. Citado 3 vezes nas páginas 4, 126 e 127.](https://www.americanas.com.br/produto/213294233/se-1000-24-fonte-alimentacao-meanwell-1000w-saida-24v-41-6a?WT.srch=1&acc=e789ea56094489dff798f86ff51c7a9&epar=bp_pl_00_go_pla_casaecomst_geral_gmv&gclid=Cj0KCQjwt4X8BRCPARIsABmcnOrvEQ0sSDxlmj_6c7j2LYPe4RUELwHJoYpoYG0DhQ8w4MCBjpqGddsaAIN7EALw_wcB&i=5d712b2d49f937f6250d8225&o=5d6e93a46c28a3cb509103ab&opn=YSMESP&sellerid=10428528000110>. Acesso em: 07 out. 2020. Citado 3 vezes nas páginas 4, 127 e 128.</p>
</div>
<div data-bbox=)

BERMEJO, J. A. A. et al. Embedded kernel customization to optimize performance and power management. an application to iot. Citado na página 171.

FILIPE FLOP. *Câmera Compatível Raspberry Pi 5MP*. 2020. Disponível em: <<https://www.filipeflop.com/produto/camera-compativel-raspberry-pi-5mp>>. Acesso em: 23 Ago. 2020. Citado 3 vezes nas páginas 4, 124 e 125.

FILIPE FLOP. *Raspberry Pi 4*. 2020. Disponível em: <<https://www.filipeflop.com/produto/raspberry-pi-4-model-2gb-4gb>>. Acesso em: 23 Ago. 2020. Citado 3 vezes nas páginas 4, 122 e 123.

ITSF. *Livro de Regras*. [S.l.: s.n.], 2008. Citado na página 14.

JUNIOR, M. A. B. de Almeida e Dante de R. *Qualidade de Vida: Evolução dos conceitos e práticas no século XXI*. [S.l.]: IPES, 2010. Citado na página 14.

LIU, H. et al. Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer vision and image understanding*, Elsevier, v. 72, n. 3, p. 271–286, 1998. Citado na página 60.

MERCADO LIVRE. *Tela Lcd Hdmi 10.1 1024x800 Ips Touch Screen Raspberry Pi*. 2020. Disponível em: <https://produto.mercadolivre.com.br/MLB-1622153583-tela-lcd-hdmi-101-1024x800-ips-touch-screen-raspberry-pi-_JM?matt_tool=69265266&matt_word&gclid=CjwKCAjwnef6BRAgEiwAgy8mQWfQasTfomlISRh4wwdKuggydawGm1S6XQV_WUap8I3FhkE67NygTBoCYxkQAvD_BwE&quantity=1>. Acesso em: 23 Ago. 2020. Citado 3 vezes nas páginas 4, 125 e 126.

ROBOCORE. *Fonte Oficial para Raspberry Pi 4*. 2020. Disponível em: <https://www.robocore.net/aceccorios-raspberry-pi/fonte-raspberry-pi-4?gclid=CjwKCAiAnIT9BRAmEiwANaoE1TbvTlc36sU_P89I9N4QDIpWYgJXVvbXUozOT69NnGewok_8_HVsTRoCtzcQAvD_BwE>. Citado 2 vezes nas páginas 4 e 129.

SERVOTRONIX. *NEMA 17, NEMA 23 and NEMA 34 Datasheet Rev. 5.2. 2014.*
Disponível em: <<https://datasheetspdf.com/pdf/1380134/Servotronix/NEMA23/1>>.
Citado 7 vezes nas páginas 2, 6, 15, 16, 17, 117 e 130.

Apêndices

APÊNDICE A – Gerenciamento geral

A.1 Análise Quantitativa

A.1.1 Matriz de Probabilidade

Probabilidade	Intervalo	Peso
Muito Baixa	Menor de 20%	1
Baixa	de 21% a 40%	2
Moderada	de 41% a 60%	3
Alta	de 61% a 80%	4
Muito Alta	Acima de 81%	5

Tabela 8 – Matriz de Probabilidade

A.1.2 Matriz de Impacto

Probabilidade	Descrição	Peso
Muito Baixo	Impacto não significativo ao projeto	1
Baixo	Impacto de pouca influência no projeto	2
Moderado	Impacto notável com poucas consequências	3
Alto	Impacto crítico, compromete o andamento do projeto	4
Muito Alto	Impacto extremo, impossibilita a continuidade do projeto	5

Tabela 9 – Matriz de Impacto

A.1.3 Matriz de Prioridade

A matriz de prioridades é realizada a multiplicação da probabilidade do risco acontecer. Na coluna se tem os pesos da probabilidade e na linha de impacto.

Probabilidade/Impacto	Muito Baixo	Baixo	Moderado	Alto	Muito Alto
Muito Baixa	1	2	3	4	5
Baixa	2	4	6	8	10
Moderada	3	6	9	12	15
Alta	4	8	12	16	20
Muito Alta	5	10	15	20	25

Tabela 10 – Matriz de Prioridade

Os níveis de definição das prioridades, foi definido na seguinte tabela:

Nível de Prioridade	Intervalo
Baixa	1 - 5
Média	6 - 14
Alta	15 - 25

Tabela 11 – Níveis de Prioridades

A.2 Escopo

A.2.1 Descrição do Produto

O Autonomous Foosball é uma máquina que objetiva dar autonomia aos jogadores amadores de pebolim. A mesa é projetada do zero, tendo como referência uma mesa padrão de pebolim e com as alterações necessárias para a automação, permitindo uma estrutura compacta. O projeto tem como intuito localizar a posição da bola na mesa e, a partir de um sistema de decisão, controlar as manoplas da mesa de forma condizente, tendo como objetivo marcar o gol. Por fim, o sistema possui um monitor para mostrar o placar do jogo para o usuário.

A.2.2 Lista É/Não é

Autonomous Foosball É	Autonomous Foosball Não é
Autônomo	Conduzido manualmente
Feito para jogar pebolim	Feito para jogar qualquer outro tipo de jogo de salão
Conectado diretamente na tomada	Movido à bateria
As decisões de movimento dos eixos da mesa são feitas com base nas jogadas do usuário	As decisões de movimento dos eixos da mesa são pré definidas
Programado para detectar a bola no plano da mesa	Programado para localizar a bola no eixo perpendicular à mesa
Move as manoplas da mesa nos sentidos longitudinal e no sentido rotacional	Move as manoplas da mesa nos três eixos
O suporte da câmera é acoplado à mesa	O suporte da câmera é independente à mesa
A bola é colocada na mesa manualmente	A bola é colocada na mesa automaticamente

Tabela 12 – Lista É/Não é

A.2.3 Premissas e Restrições

- Antes da utilização do equipamento, é importante verificar a posição inicial e o bom estado dos motores e manoplas;

- O usuário deve se certificar de não interferir na leitura da mesa pela câmera, não sujando sua lente ou bloqueando sua visão;
- O usuário não deve modificar nenhum componente da estrutura independente de trilhos e motores;
- O proprietário do equipamento deve se certificar da calibração correta dos componentes periodicamente;
- Caso seja de interesse o uso dos dois lados da mesa para dois jogadores, a desmontagem dos componentes deve ser feita seguindo os passos de instrução para evitar danos.

A.2.4 Termo de Abertura de Projeto

A.2.4.1 Introdução

O Termo de Abertura de Projeto (TAP) objetiva formalizar o início do projeto Autonomous Foosball. A partir deste, é concedida ao gerente geral e ao gerente de qualidade a autoridade para coordenar as atividades relacionadas à confecção do projeto. Assim, este documento serve de entrada às próximas etapas do projeto: gerenciamento, identificação e controle de requisitos, controle de custos e análise de mercado.

A.2.5 Propósito e Justificativa

Desde o início do século, tanto profissionalmente quanto por lazer, os jogos de mesa como sinuca, tênis de mesa e pebolim se tornam cada vez mais populares dentre todas as regiões e camadas da sociedade. O objetivo do Autonomous Foosball é proporcionar uma experiência de jogo autônoma para jogadores amadores de pebolim, os quais terão a oportunidade de experimentar uma mesa de jogo tecnológica e a opção de não necessitar de um segundo jogador humano para jogar.

A.2.5.1 Vantagens e desvantagens da substituição do homem por uma máquina

Ao vivenciar o jogo competitivo com outro jogador humano, experimentamos não só o prazer do jogo, mas também da interação humana. Com o passar dos anos, a interação homem e máquina vai se tornar mais frequente nas situações comuns do dia-a-dia e não apenas durante um jogo de mesa. Pensando nisso e em como a tecnologia evolui constantemente a cada momento, precisamos nos adequar e evoluir junto a ela.

Em termos de jogabilidade, o jogador robô nunca será capaz de reproduzir fielmente o que o jogador humano pode fazer, pelo simples motivo de que o robô é programado para agir de uma determinada forma e em uma determinada situação. Enquanto o jogador humano, provavelmente, nunca repetirá um mesmo movimento duas vezes, já que

somos sempre autuados a raciocinar e agir de acordo com o momento em que estamos, dependendo do que vemos, sentimos e ouvimos. Porém, a mesa oferece a possibilidade de jogar sem a presença de um segundo jogador, o que, em muitas situações, pode ser muito oportuno.

A.2.6 Objetivos do Produto

A.2.6.1 Objetivo Geral

O objetivo geral é projetar uma máquina de pebolim autônoma modular.

A.2.6.2 Objetivos Específicos

A máquina a ser desenvolvida também terá como objetivos:

- proporcionar uma experiência de jogo real para jogadores amadores;
- possibilitar a experiência de jogo sem a presença de um segundo jogador;
- proporcionar diferentes níveis de dificuldade para contemplação de vários públicos.

A.2.7 Requisitos de Alto Nível

O projeto em questão visa automatizar o processo de jogo, se aproximando o máximo possível da realidade. Com isto, foram levantados os seguintes requisitos de alto nível:

- o jogador robô deve ser capaz de atacar e defender;
- o sistema de rastreamento deve ser capaz de localizar a bola, desde que ela esteja no plano da mesa, e conferir à leitura uma resposta adequada pelo Decision Server;
- a mesa como um todo deve ser capaz de operar sem a necessidade de nenhuma interferência humana durante o jogo;
- a mesa deve mostrar o placar do jogo durante as partidas.

A.2.8 Riscos Iniciais

Os principais riscos do projeto envolvem sobretudo:

Riscos	Plano de Ação
A desistência de algum membro	Averiguar a situação dos membros no semestre, adaptar os horários e distribuir as tarefas entre os membros remanescentes para que nenhum membro fique sobrecarregado
O tamanho da equipe, que dificulta a comunicação e o gerenciamento da equipe	Manter uma boa integração da equipe, fazer o acompanhamento das atividades dos membros, assim como utilizar um meio de comunicação que todos usam constantemente e marcar reuniões no horário da aula
Planejamento errôneo ou incompleto	Revisar o planejamento do projeto de forma iterativa e recursiva para que variáveis ignoradas ou não encontradas anteriormente sejam consideradas, tendo em mente os pontos de controles pré-estabelecidos
Integração entre áreas de estruturas, software e motores	Manter uma comunicação efetiva entre as equipes responsáveis pelos subsistemas e considerando o impacto causado por mudanças de projeto em todos os outros subsistemas

Tabela 13 – Riscos Iniciais do Projeto

A.2.9 Stakeholders

A.2.9.1 Equipe do Projeto

Nome	Engenharia	Email
Lucas Marques Vilela	Engenharia Aeroespacial	lukevilela@yahoo.com.br
Gustavo Saraiva Lopes	Engenharia Aeroespacial	gustavosaraiva777@gmail.com
Lucas Alves Torres	Engenharia Automotiva	torresalveslucas@gmail.com
Antonio Luis Lima Junior	Engenharia Automotiva	junior-cars@hotmail.com
Daniel Maike Mendes Gonçalves	Engenharia de Software	danmke@hotmail.com
Joberth Rogers Tavares Costa	Engenharia de Software	joberth.rogers18@gmail.com
Guilherme Guy de Andrade	Engenharia de Software	guilhermeguy349@gmail.com
Marco Antônio de Lima Costa	Engenharia de Software	markinlimac@gmail.com
João Pedro de Lima Pereira	Engenharia de Software	jplpereira@gmail.com
Vinícius Guerra e Ribas	Engenharia de Energia	viniciusgribas@gmail.com
Matheus Lino do Amaral	Engenharia de Energia	lino9595@gmail.com
Rafael Marinho Nunes	Engenharia Eletrônica	rafael.etb2@gmail.com
Natanaele do Nascimento Balica	Engenharia Eletrônica	natanaelebalic@gmail.com
Nauam Victor Reis de Oliveira	Engenharia Eletrônica	nauamvictor@outlook.com

Tabela 14 – Equipe do Projeto

A.2.9.2 Professores

- Alex Reis (Engenharia de Energia)
- José Felício da Silva (Engenharia Eletrônica)
- Rhander Viana (Engenharia Automotiva)
- Ricardo Matos Chaim (Engenharia de Software)
- Paolo Gessini (Engenharia Aeroespacial)

A.2.9.3 P blico Alvo

O p blico alvo do projeto   amplo, podendo atender a estabelecimentos que possuem sal o de jogos, como bares e hoteis, bem como estabelecimentos de jogos de fliperama. Além disso, o produto tem potencial de atender o consumidor individual que possua um poder aquisitivo elevado, visto que o pre o de uma mesa de pebolim tradicional est a

a partir de R\$ 500,00 e sua automatização acrescentaria significativamente no valor final (estimativa do custo).

A.2.10 Cronograma e Marcos

O cronograma do projeto se dá ao início do retorno do semestre letivo da disciplina de Projeto Integrador 2. Este teve seu marco inicial no dia 19/08/2020, data de alinhamento das atividades e ajustes nos grupos, e a data de finalização de 04/12/2020, a qual marca a apresentação da FIT/FGA. Entre essas datas acontecerão três pontos de controle, em que deverão mostrar progressivamente a construção da solução apresentada pelo grupo, passando por toda a documentação de gerenciamento e de detalhes técnicos. Por fim a apresentação do manual de montagem e uso marcam a finalização do projeto.

Ponto de Controle	Data	Resumo
Ponto de Controle 01	18/09/2020 a 02/10/2020	Entendimento do problema, concepção da arquitetura e o plano de gerenciamento do grupo
Ponto de Controle 02	16/10/2020 a 30/10/2020	Os critérios do projeto, esquemático 3D, sistema de alimentação, diagramas esquemáticos de circuitos eletrônicos, diagramas de classe, diagramas de caso de uso e por fim o plano de construção dos subsistemas
Ponto de Controle 03	13/11/2020 a 27/11/2020	Demonstrar, para cada subsistema, a lista de materiais completa, plano de testes, plano de fabricação e montagem. Apresentar a documentação completa do projeto com diagrama de integração dos subsistemas, manual de montagem e uso, vídeo de propaganda da solução e link do repositório do projeto.
Feira de Inovação e Tecnologia da FGA	04/12/2020	Apresentação de projetos na FIT/FGA On Line

Tabela 15 – Cronograma e Marcos

A.3 Gerência

A.3.1 Metodologia

Nesta sessão abordaremos as metodologias utilizadas para o gerenciamento do projeto. Como ferramenta principal, utilizamos o Microsoft Teams disponibilizado pela instituição.

A.3.2 Microsoft Teams

Nos tempos de pandemia, o projeto vem sendo realizado totalmente de forma não presencial. Desta forma, utilizamos o Microsoft Teams tanto para reuniões, quanto para planejamento e distribuição de atividades.

De acordo com os entregáveis dos Pontos de Controle, utilizamos o Planner do Teams, ferramenta muito parecida com o Trello, para a aferição de atividades e coordenação de datas. Concomitante ao uso do planejamento inicial e do EAP criados, dividimos as entregas em semanas de acordo com cada requisito dos Pontos de Controle.

A.3.3 Organização da Equipe

A equipe deste projeto é composta apenas por alunos da Universidade de Brasília Campus Gama (UnB - FGA), divididos nos seguintes cursos de Engenharia: Software, Energia, Aeroespacial, Automotiva e Eletrônica. Para melhor usar o conhecimento de cada área, estes foram separados em equipes. A seguir a composição de cada equipe:

- Software
- Estrutura
- Motores

A formação da equipe é formada por 5 membros de Engenharia de Software, 2 de Energia, 2 de Aeroespacial, 2 de Automotiva e 3 membros de Eletrônica, totalizando assim 14 membros. Esses membros foram divididos em 3 sub-equipes de acordo com suas especializações e as suas composições são as seguintes: 5 membros na sub-equipe de Software, 4 membros na sub-equipe de Estrutura e 5 membros na sub-equipe de Motores.

Cada área possui um técnico ou gerente que é responsável por averiguar as atividades internas. Seguindo a definição de responsáveis da disciplina de Projeto Integrador de Engenharias 2, temos um Coordenador geral, um Diretor de qualidade, três Diretores técnicos e os Desenvolvedores.

A.3.4 Integrantes e Suas Responsabilidades

A.3.4.1 Diretores e Técnicos

Nome	Responsabilidade
Gustavo Saraiva Lopes	Coordenador Geral
Lucas Marques Vilela	Diretor de Qualidade
Guilherme Guy de Andrade	Diretor Técnico de Software
Antônio Luis Lima Júnior	Diretor Técnico de Estruturas
Vinicio Guerra e Ribas	Diretor Técnico de Motores

Tabela 16 – Diretores e Técnicos

A.3.4.2 Equipes e Responsabilidades

Equipe	Nome	Cargo	Responsabilidades
Software	Guilherme Guy	Diretor Técnico	Gerenciamento da equipe e auxiliar no desenvolvimento do software
	Daniel Maike	Desenvolvedor	Construir o software do mecanismo
	Joberth Rogers	Desenvolvedor	Construir o software do mecanismo
	Marco Antônio de Lima	Desenvolvedor	Construir o software do mecanismo
Aeroespacial	João Pedro de Lima Pereira	Desenvolvedor	Construir o software do mecanismo
	Gustavo Saraiva	Coordenador Geral	Coordenar as atividades do projeto, bem como verificar se todas as atividades estão sendo entregues no prazo
	Lucas Vilela	Diretor de Qualidade	Auxiliar o coordenador geral e verificar a qualidade dos artefatos que estão sendo entregues
Automotiva	Antonio Junior	Diretor Técnico	Gerenciamento da equipe e auxiliar no desenvolvimento da estrutura
	Lucas Torres	Desenvolvedor	Auxiliar no desenvolvimento da estrutura do projeto
Eletrônica	Rafael Marinho	Desenvolvedor	Trabalhar com a parte da placa controladora e motores do mecanismo
	Natanaele do Nascimento	Desenvolvedora	Trabalhar com a parte da placa controladora e motores do mecanismo
	Nauam Victor Reis	Desenvolvedor	Trabalhar com a parte da placa controladora e motores do mecanismo
Energia	Vinícius Guerra	Diretor Técnico	Gerenciamento da equipe e auxiliar no desenvolvimento da parte eletrônica e energética do projeto
	Matheus Lino	Desenvolvedor	Trabalha com a parte de motores do projeto e auxiliar a equipe de estruturas

Tabela 17 – Equipes e Responsabilidades

A divisão das equipes pode ser vista de forma visual a seguir:

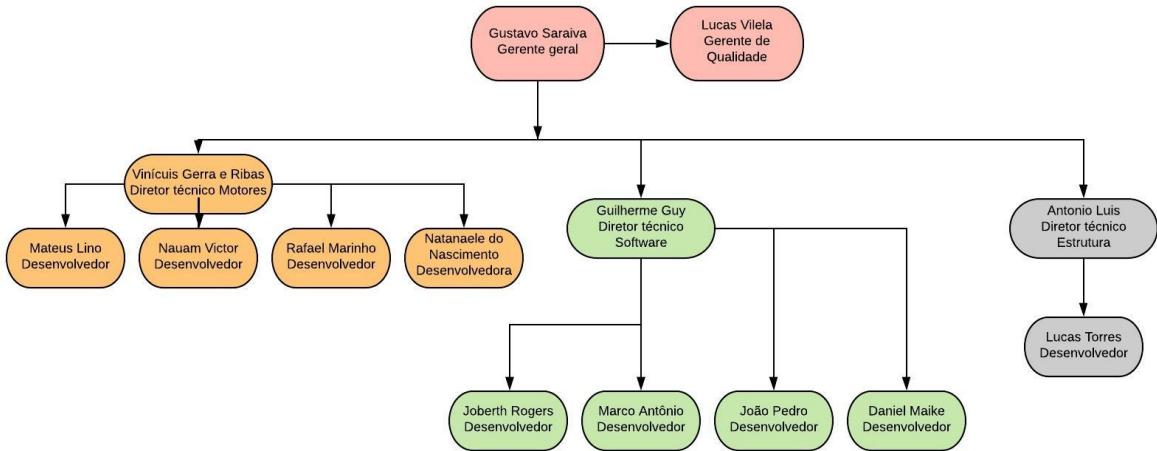


Figura 65 – Diagrama de membros

A.3.5 Gerenciamento de Comunicação

O gerenciamento de comunicação da equipe foi feito através das ferramentas apresentadas na tabela 18. As reuniões ocorreram durante o tempo de aula. Devido à pandemia, todas as reuniões foram não presenciais.

Ferramentas	Função
Telegram	Usada para comunicação rápida, alinhamento sobre tarefas e marcar reuniões.
Microsoft Teams	Usada para reuniões não presenciais entre o grupo e as sub-áreas.
Google Drive	Armazenamento de documentos, planilhas, imagens entre outros arquivos.
Microsoft Planner	Controle de atividades, prazos e entregas no formato Kan-Ban. Aplicativo online e colaborativo.
GitHub	Usado para armazenar e disponibilizar os arquivos referentes ao desenvolvimento de software.
Overleaf	Formatação dos documentos e compilação dos relatórios do projeto.

Tabela 18 – Ferramentas de comunicação

A.3.6 Atas

A.3.6.1 Reunião 1

Informações:

Data: 19/08/2020

Horário: 17h

Plataforma: Microsoft Teams

Pautas

- Levantamento de requisitos iniciais;
- Planejamento geral de execução de atividades semanais até o PC1;
- Criação e definição da subdivisão das áreas.

A.3.6.2 Reunião 2

Informações:

Data: 26/08/2020

Horário: 17h

Plataforma: Microsoft Teams

Pautas

- Entendimento de todos os membros acerca da problemática de todas as áreas;
- Escolha e listagem dos componentes;
- Definição dos parâmetros a serem seguidos no decorrer do projeto.

A.3.6.3 Reunião 3

Informações:

Data: 02/09/2020

Horário: 17h

Plataforma: Microsoft Teams

Pautas

- Estimativa de custos geral;
- Redefinição da solução;
- Controle de prazos;
- Planejamento para construção do primeiro relatório.

A.3.6.4 Reunião 4

Informações:

Data: 09/09/2020

Horário: 17h

Plataforma: Microsoft Teams

Pautas

- Alinhamento da equipe diante das soluções específicas;
- Discussão sobre os entregáveis do Ponto de Controle 1;
- Unificação e revisão dos documentos para finalização do relatório.

A.3.7 Gerenciamento de Qualidade do Projeto

O gerenciamento de qualidade foi construído a partir do cronograma planejado e das tarefas distribuídas ao longo das semanas. O gerente de qualidade foi instaurado com o intuito de monitorar e garantir o bom funcionamento do trabalho em equipe e das atividades de cada sub-setor, de forma a manter, o máximo possível, o andamento do projeto em concordância com cronograma.

O controle de qualidade será realizado utilizando um fluxograma de processos gerais e específicos de cada setor. Visando garantir a inspeção constante do processo real em relação ao teórico, de forma a buscar sempre a análise de necessidade de mudança ou de mudança de caminhos no decorrer do projeto.

O registro e as avaliações serão feitas em relatórios de desenvolvimento com descrição e recomendações para problemas enfrentados em cada ponto de controle. O relatório contará com a utilização de Cartas de controle, relacionando os parâmetros relevantes de cada fase do projeto, sempre visando uma melhor análise. O gerenciamento de qualidade do projeto se baseia também nos riscos envolvidos no projeto na qual o gerente de qualidade deve ser o responsável por acompanhar atividades críticas e de grande impacto.

A.3.8 Identificação dos Riscos

A.3.8.1 Análise SWOT

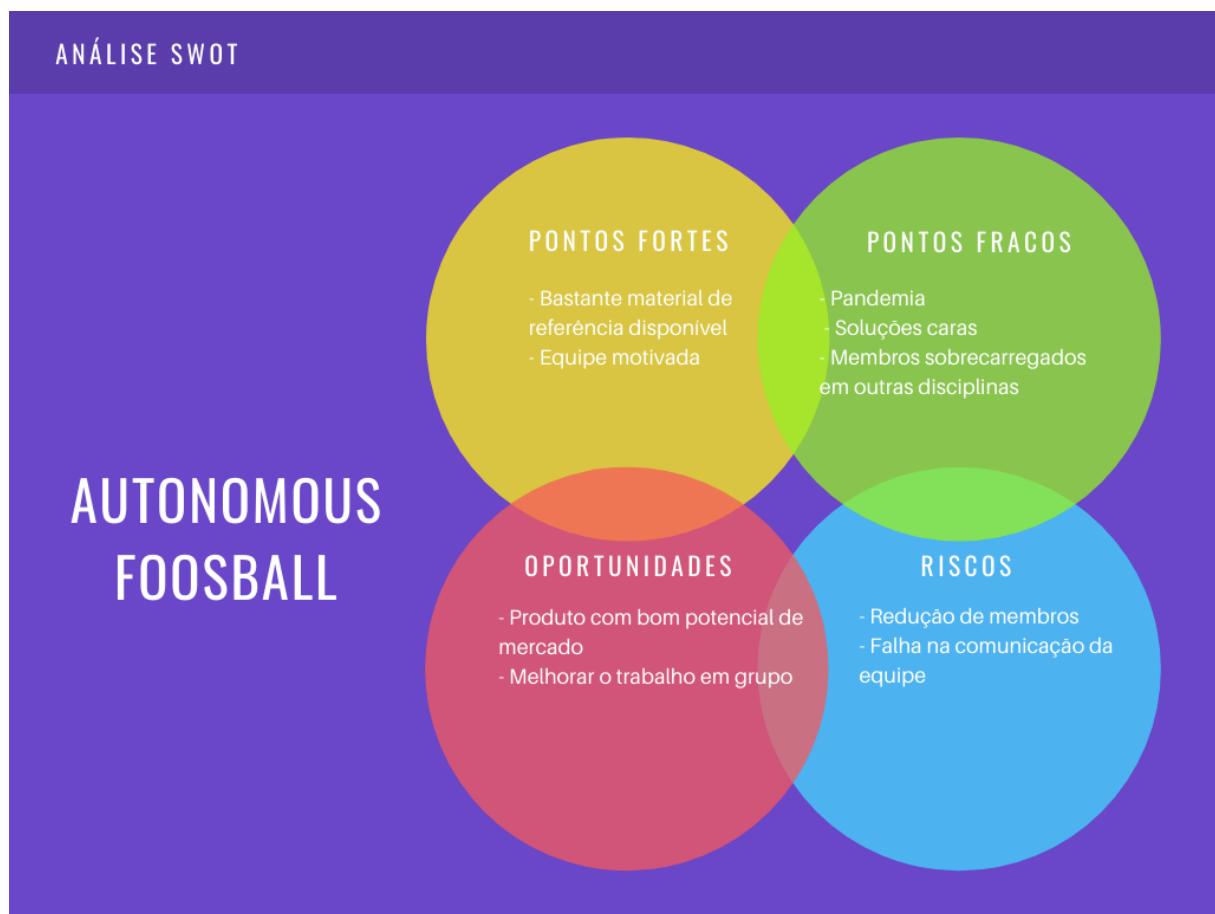


Figura 66 – Análise SWOT

A.3.9 Análise Qualitativa

A.3.9.1 Diagrama de Ishikawa

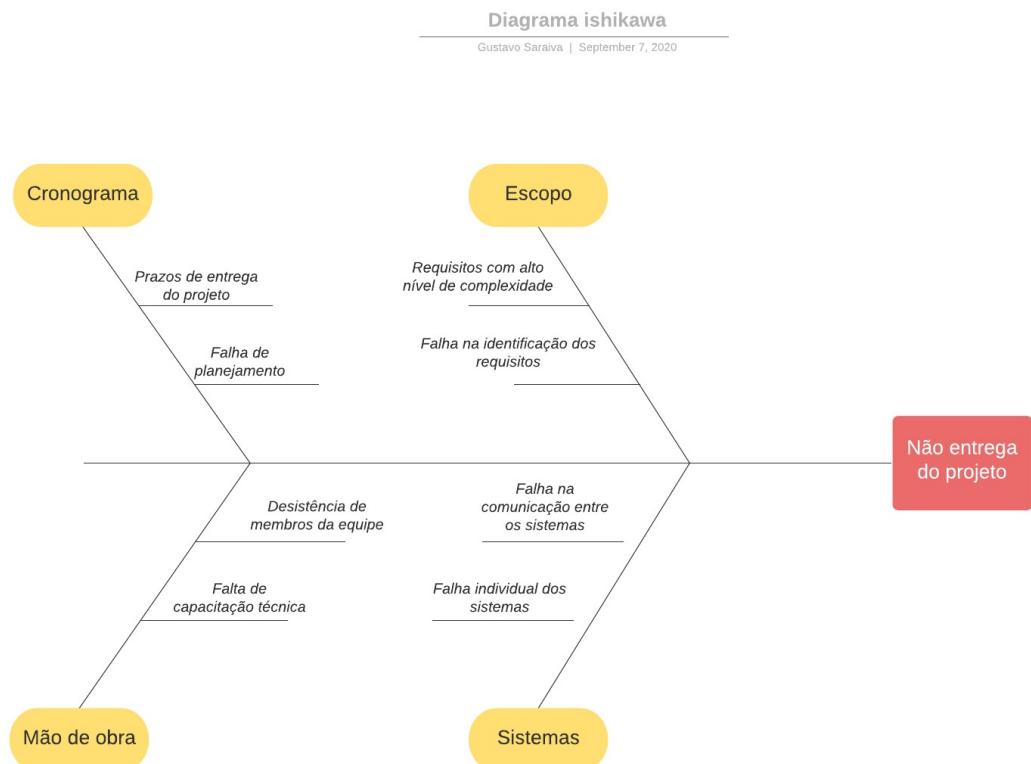


Figura 67 – Diagrama de Ishikawa

A.3.9.2 Descrição do diagrama de Ishikawa

O formato do diagrama se assemelha a uma espinha de peixe e sua função é levantar todas as causas e efeitos do projeto, no intuito de identificar facilmente a raiz de um problema.

A.3.10 Registro dos Riscos

A.3.10.1 Risco Gerais

Número	Risco	Causa	Efeito
RN1	Desistência de membros da equipe	Sobrecarga em outras disciplinas. Possibilidade de retirada da disciplina até o último dia letivo	Sobrecarga dos membros restantes. Possibilidade de não entrega do projeto.
RN2	Falta de capacitação técnica	Inexperiência dos membros da equipe	Qualidade de entrega ruim
RN3	Falha de planejamento	Má organização ou erro no levantamento de requisitos	Organização e sequenciamento de atividades falhos, gerando erros e atrasos no projeto
RN4	Falha na comunicação entre os sistemas	Administração das interfaces dos subsistemas deficiente. Falha na comunicação entre os núcleos técnicos	Não entrega do projeto
RN5	Falha individual dos sistemas	Falta de comprometimento, acompanhamento ou gestão de um subsistema	Não entrega de um subsistema e, consequentemente, não entrega do projeto
RN6	Requisitos com alto nível de complexidade	Escolha de projeto feita anterior a uma análise cuidadosa dos requisitos	Atraso ou falha na entrega do projeto
RN7	Falha na identificação dos requisitos	Inexperiência da equipe	Soluções com pouca qualidade. Produto final não atende às expectativas dos stakeholders

Tabela 19 – Riscos Gerais

A.3.10.2 Resposta aos riscos gerais

Número	Ação	Resposta	Responsável
RN1	Aceitar	Buscar formas de compensar a falta de um membro ou redefinir os requisitos do projeto	Gerente geral e Diretores técnicos
RN2	Mitigar	Buscar formas de balancear o conhecimento dentro do núcleo técnico	Gerente geral e Diretores técnicos
RN3	Mitigar	Decompor o trabalho necessário para a entrega dos produtos e estabelecer um cronograma	Gerente geral
RN4	Mitigar	Reforçar acompanhamento dos gerentes geral e de qualidade para manutenção da comunicação do projeto	Gerente geral
RN5	Mitigar	Acompanhamento da qualidade das soluções encontradas e a satisfação adequada aos requisitos	Gerente geral, diretor de qualidade e diretores técnicos
RN6	Aceitar	Buscar ajuda com alunos mais experientes ou com o corpo de professores disponível	Todos os membros do projeto
RN7	Reanalisar	Promover seções de brainstorming para reanálise dos requisitos gerais e específicos do projeto	Gerente geral diretores técnicos

Tabela 20 – Resposta aos riscos gerais

A.3.10.3 Riscos da Estrutura

Número	Risco	Causa	Efeito
RN8	Colapso do suporte da câmera	Montagem inadequada do suporte da câmera. Dimensionamento inadequado	A câmera pode ser danificada. O sistema não funcionará
RN9	Carga excessiva nos motores	Montagem inadequada dos motores	Diminuição da vida útil. O sistema não funcionará corretamente
RN10	Queda dos trilhos	Os trilhos são parafusados inadequadamente	Desacoplamento de parte da estrutura. O sistema não funcionará. Dano aos motores
RN11	A câmera não detecta a bola apropriadamente	Suporte da câmera desalinhado. Altura inadequada do suporte da câmera	O sistema não funcionará corretamente. Os eixos da mesa não serão posicionados corretamente
RN12	Vibração excessiva do sistema	Montagem inadequada. Dimensionamento errado dos componentes	Diminuição da vida útil do sistema. Experiência do usuário comprometida
RN13	Quebra dos eixos da mesa	Falha por fadiga. Deformação plástica	O sistema não funcionará. Detritos podem atingir o usuário. Os motores podem ser danificados
RN14	Pane do sistema. Travamento dos mecanismos	Objeto estranho entre a cremalheira e a engrenagem	Danos à estrutura
RN15	Falta de informação dos outros subsistemas	Gestão de comunicação da equipe ruim	Dimensionamento errado da estrutura

Tabela 21 – Riscos de estruturas

A.3.10.4 Resposta aos riscos de estruturas

Número	Ação	Resposta	Responsável
RN8	Prevenir	Checar o dimensionamento. Preparar procedimento de montagem	Equipe de estruturas
RN9	Prevenir	Preparar procedimento de montagem	Equipe de estruturas
RN10	Prevenir	Checagem da montagem dos trilhos	Equipe de estruturas
RN11	Prevenir	Checagem do dimensionamento da altura da câmera e do alinhamento com a mesa	Equipe de estruturas
RN12	Prevenir	Checagem do dimensionamento dos componentes e realização de simulação estrutural	Equipe de estruturas
RN13	Prevenir	Checagem do dimensionamento adequado das cargas sofridas pelos eixos da mesa	Equipe de estruturas
RN14	Prevenir	Checagem dos componentes dos motores antes do início do jogo	Equipe de estruturas
RN15	Prevenir	Garantir comunicação adequada dos subsistemas	Equipe de estruturas

Tabela 22 – Resposta aos riscos de estruturas

A.3.10.5 Riscos Equipe de Motores - Engenharias de Energia e Eletrônica

Número	Risco	Causa	Efeito
RN16	Mal dimensionamento de motores	Falta de testes práticos	Possibilidade de o motor não ser aplicável no projeto
RN17	Mal dimensionamento dos trilhos	Falta de testes práticos	Possibilidade de o trilho não ser aplicável no projeto
RN18	Mal funcionamento da câmera	Falta de requisitos - Distância da câmera até a mesa, qualidade das imagens e transferência dos dados para IA	Erro na interpretação da IA
RN19	Atraso de resposta dos motores	Delay de comunicação entre IA e o conjunto microcontrolador/microprocessador.	Baixo desempenho da máquina no jogo.
RN20	Mal dimensionamento do microcontrolador	Falta de requisitos - Erro na definição da quantidade de portas que serão utilizadas; a distância entre os drivers e motores.	Aumento do custo do projeto.

Tabela 23 – Riscos equipe de motores

A.3.10.6 Resposta aos Riscos Equipe de Motores

Número	Ação	Resposta	Responsável
RN16	Mitigar	Ampliar possibilidades de motores aplicáveis no projeto	Equipe de motores (Eletrônica e Energia)
RN17	Mitigar	Ampliar possibilidades de trilhos aplicáveis no projeto	Equipe de motores (Eletrônica e Energia)
RN18	Mitigar	Interação entre as equipes e definir os requisitos juntos	Eletrônica, Software e Estrutura
RN19	Mitigar	Investigar a causa do delay e otimizar a resposta dos dados	Eletrônica e Software
RN20	Mitigar	Escolher a melhor configuração, para não ter muito gasto com placas e nem ter fios com alta extensão; Definir a quantidade de portas necessárias para o projeto, a distância entre os drivers.	Eletrônica

Tabela 24 – Resposta aos riscos equipe de motores

A.3.10.7 Riscos da Equipe de Software

Categoria	Subcategoria	Causa	Riscos	Descrição	Cujo impacto é
Técnico	Qualidade	Qualidade do algoritmo	RN21	Algorítimo de inteligência artificial e visão computacional não produz resultados adequados	As respostas dos algorítmicos não serão as ideais em relação ao esperado
	Tecnologia	Ambiente de desenvolvimento	RN22	Complexidade em configurar o ambiente de desenvolvimento de algumas ferramentas a serem utilizadas	Atraso na entrega de tarefas
	Qualidade	Qualidade de simulação	RN23	Simulação não ser suficiente para validar a inteligência artificial	Falha na validação do projeto de inteligência artificial e visão computacional
	Complexidade	Aumento do escopo do projeto	RN24	Novas Funcionalidades no Roadmap	Atraso ou não entrega de alguma funcionalidade
	Qualidade	Desempenho do algoritmo	RN25	Algoritmos de inteligência artificial ou de visão computacional consome mais recursos do que disponíveis no sistema embarcado	Demora na resposta das ações do sistema, prejudicando a experiência do usuário da mesa de pebolim
	Tecnologia	Dificuldade em integrar o software com o hardware	RN26	Adaptação do software junto ao hardware descrito pelo time de motores	A sobrecarga de ambos os times em definir novas estratégias para mitigar o problema

Tabela 25 – Riscos da Equipe de Software

Gerencia- mento	Planejamento	Falha na comunicação da sub-equipe de Software, devido a disciplina ser realizada remotamente	RN27	Devido a dificuldade de comunicação técnicas de alguns membros do grupo, ou até mesmo falta de comunicação entre as frentes	Discrepância entre os modelos propostos e a realização final do software
	Planejamento	Sobrecarga de membros da sub-equipe de software	RN28	Membros trabalharem mais que os outros no desenvolvimento do software, além da expectativa alta pelas entregas da área de software	Baixa motivação da equipe, atrasos em entregas
	Planejamento	Dificuldade em enxergar o escopo final e aplicar os modelos de forma adequada	RN29	Com as diversas possibilidades e um escopo aberto do projeto, pode ocorrer a priorização e criação de funcionalidades que não seriam importantes para o Roadmap	Gasto de tempo dos desenvolvedores em funcionalidades não necessárias

Tabela 26 – Continuação dos Riscos Negativos da Engenharia de Software

A.3.10.8 Resposta aos Riscos da Equipe de Software

Número	Ação	Resposta	Responsável
RN21	Prevenir	Estudar as possibilidades de algoritmos que podem ser utilizados no projeto	Equipe do Servidor de Decisão
RN22	Mitigar	Utilizar a containerização da aplicação	Equipe de Software
RN23	Mitigar	Estudar formas de otimizar a simulação	Equipe de Software
RN24	Prevenir	Priorizar as funcionalidades realmente necessárias para o projeto	Equipe de Software
RN25	Prevenir	Estudar os recursos consumidos pelos algoritmos e a capacidade do sistema embarcado	Equipe de Software e Eletrônica
RN26	Prevenir	Obter o feedback do time de motores a cada passo que o time de software progredir durante o desenvolvimento e verificar a viabilidade deste com o hardware escolhido pela frente de motores.	Equipe de Software e Motores

Tabela 27 – Resposta aos Riscos da Equipe de Software

Número	Ação	Resposta	Responsável
RN27	Prevenir	Criar rituais de comunicação entre o time, como, por exemplo, reuniões diárias rápidas, para verificar o progresso do grupo e poder mitigar riscos do projeto não ser finalizado no tempo hábil.	Equipe de Software
RN28	Prevenir	Dividir o trabalho da equipe igualmente com os membros do time	Equipe de Software
RN29	Prevenir	Elicitar o máximo de requisitos para o escopo atual e criar um cronograma inicial com as tarefas para o time de desenvolvimento com as atividades a serem desenvolvidas durante o semestre	Equipe de Software

Tabela 28 – Continuação da Resposta aos Riscos da Equipe de Software

A.4 Tempo

A.4.1 Estrutura Analítica do Projeto

A Estrutura Analítica do Projeto foi baseada nos pontos de controle propostos da disciplina, onde temos o planejamento, execução e a finalização, correspondendo aos pacotes de entrega dos três pontos de controle.

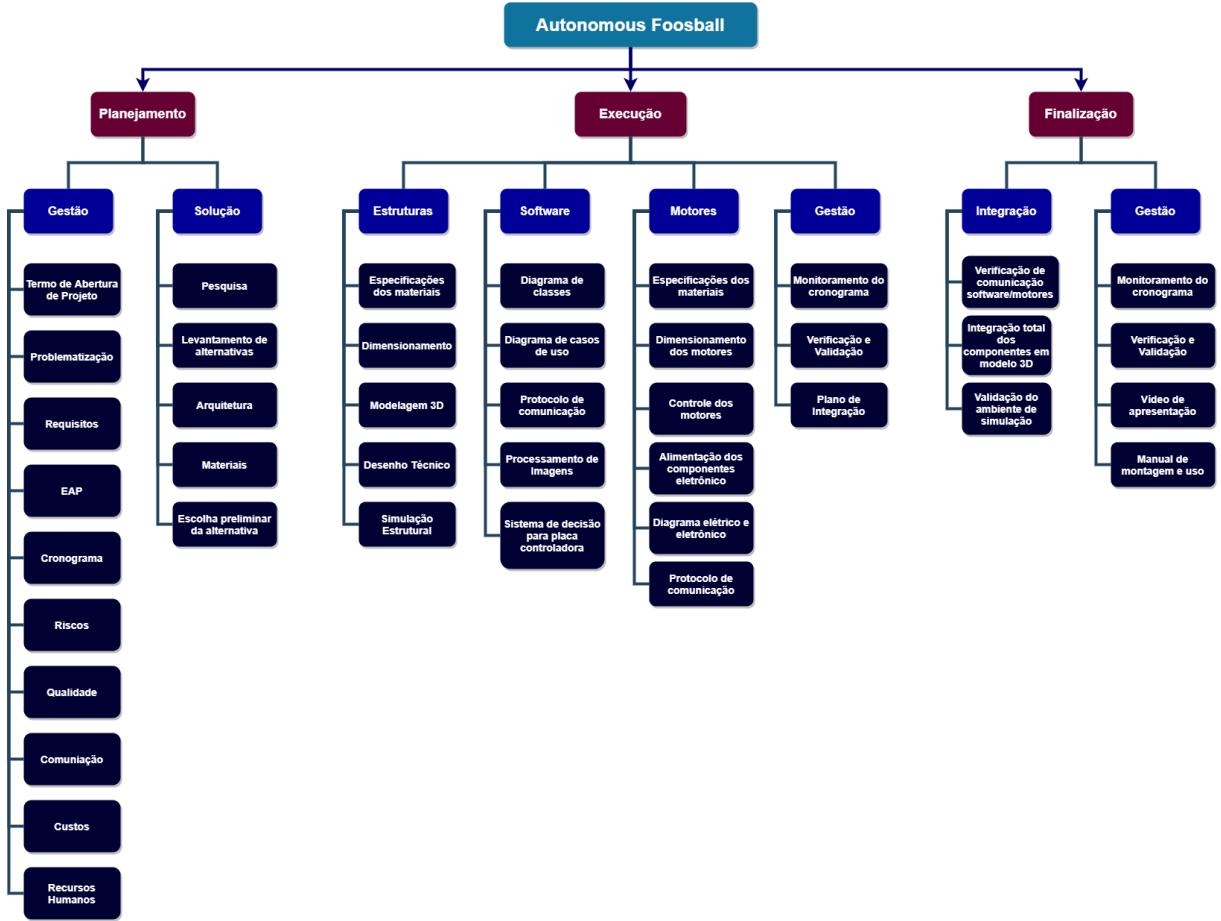


Figura 68 – EAP - Estrutura Analítica do Projeto Autonomous Foosball

A.4.2 Cronograma

O cronograma foi escrito utilizando a EAP como entrada, e nele estarão as datas das atividades previstas com seu início e fim.

A partir da EAP é possível saber o trabalho necessário para implementar e integrar os produtos necessários para a finalização do sistema, além do trabalho de gerência. Sendo assim, ao estabelecer prazos para a realização desses trabalhos, tendo em vista o prazo total para a entrega do projeto, é possível fazer um controle do progresso do projeto, comparando o planejado com o executado.

Para o desenvolvimento do cronograma, foi utilizada a ferramenta do Gantt Projectt. O cronograma pode ser visto na forma de gráfico e detalhado em atividades a seguir:

Autonomous foosball project

12/09/2020

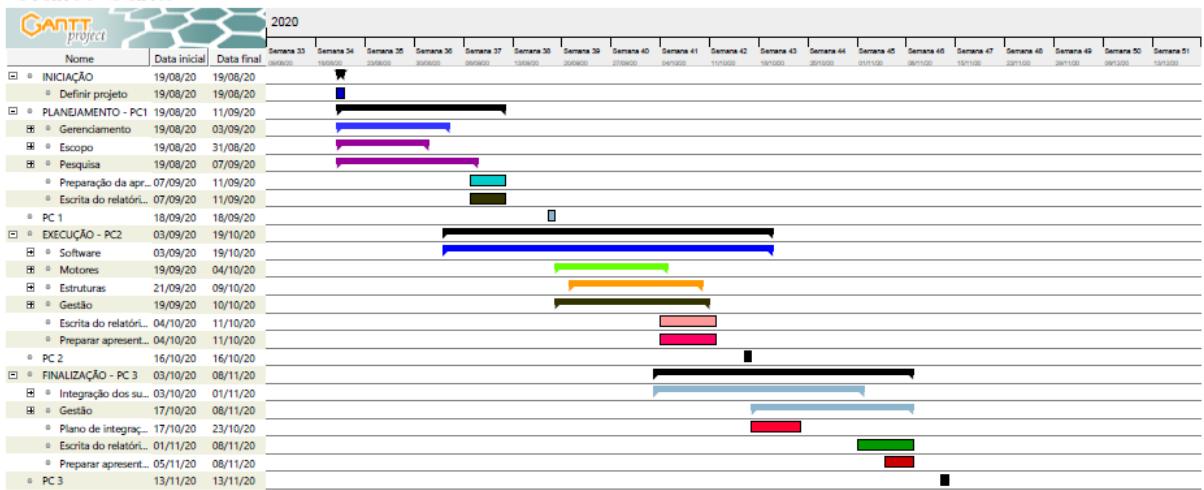
Gráfico Gantt

Figura 69 – Gráfico Gantt

Autonomous foosball project

12/09/2020

Tarefa

Nome	Data inicial	Data final
INICIAÇÃO	19/08/20	19/08/20
Definir projeto	19/08/20	19/08/20
PLANEJAMENTO - PC1	19/08/20	11/09/20
Gerenciamento	19/08/20	03/09/20
EAP	19/08/20	03/09/20
TAP	19/08/20	03/09/20
Gestão de comunicação	19/08/20	24/08/20
Requisitos	19/08/20	24/08/20
Gestão de riscos	19/08/20	24/08/20
Escopo	19/08/20	31/08/20
Problematização	19/08/20	19/08/20
Lista É/Não É	19/08/20	31/08/20
Pesquisa	19/08/20	07/09/20
Levantamento de alternativas	19/08/20	24/08/20
Avaliação das alternativas encontradas	26/08/20	05/09/20
Escolha de materiais	02/09/20	07/09/20
Dimensionamento dos componentes	19/08/20	03/09/20
Preparação da apresentação do PC 1	07/09/20	11/09/20
Escrita do relatório do PC 1	07/09/20	11/09/20
PC 1	18/09/20	18/09/20
EXECUÇÃO - PC2	03/09/20	19/10/20
Software	03/09/20	19/10/20
Elaborar diagrama de casos de uso	07/09/20	11/09/20
Elaborar diagramas de classe	11/09/20	28/09/20
Elaborar protocolo de comunicação	07/09/20	19/10/20
Desenvolver sistema de decisão para a placa controladora	03/09/20	10/10/20

Figura 70 – Cronograma Parte 1

Autonomous foosball project		12/09/2020
Tarefa		3
Nome		
Desenvolver sistema de processamento de imagem	19/09/20	03/10/20
Desenvolver ambiente de simulação	07/09/20	10/10/20
Motores	19/09/20	04/10/20
Dimensionamento dos motores	19/09/20	25/09/20
Controle dos motores	26/09/20	02/10/20
Protocolo de comunicação	27/09/20	03/10/20
Alimentação dos componentes eletrônicos	27/09/20	03/10/20
Precificação dos componentes	03/10/20	04/10/20
Diagrama elétrico/eletônico	27/09/20	03/10/20
Estruturas	21/09/20	09/10/20
Dimensionamento	21/09/20	25/09/20
Desenhos técnicos	25/09/20	29/09/20
Modelagem 3D	25/09/20	29/09/20
Simulação estrutural	01/10/20	05/10/20
Cálculos estruturais	01/10/20	05/10/20
Escolha dos materiais	08/10/20	09/10/20
Precificação dos componentes	09/10/20	09/10/20
Gestão	19/09/20	10/10/20
Monitoramento e controle	19/09/20	10/10/20
Escrita do relatório do PC2	04/10/20	11/10/20
Preparar apresentação do PC 2	04/10/20	11/10/20
PC 2	16/10/20	16/10/20
FINALIZAÇÃO - PC 3		
Integração dos subsistemas	03/10/20	08/11/20
Integração dos componentes em modelo 3D	17/10/20	01/11/20
Validação do ambiente de simulação	03/10/20	30/10/20
Embarcar sistema na Raspberry	26/10/20	30/10/20
		01/11/20

Figura 71 – Cronograma Parte 2

Autonomous foosball project		12/09/2020
Tarefa		4
Nome		
Gestão		
Monitoramento do cronograma	17/10/20	08/11/20
Verificação e validação	17/10/20	08/11/20
Manual de montagem e uso	17/10/20	08/11/20
Vídeo de apresentação	01/11/20	08/11/20
Plano de integração	17/10/20	23/10/20
Escrita do relatório do PC 3	01/11/20	08/11/20
Preparar apresentação do PC 3	05/11/20	08/11/20
PC 3	13/11/20	13/11/20

Figura 72 – Cronograma Parte 3

A.5 Requisitos

A.6 Requisitos Gerais

A.7 Requisitos de motores

Número	Requisitos em Eletrônica e Energia
1	O sistema deverá realizar a movimentação individual da haste suporte para os jogadores, sendo assim responsável pela movimentação linear dos jogadores. Serão no total quatro hastas que o sistema ficará responsável por movimentar.
2	O sistema será responsável por realizar a movimentação sobre o eixo da haste suporte para os jogadores, sendo responsável pelo chute (força pelo torque, devido ao movimento rotacional) dos jogadores. Serão no total quatro hastas que o sistema ficará responsável por movimentar.
3	O protótipo não será autônomo energeticamente, terá capacidade de estar ligado á uma fonte de tensão bivolt (110V/220V), sendo esta sua única fonte energética.
4	O sistema deverá fornecer imagens da mesa, de modo que seja feito um mapeamento da mesma, para que em seguida tais dados sejam processados.
5	O sistema deve fazer a comunicação entre componentes eletrônicos.
6	O sistema fornecerá um display conectado ao sistema, onde as configurações do jogo possam ser acessadas.

Tabela 29 – Tabela de Requisitos da equipe de motores.

A.8 Requisitos de Estrutura

Número	Requisitos estruturais
1	A estrutura deve atender aos esforços requisitados sem que haja falha estrutural
2	A estrutura deve manter o nível de vibração do sistema no mínimo
3	O sistema de motores deve ter um tempo de desmontagem de até 1h, permitindo o jogo contra um oponente humano
4	A estrutura deve permitir uma movimentação linear e rotacional dos eixos da mesa
5	A estrutura deve permitir o movimento independente dos eixos da mesa
6	As alterações feitas na mesa necessárias para acomodar a estrutura do sistema não devem atrapalhar a experiência do usuário
7	A superfície de jogo deve seguir as dimensões oficiais estabelecidas pela ITSF
8	A lente da câmera deve ter uma distância mínima de 15,58 cm da superfície do campo
9	O tamanho total do sistema deve considerar limitações de espaço, tanto para o transporte do sistema quanto para espaço necessário para seu uso
10	Uma vez montada e pronta para o jogo, a estrutura deve funcionar sem necessidade de interferência do usuário

Tabela 30 – Tabela de Requisitos Estruturais

A.9 Requisitos de Software

Para detalhar a solução de software, foi utilizada a divisão entre temáticas, funcionalidades e tarefas que precisam ser executadas. Neste sentido, a solução foi dividida em três temáticas principais: Servidor de Decisão, Controlador de Máquina e Elaboração do *Pebolim Unified Data Model* (PUDM). A definição detalhada de cada temáticas será fornecidos na Seção 2.5.

As tarefas foram elicitadas e organizadas em um Backlog de Produto e priorizadas por meio da técnica de priorização de requisitos MOSCOW onde cada tarefa possui uma *feature* relacionada ao seu escopo.

A.9.1 Técnica de Elicitação

As *features* foram levantadas com a aplicação da técnica de *brainstorming* com a equipe de software e com o *feedback* das outras áreas.

A.9.2 Backlog do Produto

A.9.2.1 Temáticas

ID	Descrição
TM01	Elaboração do PUDM
TM02	Controlador de Máquina
TM03	Servidor de Decisão

Tabela 31 – Temáticas de Software

A.9.2.2 Features

ID	Descrição	Temática
FT01	PUDM - Documentação	TM01
FT02	Controlador de Máquina - Simulação	TM02
FT03	Servidor de Decisão - Comunicação	TM03
FT04	Servidor de Decisão - Interpretação	TM03
FT05	Servidor de Decisão - Decisão	TM03
FT06	Servidor de Decisão - Interface	TM03
FT07	Servidor de Decisão - Documentação	TM03

Tabela 32 – Features de Software

A.9.2.3 Tarefas

ID	Feature	Descrição	Prioridade
T01	FT01	Definir um padrão de comunicação para os sistemas	Must
T02	FT01	Estabelecer requisitos mínimos para a compatibilidade de um componente	Must
T03	FT02	Implementar um cliente PUDM	Must
T04	FT02	Criar um campo de pebolim virtual	Must
T05	FT02	Atualizar o campo de pebolim para o modelo 3D feito por estrutura	Should
T06	FT02	Adequar dimensões do campo virtual de pebolim	Must
T07	FT02	Simular o comportamento de motores na mesa	Must
T08	FT02	Executar ações recebidas via cliente PUDM	Must
T09	FT02	Implementar uma câmera virtual	Must
T10	FT02	Enviar o estado da mesa ao servidor PUDM	Must
T11	FT02	Fazer a simulação realizar os cálculos necessários para criar os dados do PUDM	Must
T12	FT02	Possibilitar o controle a partir de 2 AI ao mesmo tempo	Could
T13	FT02	Controlar o movimento da bola de pebolim a partir do teclado	Must
T14	FT02	Controlar o movimento das fileiras de jogadores	Must
T15	FT02	Ajustar as configurações de física da engine de jogo	Must
T16	FT03	Implementar lado servidor do PUDM	Must
T17	FT04	Converter imagem codificada recebida via PUDM em imagem real	Must
T18	FT04	Identificar posição da bola na imagem	Must
T19	FT04	Identificar posição dos jogadores inimigos na imagem	Should
T20	FT04	Identificar posição dos jogadores aliados na imagem	Could
T21	FT05	Interpretar todos os dados gerados pelo cliente PUDM	Must
T22	FT03	Enviar decisão do servidor para o cliente PUDM	Must
T23	FT06	Ver informações atuais sobre a partida que estou jogando	Could
T24	FT06	Ver dados estatísticos do meu histórico de partidas	Could

Tabela 33 – Histórias de Usuário

ID	Feature	Descrição	Prioridade
T25	FT06	Metodo para calibrar a mesa	Must
T26	FT06	Controlar o estado e as configurações da partida	Should
T27	FT06	Controle de dificuldade	Could
T28	FT07	Documentar empacotamento do Decision Server	Must

Tabela 34 – Histórias de Usuário

A.10 Custo

Os custos considerados para o projeto foram divididos em custos dos componentes estruturais/mecânicos e custos dos componentes eletrônicos/alimentação. Os levantamentos feitos são mostrados nas tabelas 35 e 36. A soma desses dois levantamentos resulta no custo total do projeto que é de R\$ 10316,02.

Produto	Preço Unitário (R\$)	Qntd	Preço Total (R\$)	Fornecedor
NEMA 23 - Short (IP20) - 24Vcc - 4,5A	229,00	4	916,00	Robocore
NEMA 34 - Long (IP20) - 24Vcc - 4,5A	469,00	4	1876,00	Mercado Livre
Driver Para Motor de Passo 5A WD-TB6600 - Wotiom	79,90	8	639,20	Baú da Eletrônica
Tela Lcd Hdmi 10.1 1024x800 Ips Touch Screen Raspberry Pi	999,00	1	999,00	Mercado livre
Câmera raspberry pi de 5mp	199,90	1	199,90	Filipeflop
Microprocessador raspberry pi 4 modelo B	439,90	1	439,90	Filipeflop
Microcontrolador Arduino mega 2560	109,90	1	109,90	Eletrogate
Módulo IR Detector de Linha	8,90	4	35,60	Filipeflop
Chave Fim de Curso com Roloete 5A 250V	2,90	8	23,20	Filipeflop
Fonte Meanwell 1000-2 Modelo: SE-1000-24	2020,80	1	2020,80	Americanas
Fonte Oficial para Raspberry Pi	99,90	1	99,90	Robocore
Total			7359,40	

Tabela 35 – Custos com componentes eletrônicos/alimentação

Produto	Preço Unitário (R\$)	Qntd	Preço Total (R\$)	Fornecedor
Manopla Punho Pebolim Totó Fla Flu Ferro 9/16 E 5/8. Kit com 8 unidades	24,99	1	24,99	MINEIROS DIVERSÕES
Bonecos Pebolim Ferro Passante 9/16. Kit com 6 unidades	20,99	4	84,00	BILBRAG
Suporte Metálico para Motor NEMA 34	36,60	4	148,00	TUDO BRINDES
Barra Redonda 16mm aço 1020 6 metros	56,50	2	113,00	SOLUTOS
Chapa Mdf 275 X 185 X 15 mm	250,00	3	750,00	BEIJAFLOR MADEIRAS LTDA
Campo Tampo Pebolim Totó Fla Flu Pacau 116 X77 X0,3 (4pçs) Sapata, Pé Nivelador C/ Bucha 13mm Rosca 3/8 X 25mm	69,99	1	69,99	MINEIROS DIVERSÕES
Cremalheira Reta módulo 2 - barra 2 metros	259,00	1	259,00	TECMASF
Acoplamento Flexivel Cnc Motor Passo Eixo 12 X 12mm Fuso	20,90	4	83,60	SERGIO VIOLANTE
bloco ferro fundido sob medida	50,00	4	200,00	METALÚRGICA INDIANÁPOLIS
treliças de alumínio	100,00	4	400,00	DULONG
trilho telescópico para gaveta	18,00	4	72,00	MARK FERRANGENS
Suporte Metálico para Motor NEMA 23	40,00	4	160,00	LGV
egrenagem 15 dentes modulo 1,5	25,00	4	100,00	TECMASF
cobertura fibra de vidro	250,00	1	250,00	SUPPORT
Chapa compensado naval cru 1x1m	64,00	1	64,00	LEROY MERLIN
Placa de acrilico cortada a laser	60,00	1	60,00	PERFECTVISION
porcas, parafusos e arruelas			90,00	POPA
Total			2956,62	

Tabela 36 – Custos dos componentes estruturais/mecânicos

APÊNDICE B – Motores/eletrônica

B.1 Cálculos associados aos motores e alimentação

B.1.1 Cálculos associados aos motores de rotação

B.1.1.1 Dados Iniciais

$r = \text{raio} [m];$

$m_1 = \text{Massa Jogador} [Kg];$

$u_1 = \text{Velocidade Inicial Jogador} [m/s];$

$v_{1=} = \text{Velocidade Final Jogador} [m/s];$

$m_2 = \text{Massa Bola} [kg];$

$u_2 = \text{Velocidade Inicial Bola} [m/s];$

$v_2 = \text{velocidade final bola} [m/s];$

$\omega_i = \text{Velocidade Angular Inicial do Jogador} [m/s];$

$\Delta\theta = \text{Deslocamento Angular} [rad];$

$\omega_f^1 = \text{Velocidade Angular Final do Jogador} [rad/s];$

$\alpha = \text{Aceleração Angular} [rad/s^2];$

$I = \text{Momento de Inércia}^2;$

B.1.1.1.1 Equações Utilizadas

- Colisão perfeitamente elástica:

$$v_2 = \frac{2m_1}{m_1 + m_2}u_1 + \frac{m_2 - m_1}{m_1 + m_2}u_2 \quad (\text{B.1})$$

- Torricelli (MCUV)³:

$$\omega^2 = \omega^2 + 2\alpha\Delta\theta \quad (\text{B.2})$$

¹ determinado pelo datasheet([SERVOTRONIX, 2014](#))

² Dado de projeto

³ Movimento Circular Uniformemente Variado

- Velocidade Angular:

$$\omega = V.r \quad (\text{B.3})$$

V = Componente de Velocidade Escalar

- Torque (MCUV):

$$\tau = I.\alpha \quad (\text{B.4})$$

B.1.1.1.2 Determinando a velocidade final da bola

$$\omega_f = 350 \text{ rpm}$$

$$\omega_f = 350 \text{ rpm} = 36,6 \text{ rps}$$

$$u_1 = 2,56 \text{ m/s}$$

$$m_1 = 0,36 \text{ kg}$$

$$v_1 \leq 2,56 \text{ m/s}$$

$$u_2 \geq 0$$

$$m_2 = 0,04 \text{ kg}$$

$$v_2 \geq 5 \text{ m/s}$$

A velocidade final da bola é aceitável para o projeto.

B.1.1.1.3 Determinando o torque

$$\Delta\theta = 40\ddot{\theta} = 0,698 \text{ rad}$$

$$\omega_i = 0 \text{ rps}$$

$$\omega_f = 36,6 \text{ rps}$$

$$\alpha = 962,1 \text{ rad/s}^2$$

$$I = 0,002 \quad \tau = 1,92 \text{ Nm}$$

Portanto, o motor NEMA 34 - Long (IP20) - 24Vcc - 4,5A, é o ideal para o projeto no quesito rotação.

B.1.1.2 Cálculos associados aos motores de Deslocamento

B.1.1.2.1 Dados Iniciais

$ac = \text{Ângulo de Carga} = 0^{\text{4}}$;

$Dg = \text{Diametro Primitivo do Pinhão} = 0,068\text{m}$;

$m = \text{Massa Conjunto} = 8,6\text{kg}$;

$ec = \text{Eficiência Cremalheira} = 0,9$;

$rd = \text{Redução} = 1^{\text{5}}$;

$fs = \text{Fator de Segurança} = 2$;

$\Delta S = \text{Maior Deslocamento Possível Haste} = 0,32\text{m}^{\text{6}}$.

$t=1\text{s}^{\text{7}}$

B.1.1.2.2 Equações Utilizadas

- Equação horária das posições

$$S = S_0 + V_0 t + \frac{at^2}{2} \quad (\text{B.5})$$

- Segunda Lei de Newton

$$F = ma \quad (\text{B.6})$$

- Força atrito na cremalheira

$$Fat = m \cdot \text{Cos}(ac) \cdot 0,3924 \quad (\text{B.7})$$

- Torque

$$\tau = \frac{fat}{ec} \frac{Dg}{2} \quad (\text{B.8})$$

⁴ Não há carga

⁵ Não há redução

⁶ Considera a haste com o maior deslocamento possível (Goleiro)

⁷ tempo limite

- Torque de Aceleração/Desaceleração

$$\tau a = \frac{Fat + Fa}{ec} \frac{Dg}{2} \quad (B.9)$$

- Torque Exigido do Motor

$$\tau f = \frac{\tau + \tau a}{rd.er} \cdot fs \quad (B.10)$$

B.1.1.2.3 Determinando Aceleração

$$\Delta S = \frac{a \cdot t^2}{2}$$

$$a = 0,64 \text{m/s}^2$$

B.1.1.2.4 Determinando Forças

$$Fat = m \cdot 1.0,3924 = 3,37N$$

$$Fa = 5,504 \text{ N}$$

B.1.1.2.5 Determinando Torque

$$\tau = 0,127Nm$$

$$\tau a = 0,335Nm$$

$$\tau f = 0,92$$

Logo, o motor NEMA 23 - Short (IP20) - 24Vcc - 4,5 A é o ideal para o projeto no questão Deslocamento.

B.1.1.3 Cálculos associados à Movimentação linear do NEMA 23

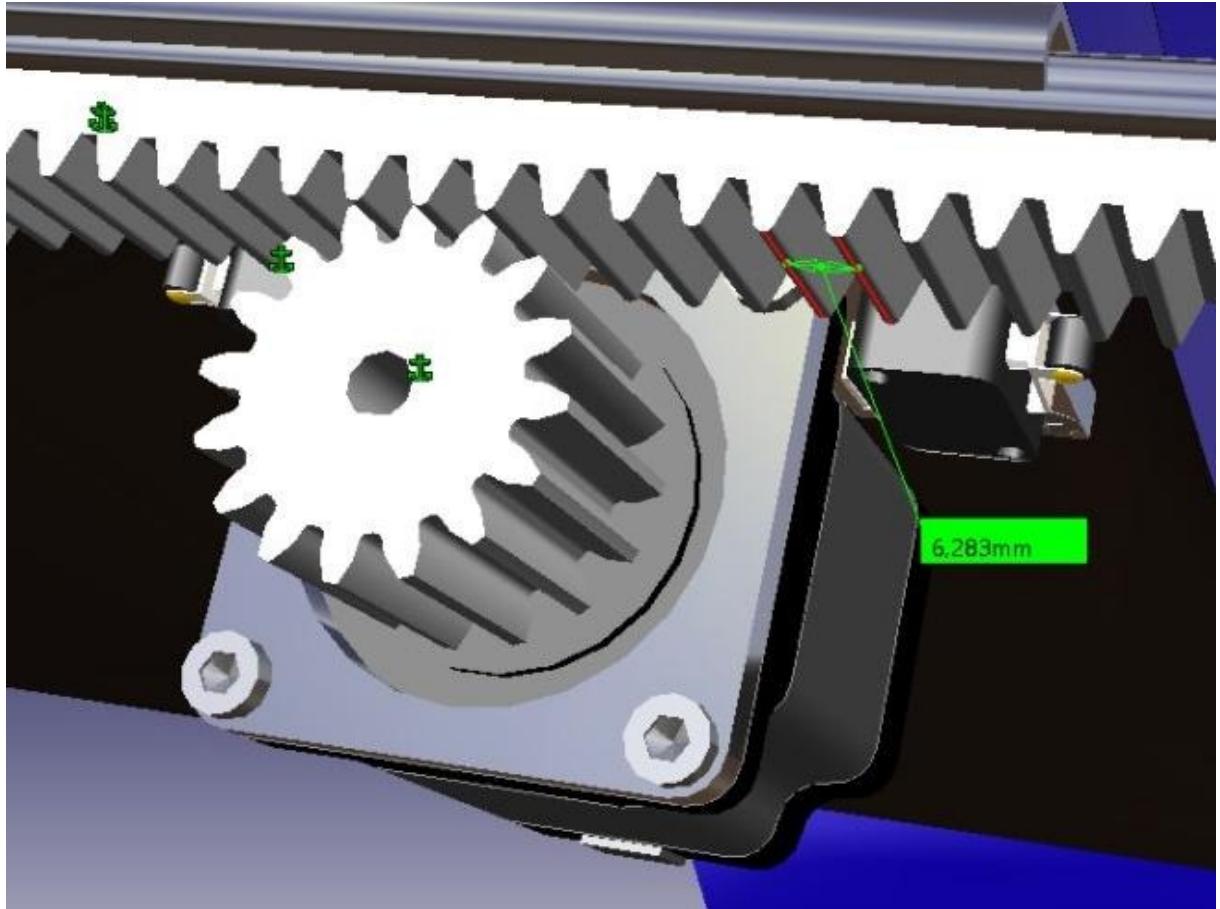


Figura 73 – Características do motor no sistema

Distância entre um dente e outro = Pitch = 6,28 mm

Número de dentes da polia

$$N = 1$$

1 rotação do motor

$$= rot = 200$$

passos

Resolução do motor(RES)

$$RES = rot / (pitch * N) = 0,53mm \quad (B.11)$$

B.1.1.4 Cálculos associados à alimentação

Sendo 8 motores de passo, são necessários 8 drivers. O driver pode operar em diferentes níveis de tensão e corrente, sendo o valor máximo de operação do driver 5A.

Quantidade de motores = N = 8; Corrente de operação máxima do driver = i = 5A; e Tensão de operação dos motores = v = 24V.

$$PotênciaTotal = N * i * v = 960W \quad (\text{B.12})$$

B.2 Características dos dispositivos eletrônicos e energéticos do projeto

B.2.1 Elementos em eletrônica

B.2.1.1 Microcontrolador principal

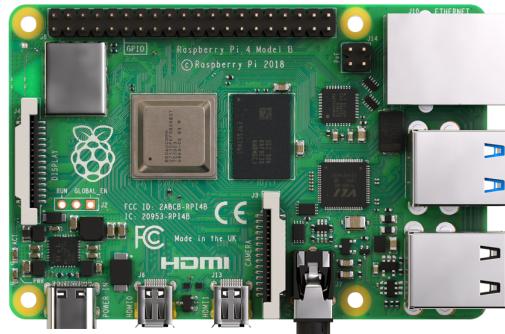


Figura 74 – Microprocessador Raspberry Pi 4 Model B Anatel.(FILIPE FLOP, 2020b)

Especificações do Processador:

- Chip: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC
- Memória: 2GB ou 4GB DDR4
- Armazenamento: Slot para cartão microSD
- GPU : VideoCore VI

Interface:

- 2 portas micro HDMI com suporte a vídeos 4K e 60fps
- 2 portas USB 3.0 e 2 portas USB 2.0
- GPIO de 40 pinos
- Interface para display (DSI)
- Interface para câmera (CSI)

Compatibilidade de Softwares:

- sistema operacional Raspbian oficial
- Ubuntu Mate
- Snappy Ubuntu Core
- centros de mídia baseados em Kodi OSMC e LibreElec
- Risc não baseado em Linux
- Windows 10 IoT Core

Especificações Gerais:

- Peso: 50 g
- Temperatura de operação: 0–50°C
- Dimensões: 85 x 56 x 17 mm
- 5 V DC via conector USB-C (mínimo 3 A *)
- 5 V DC via cabeçalho GPIO (mínimo 3A *)
- Preço: R\$ 439,90 até R\$ 619,90([FILIPE FLOP, 2020b](#))

B.2.1.2 Microcontrolador secundário

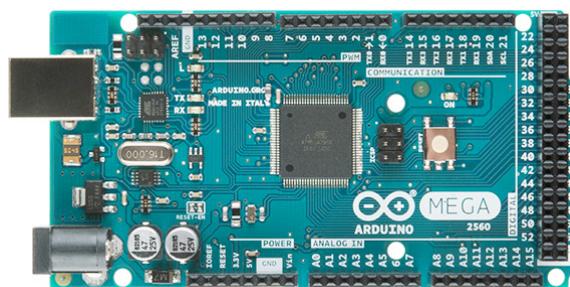


Figura 75 – Microcontrolador Mega 2560 R3

Especificações gerais

- Modelo: Arduino Mega 2560
- Microcontrolador: ATmega2560
- Conversor USB/Serial: ATmega16U2;

- Velocidade do Clock: 16 MHz;
- Temperatura de trabalho: -40 °à +85 °C;
- Dimensões: 101,6mm x 53,4mm
- Peso: 150 g;

Memória

- EEPROM: 4KB
- Memória SRAM: 8KB
- Memória Flash: 256KB (8KB usado no bootloader)

Alimentação

- Tensão de Alimentação: 7 à 12 Vdc (Conector Jack e pino Vin);
- Tensão de Operação: 5 Vdc;
- Tensão de Nível Lógico: 5,0 Vdc (Tolera 3,3 Vdc);

Interfaces

- Portas Analógicas: 16
- Portas Digitais: 54 (15 podem ser usadas como PWM)
- SPI, I2C

B.2.1.3 Câmera



Figura 76 – Câmera Compatível Raspberry Pi 5MP.([FILIPE FLOP, 2020a](#))

Especificações

- Sensor OV5647
- Resolução: 5MP
- Vídeo: 1080p30F, 960p45, 720p60, VGA(640x480)90 e QVGA(320x240)120
- CCD Size: 1/4 polegadas
- Abertura (F): 1.8
- Comprimento focal: 3.6mm (ajustável)
- Ângulo de visão (Diagonal): 75.7 graus
- Dimensões: 25 x 24 x 23,5 mm
- Cabo flat: 15cm
- Preço: R\$199,90([FILIPE FLOP, 2020a](#))

B.2.1.4 Display



Figura 77 – Tela Lcd Hdmi 10.1 1024x800 Ips Touch Screen Raspberry Pi.([MERCADO LIVRE, 2020](#))

Especificações:

- Tamanho do display LCD: 10.1 polegada IPS
- Estrutura do módulo: Display HDMI + toque usb + pcb
- Resolução da tela: 1280rgbx800 pixel
- Dimensões do módulo: 229.46x149.10x4.9mm
- Dimensões de área ativa: 216.96x135.60mm
- Passo do pixel: 0.1695x0.1695mm

- Arranjo do pixel da cor: lista do rgb
- Modo de exibição: tipo transmissivo, modo normalmente preto
- Direção de visualização (inversão cinza): livre (85/85/85)
- Tipo da relação do lcd: hdmi
- Tipo de interface ctp: usb
- Números de cores: 16.7 m
- Tipo de estrutura ctp: vidro de cobertura + vidro ito (g + g)
- Brightnees: 200 (typ) cd/m²
- Relação de contraste: 500: 1
- Temperatura de funcionamento:-20°C a 70°C
- Temperatura de armazenamento:-30°C a 80°C
- Preço: R\$999,00([MERCADO LIVRE, 2020](#))

B.2.1.5 Driver



Figura 78 – Driver Para Motor de Passo 5A WD-TB6600 - Wotiom. ([BAÚ DA ELETRO-NICA, 2020](#))

Especificações Técnicas:

- Tensão de Alimentação: 12 a 48VDC
- Configurações de saída de corrente
- Corrente de Saída: de 1A a 5A dependendo da seleção dos switches
- Corrente de saída do motor de passo de 0.2A a 5A

- Temperatura de operação: -10°C a 45°C
- Temperatura de armazenamento: -40°C a 70°C
- Peso: 230g
- Preço: R\$79,90([BAÚ DA ELETRÔNICA, 2020](#))

Nota:

- Para Sinais de 5V, não é necessário utilizar um resistor (R) nas entradas de sinais;
- Para Sinais de 12V, deverá ser utilizado um resistor (R) de 1k (1/4W);
- Para Sinais de 24V, deverá ser utilizado um resistor (R) de 2k (1/4W);
- Os resistores (R) devem ser conectados aos terminais de saída de sinais do controlador.

B.2.2 Alimentação do projeto

B.2.2.1 Alimentação dos Motores e Drivers



Figura 79 – Fonte Meanwell 1000-24([AMERICANAS, 2020](#))

Especificações Técnicas:

- Modelo: SE-1000-24;
- Tensão de Alimentação: entre 90 VAC e 132 VAC / entre 180 VAC e 230 VAC / entre 254 VDC e 370 VDC;
- Tensão de saída: 24VDC;

- Variação tensão de saída: entre 22VDC e 27VDC;
- Corrente de operação: entre 0 e 41.7A;
- Potência: 1000,8W;
- Eficiência: 88%;
- Temperatura de operação: entre -20°C e +60°C;
- Proteção sobrecarga: entre 105% e 125% da potência permitida;
- Proteção sobretensão: entre 28V e 32.4V;
- proteção de superaquecimento: 70°C +- 5°C;
- Tipo de proteção: desligamento automático até a temperatura abaixar;
- Peso: 2,5kg
- R\$ 2.020,80([AMERICANAS, 2020](#))

Recursos:

- Limitador de onda de corrente alternada;
- Interruptor de seleção para tensão de entrada;
- Proteção: Curto Circuito / Sobrecarga / Sobretensão / Superaquecimento;
- Sistema de resfriamento interno por ventuínha (*DC ball bearing fan*);
- Saída *DC_OK signal output*;
- Botão de Liga-Desliga;
- aprovação UL / CUL;e
- 2 anos de garantia;

B.2.2.2 Alimentação Microprocessador

Especificações Técnicas:

- Modelo: Raspberry pi
- Tensão de Alimentação: 96 a 264 VAC;
- Tensão de saída: 5,1VDC; 3,0A



Figura 80 – Fonte RaspBerry Oficial ([ROBOCORE, 2020](#))

- Cabo de 1,5m 18AWG com conector USB-C;
- R\$ 99,90([ROBOCORE, 2020](#))

Recursos:

- Proteção: Curto Circuito / Sobrecarga / Sobretensão / Superaquecimento;

B.2.3 Motores do projeto

B.3 Código Fonte da Solução de Eletrônica

O código fonte para a solução de eletrônica é organizado por meio do software de controle de versão Git. Os repositórios Git do projeto são disponibilizado de forma aberta na organização no GitHub⁸ criada para este projeto.

Nesta organização os repositórios utilizados são:

- [SESC](#): Contém a documentação relacionada ao SESC;
- Habilitar os GPIO do Arduino direto da Raspberry

```
from pyfirmata import ArduinoMega, util      #Biblioteca pyfirmata

mega = ArduinoMega('COM3')                      #pyfirmata Arduino Mega 2560
iterator = util.Iterator(mega)
iterator.start()

DM_pin_YGOL_ENE = mega.get_pin('d:18:o') # Habilitar Motor do eixo y
```

⁸ <https://github.com/pi2-2020-1-pebolim>

height	Unit	Value	Value
Model	-	IP20	IP20
NEMA	-	23 Short	34L
Input Power, Nominal ($\pm 10\%$)	VDC	14–48	14–48
Auxiliary Input Power, Nominal ($\pm 10\%$)	VDC	6–24	6–24
Auxiliary Input Power, Maximum	W	1	1
Detent Torque	mNm	40	350
Thrust Load Limit	kg	0.6	3.8
Overhung Load Limit (from shaft end)	N	50	260
Rotor Inertia	$g \cdot cm^2$	260	2750
Holding torque at continuous current	Nm	1.1	5.5
Holding torque at peak current	Nm	1.3	7
Continuous Output Current	A	4.5	7
Peak Output Current (application dependent)	A	6.5	11.5
Step Angle	deg	1.8	1.8
Magnetic Encoder, Resolution	ppr	4096	4096
Circuit Loss	W	6	6
Weight	kg	0.80	4.30
Connection Hardware Screw Size/Torque	Nm	3	5.2
Under-Voltage Trip, Nominal	VDC	Logic	Logic
Over-Voltage Trip	VDC	Logic	Logic

Tabela 37 – Motores NEMA 23s e 34s ([SERVOTRONIX, 2014](#))

```

DM_pin_YGOL_DIR = mega.get_pin('d:19:o') # Direção do Motor do eixo y
DM_pin_YGOL_PAS = mega.get_pin('d:20:o') # Passo do Motor do eixo y
DM_pin_RGOL_ENE = mega.get_pin('d:21:o') # Habilitar Motor de rotação
DM_pin_RGOL_DIR = mega.get_pin('d:22:o') # Direção do Motor de rotação
DM_pin_RGOL_PAS = mega.get_pin('d:23:o') # Passo do Motor de rotação
FC_pin_MGOL_MIN = mega.get_pin('d:24:i') #Lendo o Fim de Curso Mínimo

```

```

FC_pin_MGOL_MAX = mega.get_pin('d:25:i') #Lendo o Fim de Curso Maximo
FC_pin_EGOL      = mega.get_pin('d:26:i') #Lendo o Fim de Curso Eletr.

DM_pin_YDEF_ENE = mega.get_pin('d:27:o')
DM_pin_YDEF_DIR = mega.get_pin('d:28:o')
DM_pin_YDEF_PAS = mega.get_pin('d:29:o')
DM_pin_RDEF_ENE = mega.get_pin('d:30:o')
DM_pin_RDEF_DIR = mega.get_pin('d:31:o')
DM_pin_RDEF_PAS = mega.get_pin('d:32:o') #Habilitar pin digital 33 como OUT
FC_pin_MDEF_MIN = mega.get_pin('d:33:i') #Habilitar pin digital 34 como iN
FC_pin_MDEF_MAX = mega.get_pin('d:34:i')
FC_pin_EDEF      = mega.get_pin('d:35:i')

DM_pin_YMEI_ENE = mega.get_pin('d:36:o')
DM_pin_YMEI_DIR = mega.get_pin('d:37:o')
DM_pin_YMEI_PAS = mega.get_pin('d:38:o')
DM_pin_RMEI_ENE = mega.get_pin('d:39:o')
DM_pin_RMEI_DIR = mega.get_pin('d:40:o')
DM_pin_RMEI_PAS = mega.get_pin('d:41:o')
FC_pin_MMEI_MIN = mega.get_pin('d:42:i')
FC_pin_MMEI_MAX = mega.get_pin('d:43:i')
FC_pin_EMEI      = mega.get_pin('d:44:i')

DM_pin_YATA_ENE = mega.get_pin('d:45:o')
DM_pin_YATA_DIR = mega.get_pin('d:46:o')
DM_pin_YATA_PAS = mega.get_pin('d:47:o')
DM_pin_RATA_ENE = mega.get_pin('d:48:o')
DM_pin_RATA_DIR = mega.get_pin('d:49:o')
DM_pin_RATA_PAS = mega.get_pin('d:50:o')
FC_pin_MATA_MIN = mega.get_pin('d:51:i')
FC_pin_MATA_MAX = mega.get_pin('d:52:i')
FC_pin_EATA      = mega.get_pin('d:53:i')

```

- Sistema de controle e simulação

```

#Declarando as variáveis como GLOBAL
global INICIAR,MAPEAMENTO
global yPASS,aPASS,yGOL,aGOL,yDEF,aDEF,yMEI,aMEI
global yATA,aATA,DM_YGOL,DM_YDEF,DM_YMEI,DM_YATA

```

```

#Iniciando o SESC
while INICIAR == True:
    MESA()
    CONFIG()
    INIC()

#Fazendo o Mapeamento do SESC
while MAPEAMENTO == True:
    MAPE()
    PASS()

#Desativando os motores
DM_YGOL = "100"
DM_YDEF = "100"
DM_YMEI = "100"
DM_YATA = "100"

#Movimentando do SESC
MOVER(ID,yPOS,CHUT)
return yGOL,aGOL,yDEF,aDEF,yMEI,aMEI,yATA,aATA

```

- Lendo os sensores de Fim de Curso

```

#Declarando as variáveis como GLOBAL

global yGOL,aGOL,mGOL,FC_pin_MGOL_MIN,FC_pin_MGOL_MAX
global FC_pin_EGOL,FCMGOL_MIN,FCMGOL_MAX,FCEGOL
global yDEF,aDEF,mDEF,FC_pin_MDEF_MIN,FC_pin_MDEF_MAX
global FC_pin_EDEF,FCMDEF_MIN,FCMDEF_MAX,FCEDEF
global yMEI,aMEI,mMEI,FC_pin_MMEI_MIN,FC_pin_MMEI_MAX
global FC_pin_EMEI,FCMMEI_MIN,FCMMEI_MAX,FCEMEI
global yATA,aATA,mATA,FC_pin_MATA_MIN,FC_pin_MATA_MAX
global FC_pin_EATA,FCMATA_MIN,FCMATA_MAX,FCEATA
global aPASS,MESA

#Se a mesa física estiver conectada, os dados
#do fim de curso será coletado a partir dela.

```

```

if MESA == "FISICA":
    FCMGOL_MIN = int(FC_pin_MGOL_MIN.read())

```

```

FCMGOL_MAX = int(FC_pin_MGOL_MAX.read())
FCEGOL      = int(FC_pin_EGOL.read())

FCMDEF_MIN = int(FC_pin_MDEF_MIN.read())
FCMDEF_MAX = int(FC_pin_MDEF_MAX.read())
FCEDEF     = int(FC_pin_EDEF.read())

FCMMEI_MIN = int(FC_pin_MMEI_MIN.read())
FCMMEI_MAX = int(FC_pin_MMEI_MAX.read())
FCEMEI     = int(FC_pin_EMEI.read())

FCMATA_MIN = int(FC_pin_MATA_MIN.read())
FCMATA_MAX = int(FC_pin_MATA_MAX.read())
FCEATA     = int(FC_pin_EATA.read())

else:
    #Caso não exista mesa física, os dados do fim de curso
    #será coletado a partir da simulação.
    FCMGOL_MIN = _FCM(yGOL,-mGOL)
    FCMGOL_MAX = _FCM(mGOL, yGOL)
    FCEGOL     = _FCE(aGOL,aPASS)

    FCMDEF_MIN = _FCM(yDEF,-mDEF)
    FCMDEF_MAX = _FCM(mDEF, yDEF)
    FCEDEF     = _FCE(aDEF,aPASS)

    FCMMEI_MIN = _FCM(yMEI,-mMEI)
    FCMMEI_MAX = _FCM(mMEI, yMEI)
    FCEMEI     = _FCE(aMEI,aPASS)

    FCMATA_MIN = _FCM(yATA,-mATA)
    FCMATA_MAX = _FCM(mATA, yATA)
    FCEATA     = _FCE(aATA,aPASS)

```

- Movimentando os motores

#Declarando as variáveis como GLOBAL

```

global yGOL,DM_YGOL,DM_pin_YGOL_ENE,DM_pin_YGOL_DIR,DM_pin_YGOL_PAS
global aGOL,DM_RGOL,DM_pin_RGOL_ENE,DM_pin_RGOL_DIR,DM_pin_RGOL_PAS
global yDEF,DM_YDEF,DM_pin_YDEF_ENE,DM_pin_YDEF_DIR,DM_pin_YDEF_PAS

```

```

global  aDEF,DM_RDEF,DM_pin_RDEF_ENE,DM_pin_RDEF_DIR,DM_pin_RDEF_PAS
global  yMEI,DM_YMEI,DM_pin_YMEI_ENE,DM_pin_YMEI_DIR,DM_pin_YMEI_PAS
global  aMEI,DM_RMEI,DM_pin_RMEI_ENE,DM_pin_RMEI_DIR,DM_pin_RMEI_PAS
global  yATA,DM_YATA,DM_pin_YATA_ENE,DM_pin_YATA_DIR,DM_pin_YATA_PAS
global  aATA,DM_RATA,DM_pin_RATA_ENE,DM_pin_RATA_DIR,DM_pin_RATA_PAS
global  yPASS,aPASS

DM_pin_YGOL_ENE.write(int(DM_YGOL[0])) #Escrevendo na porta do Arduíno
DM_pin_YGOL_DIR.write(int(DM_YGOL[1])) #a movimentação do motor
DM_pin_YGOL_PAS.write(int(DM_YGOL[2])) #Escrevendo na Simulação
yGOL = _PASS(yGOL,DM_YGOL, yPASS)      #a movimentação do motor

DM_pin_RGOL_ENE.write(int(DM_RGOL[0]))
DM_pin_RGOL_DIR.write(int(DM_RGOL[1]))
DM_pin_RGOL_PAS.write(int(DM_RGOL[2]))
aGOL = _PASS(aGOL,DM_RGOL,aPASS)

DM_pin_YDEF_ENE.write(int(DM_YDEF[0]))
DM_pin_YDEF_DIR.write(int(DM_YDEF[1]))
DM_pin_YDEF_PAS.write(int(DM_YDEF[2]))
yDEF = _PASS(yDEF,DM_YDEF, yPASS)

DM_pin_RDEF_ENE.write(int(DM_RDEF[0]))
DM_pin_RDEF_DIR.write(int(DM_RDEF[1]))
DM_pin_RDEF_PAS.write(int(DM_RDEF[2]))
aDEF = _PASS(aDEF,DM_RDEF,aPASS)

DM_pin_YMEI_ENE.write(int(DM_YMEI[0]))
DM_pin_YMEI_DIR.write(int(DM_YMEI[1]))
DM_pin_YMEI_PAS.write(int(DM_YMEI[2]))
yMEI = _PASS(yMEI,DM_YMEI, yPASS)

DM_pin_RMEI_ENE.write(int(DM_RMEI[0]))
DM_pin_RMEI_DIR.write(int(DM_RMEI[1]))
DM_pin_RMEI_PAS.write(int(DM_RMEI[2]))
aMEI = _PASS(aMEI,DM_RMEI,aPASS)

DM_pin_YATA_ENE.write(int(DM_YATA[0]))

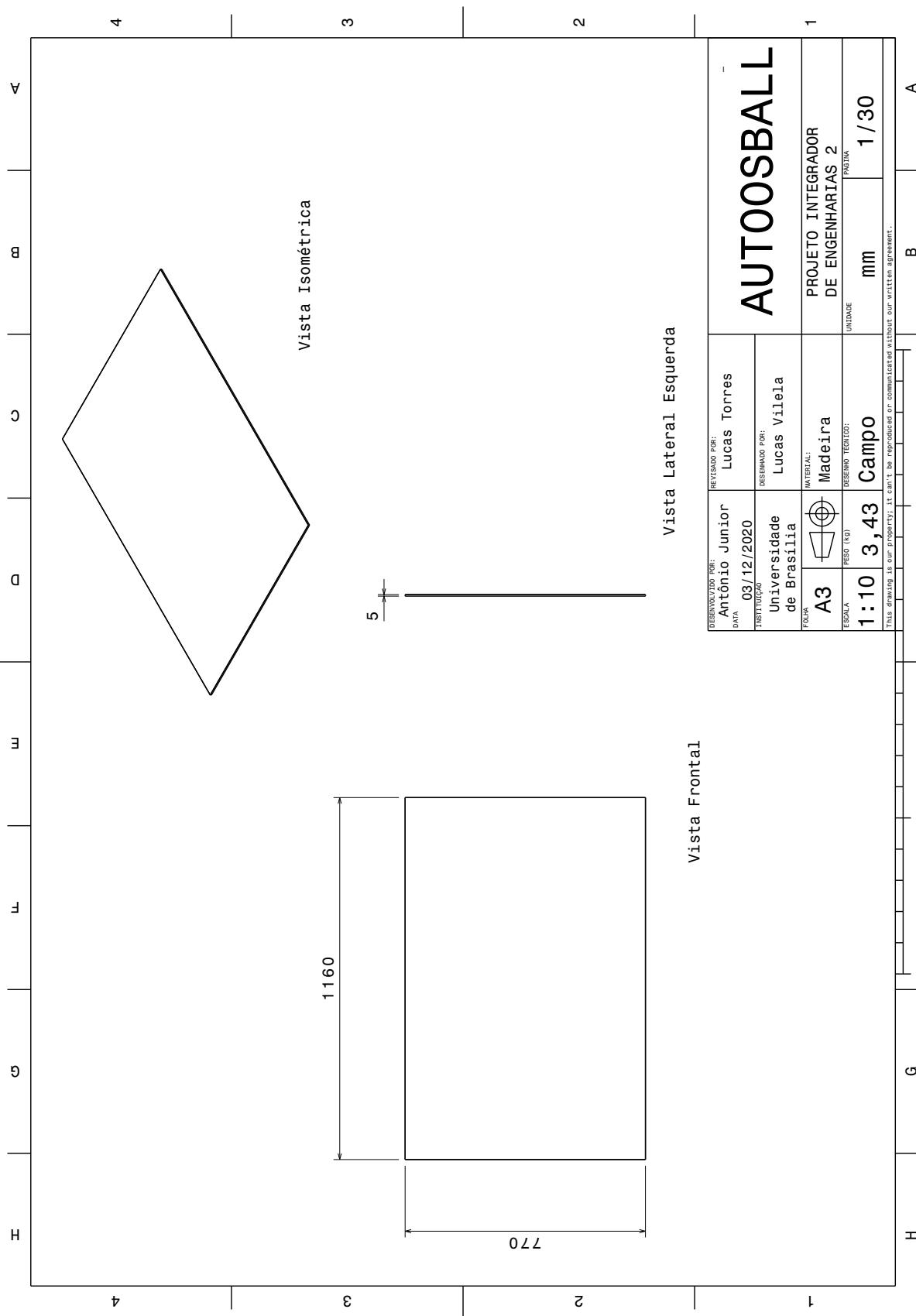
```

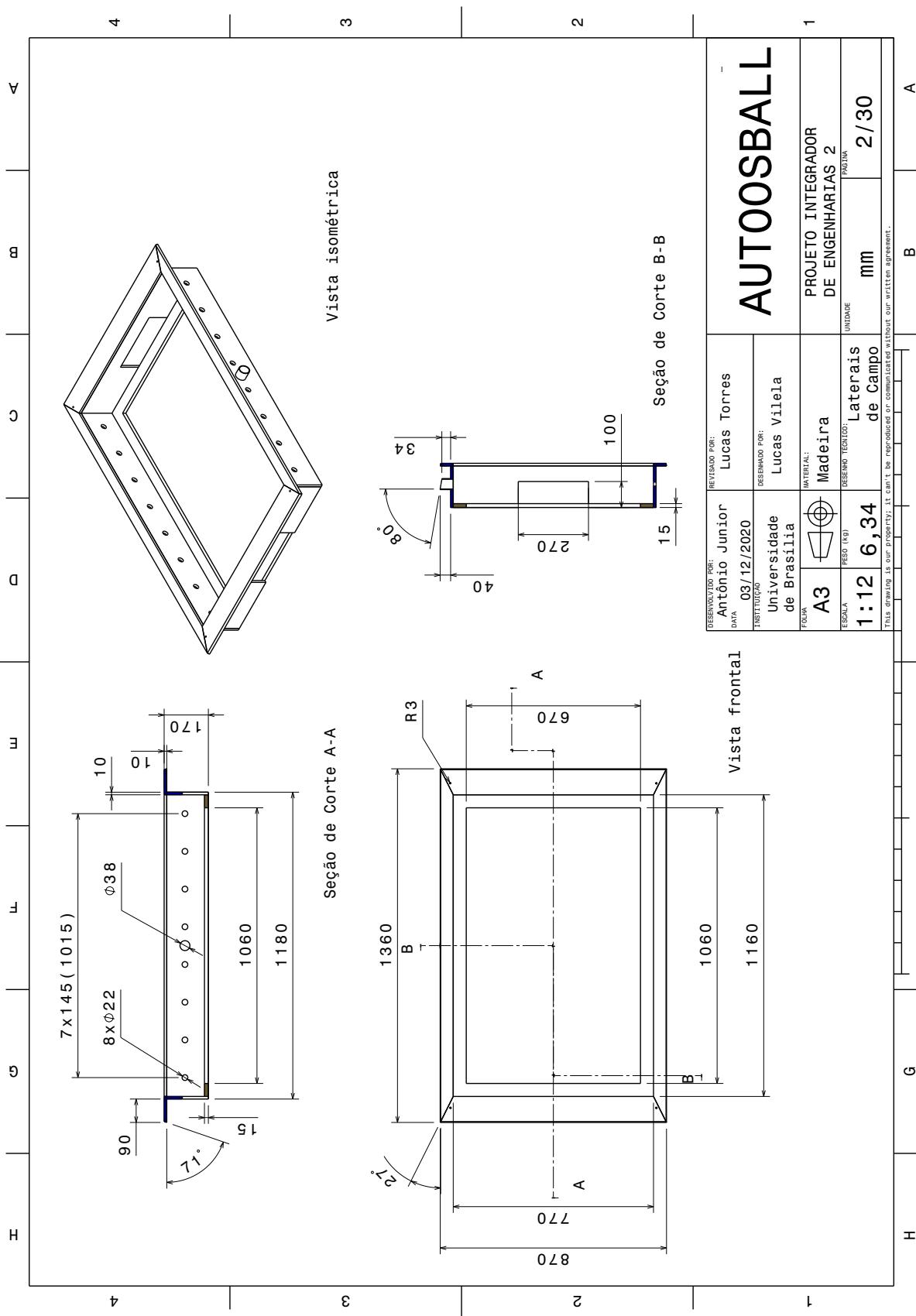
```
DM_pin_YATA_DIR.write(int(DM_YATA[1]))
DM_pin_YATA_PAS.write(int(DM_YATA[2]))
yATA = _PASS(yATA,DM_YATA, yPASS)

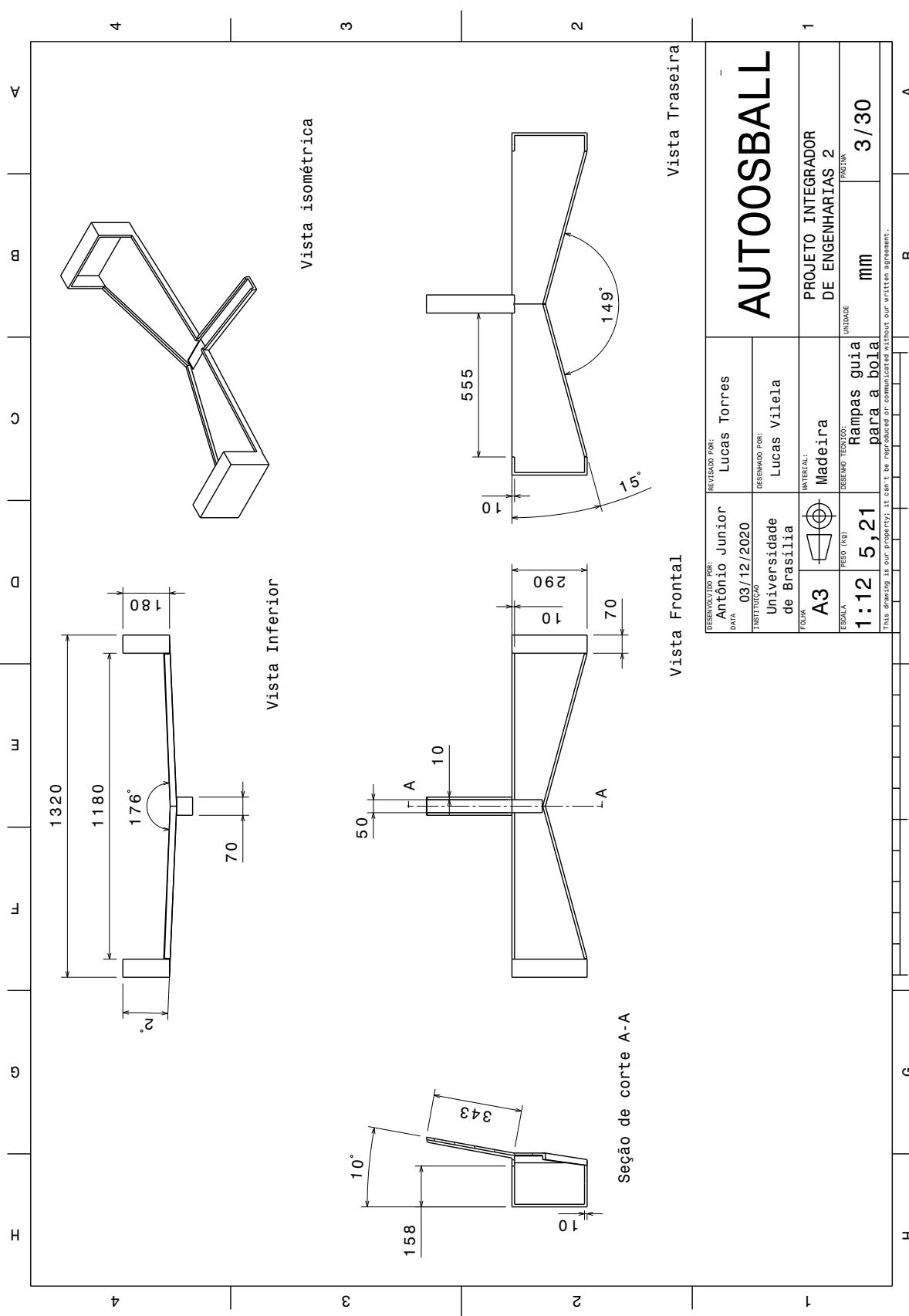
DM_pin_RATA_ENE.write(int(DM_RATA[0]))
DM_pin_RATA_DIR.write(int(DM_RATA[1]))
DM_pin_RATA_PAS.write(int(DM_RATA[2]))
aATA = _PASS(aATA,DM_RATA,aPASS)

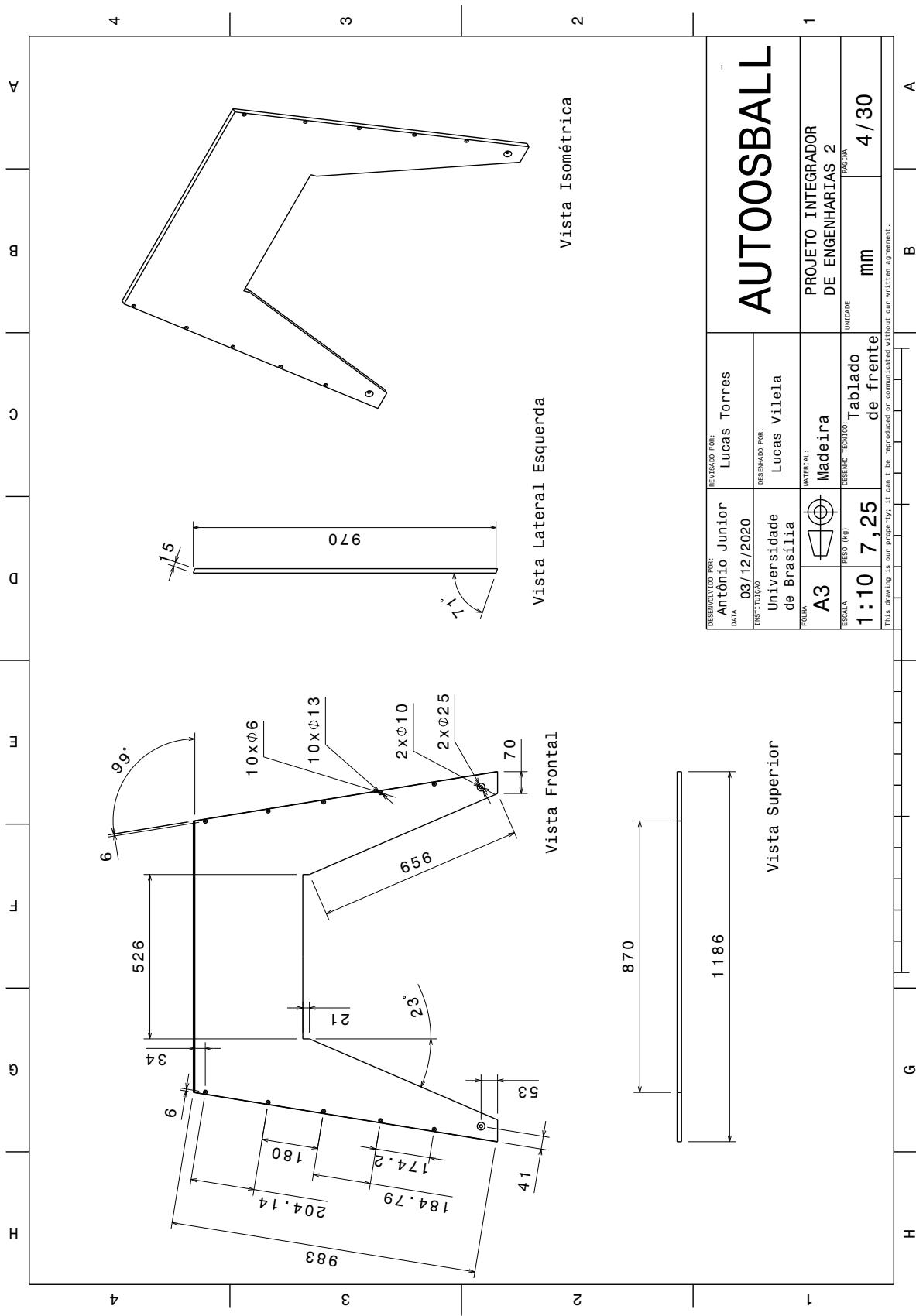
#Atualizando o campo simulado com a programa SESCdraw.py
_DRAW(yGOL,aGOL,yDEF,aDEF,yMEI,aMEI,yATA,aATA)
```

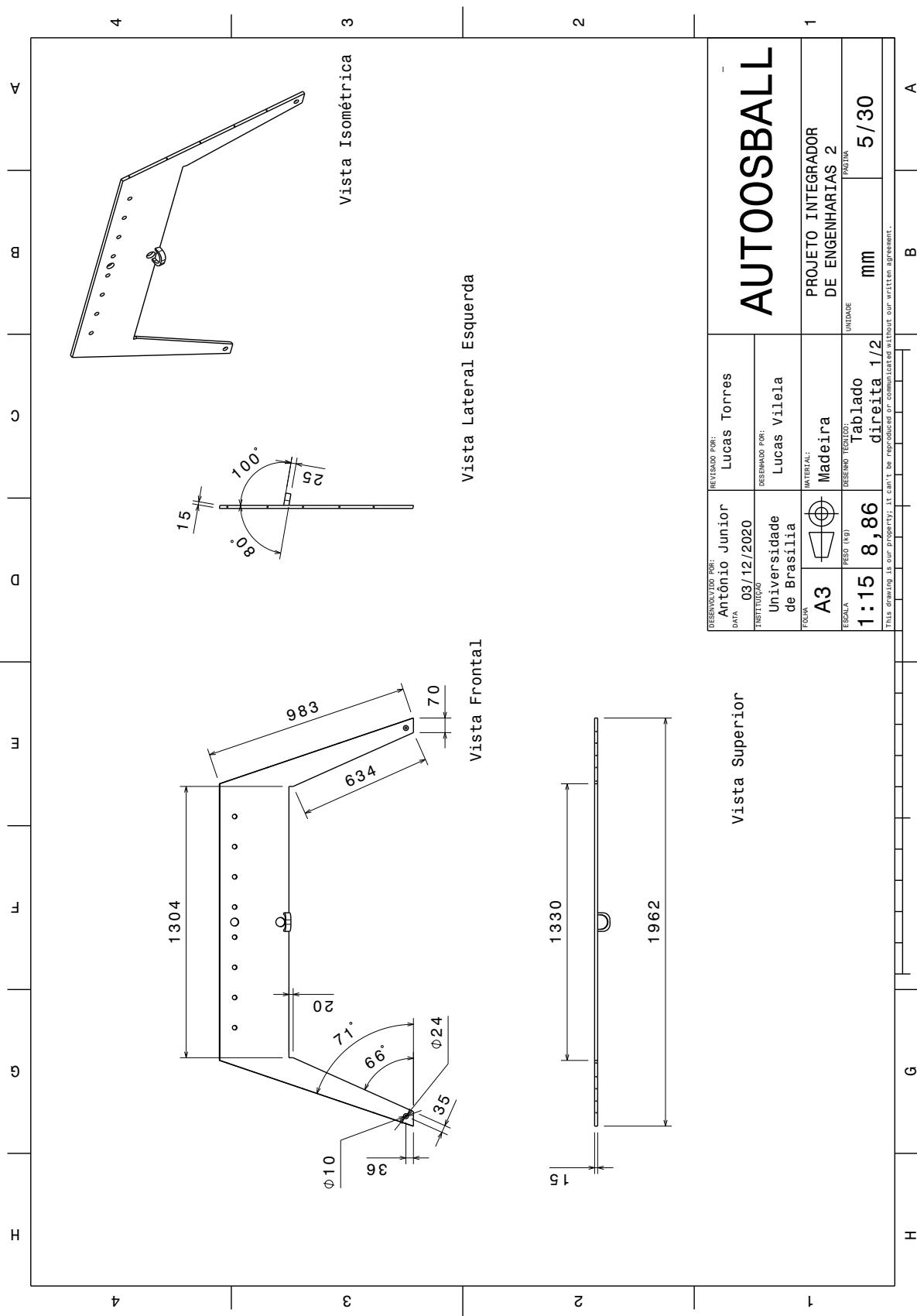
APÊNDICE C – Draftings

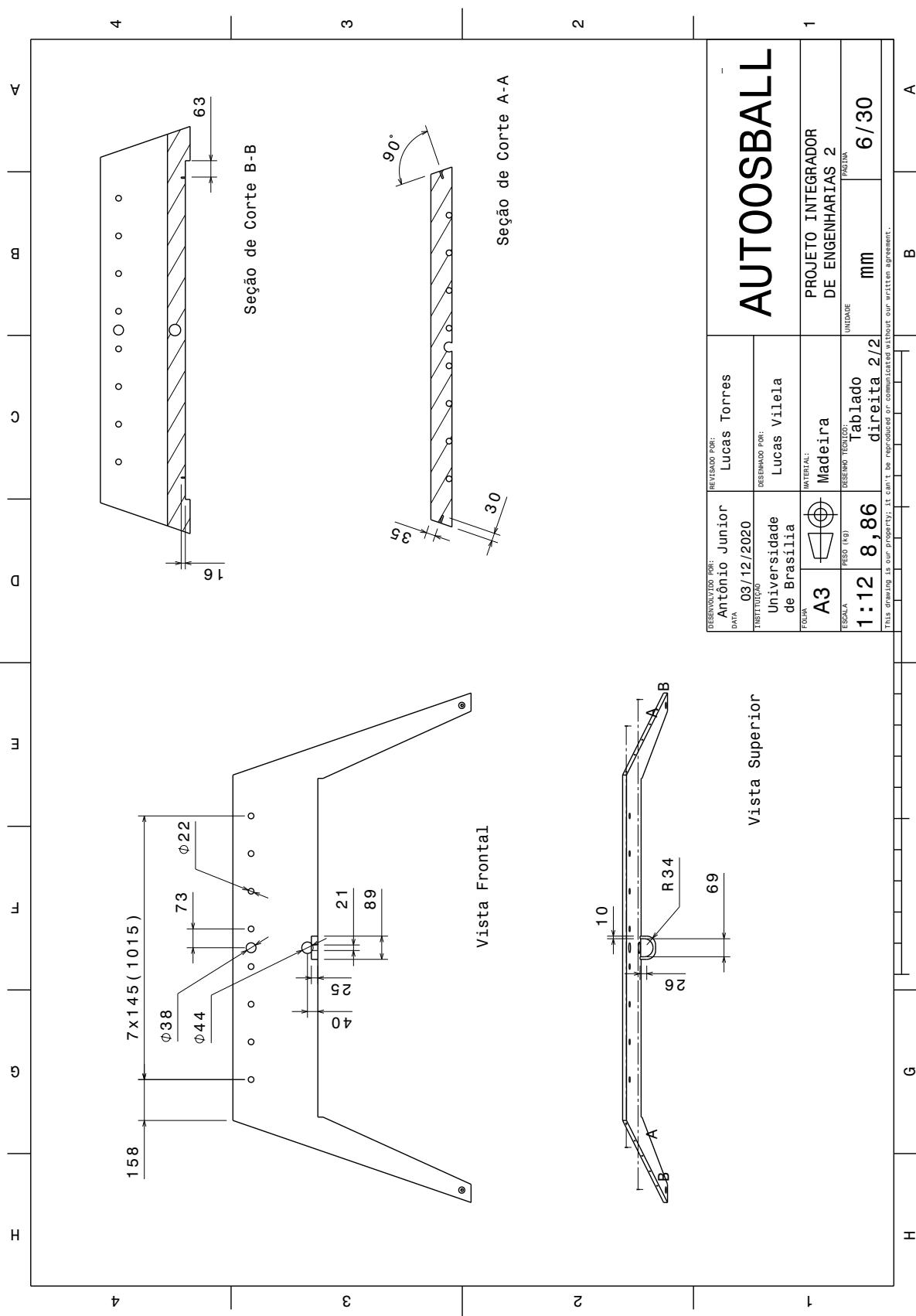


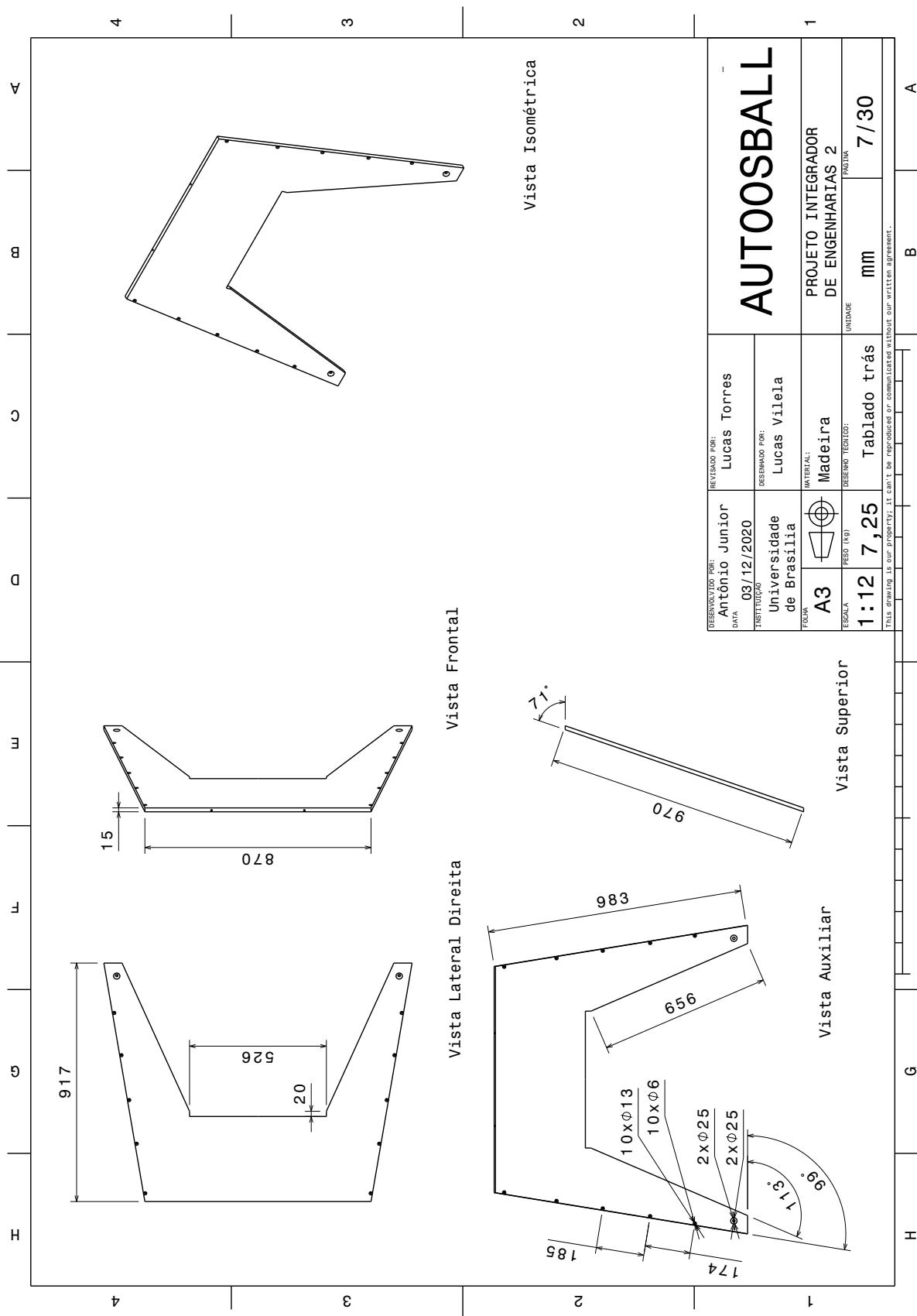


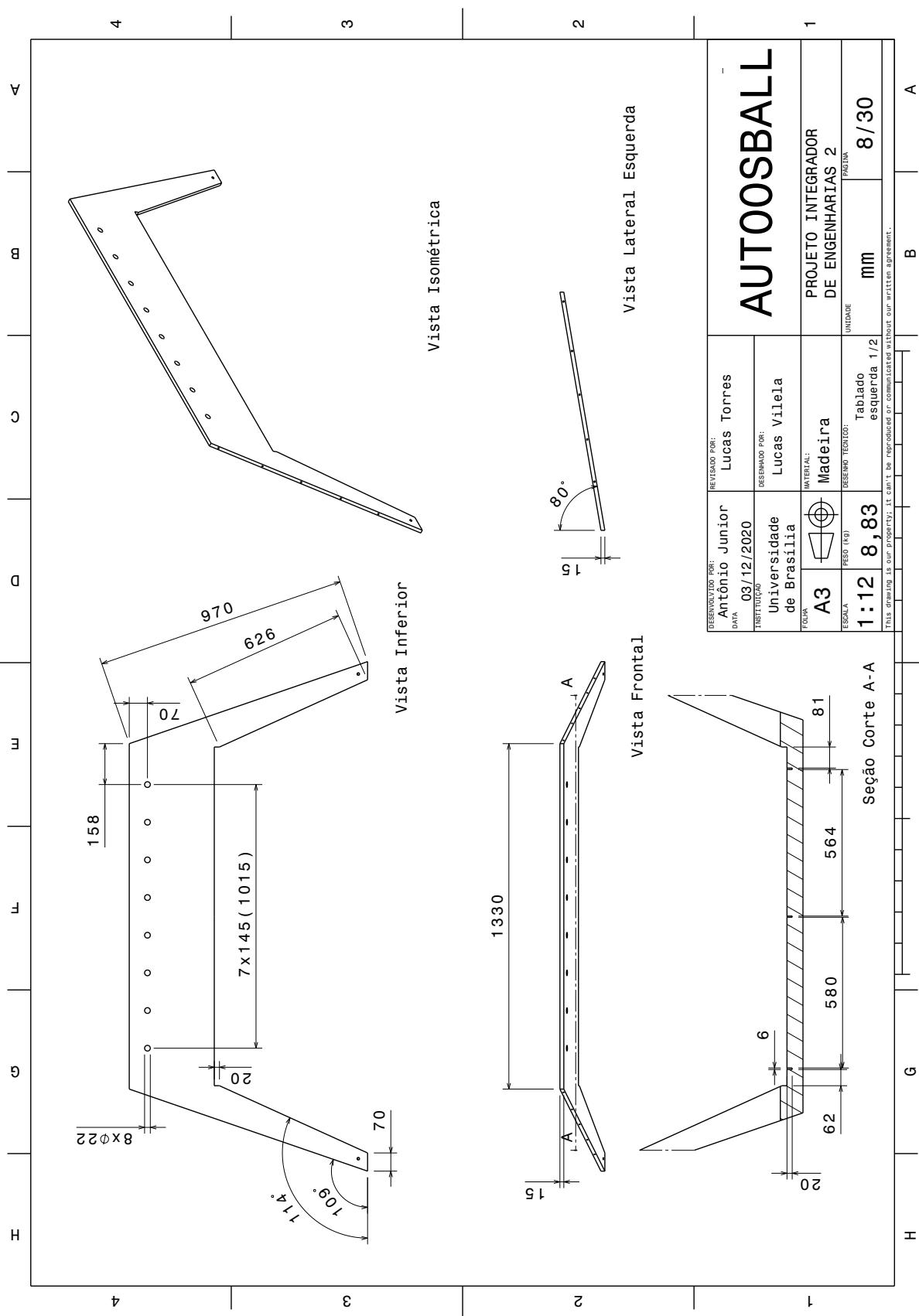


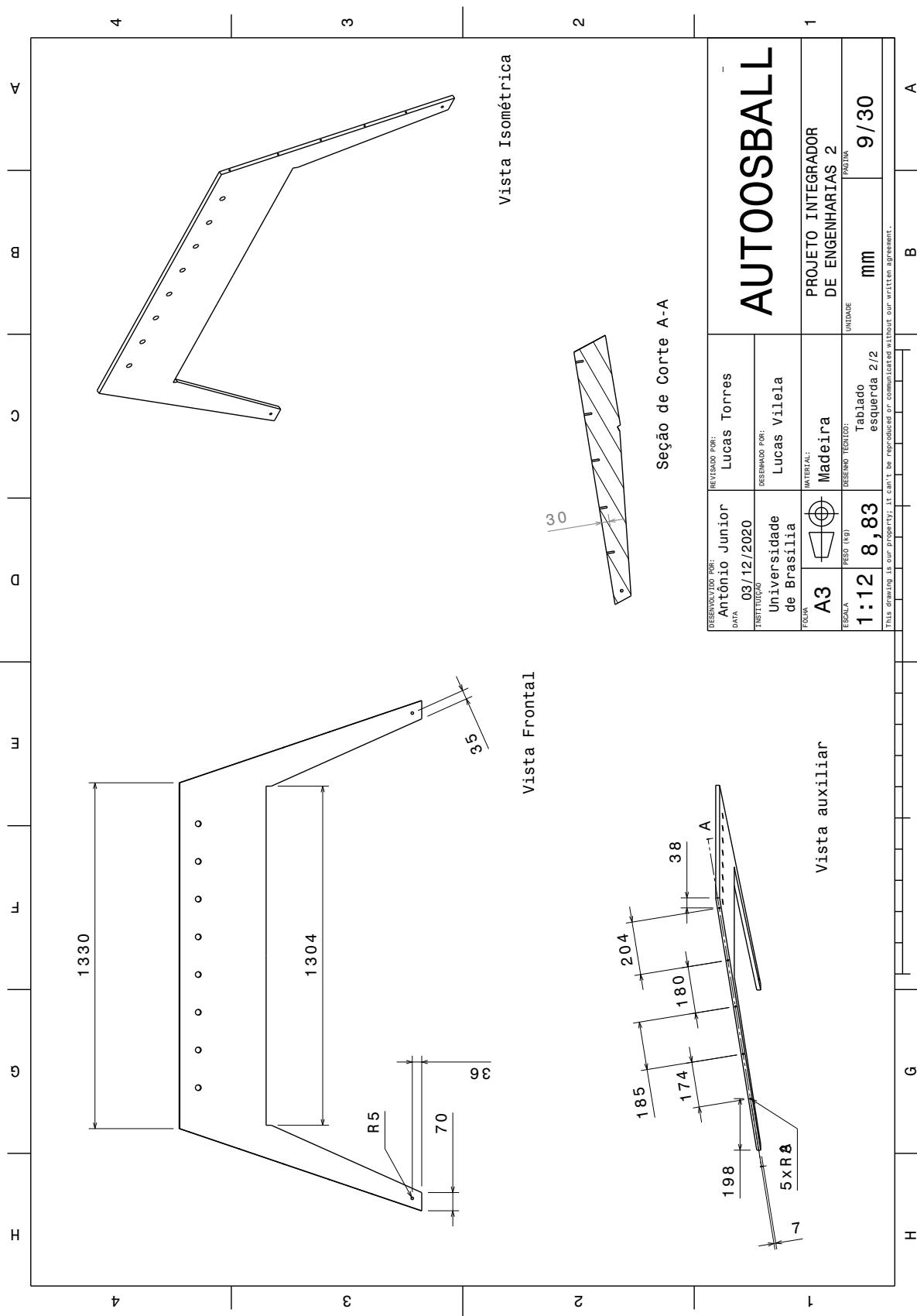


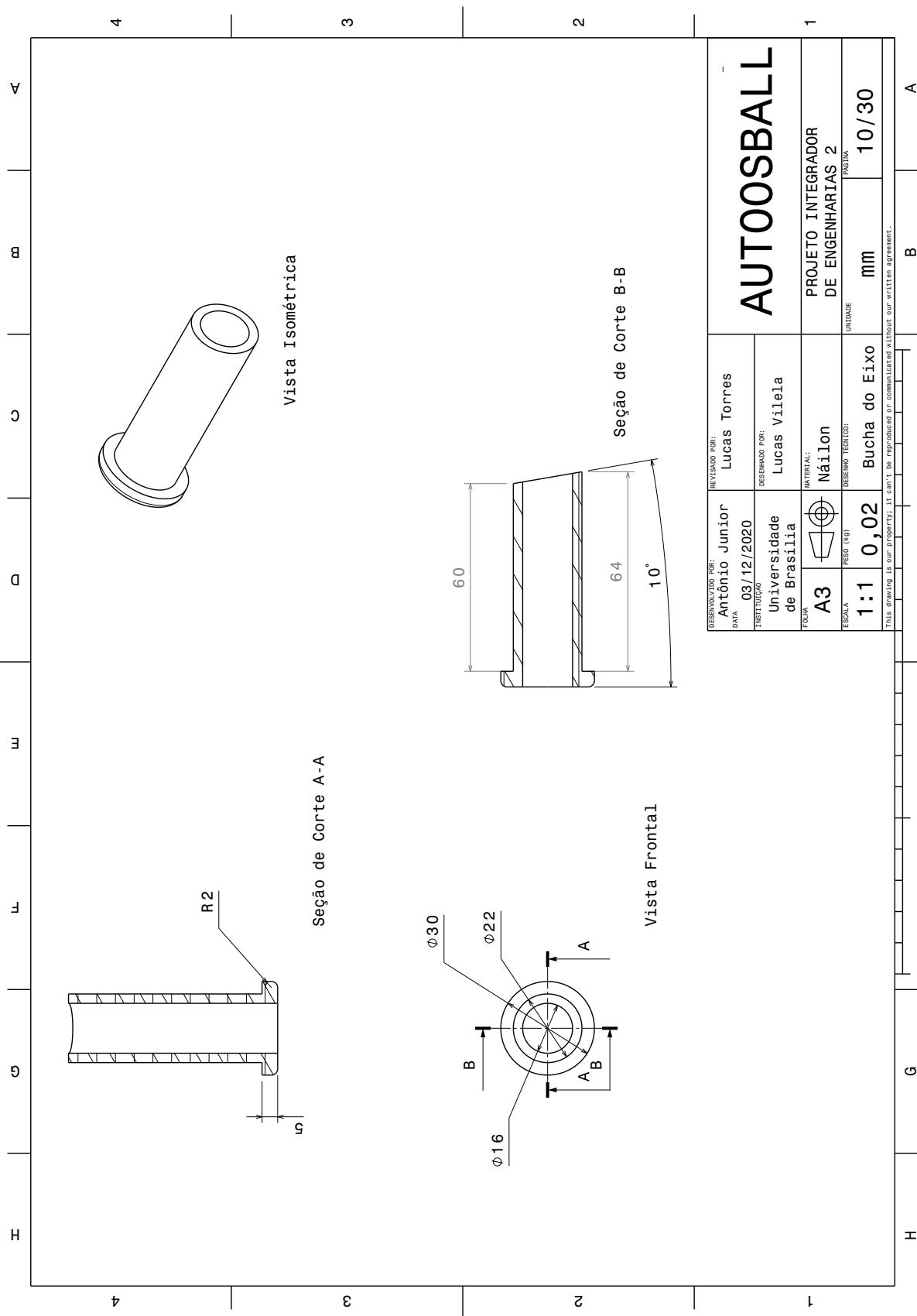


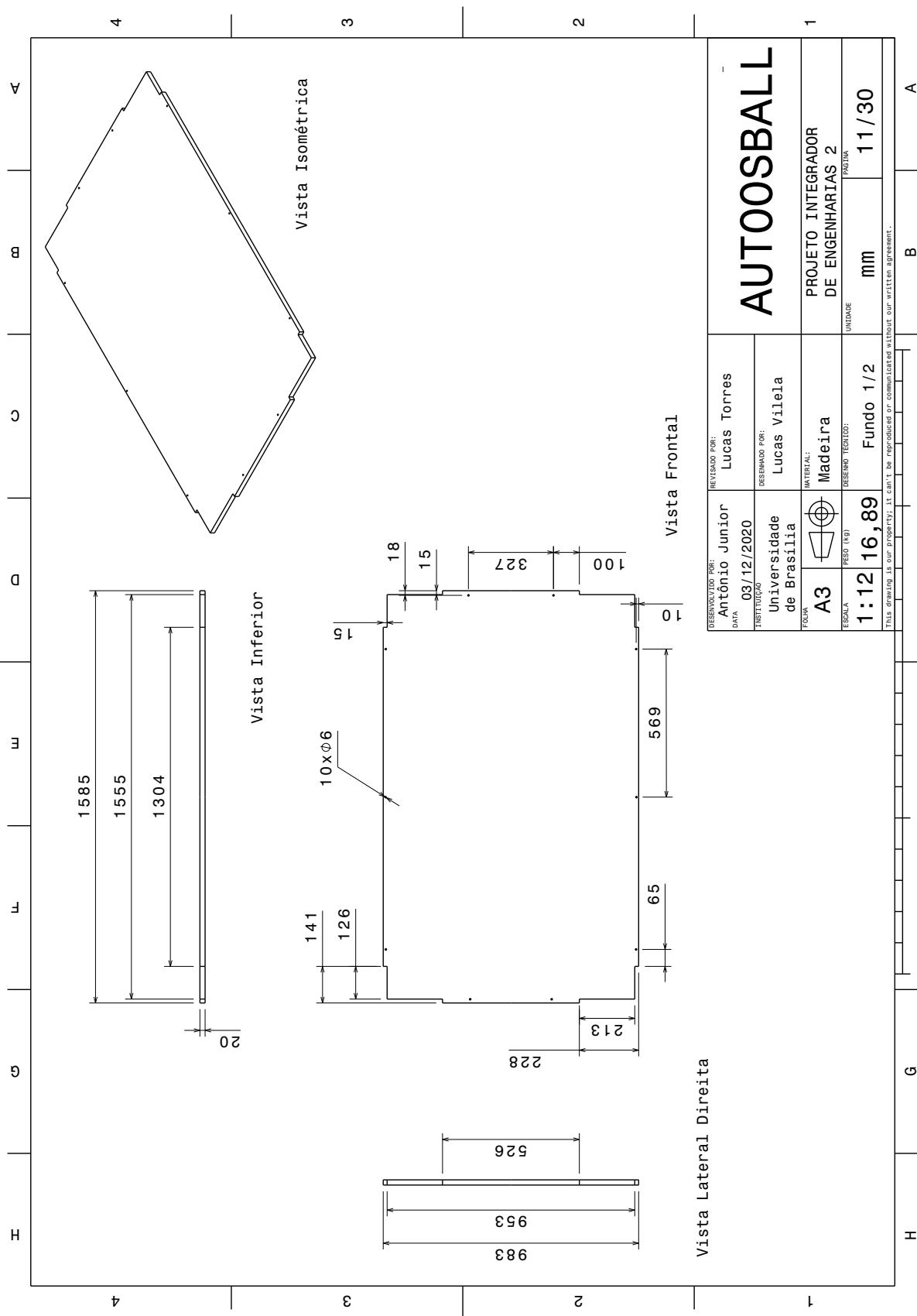


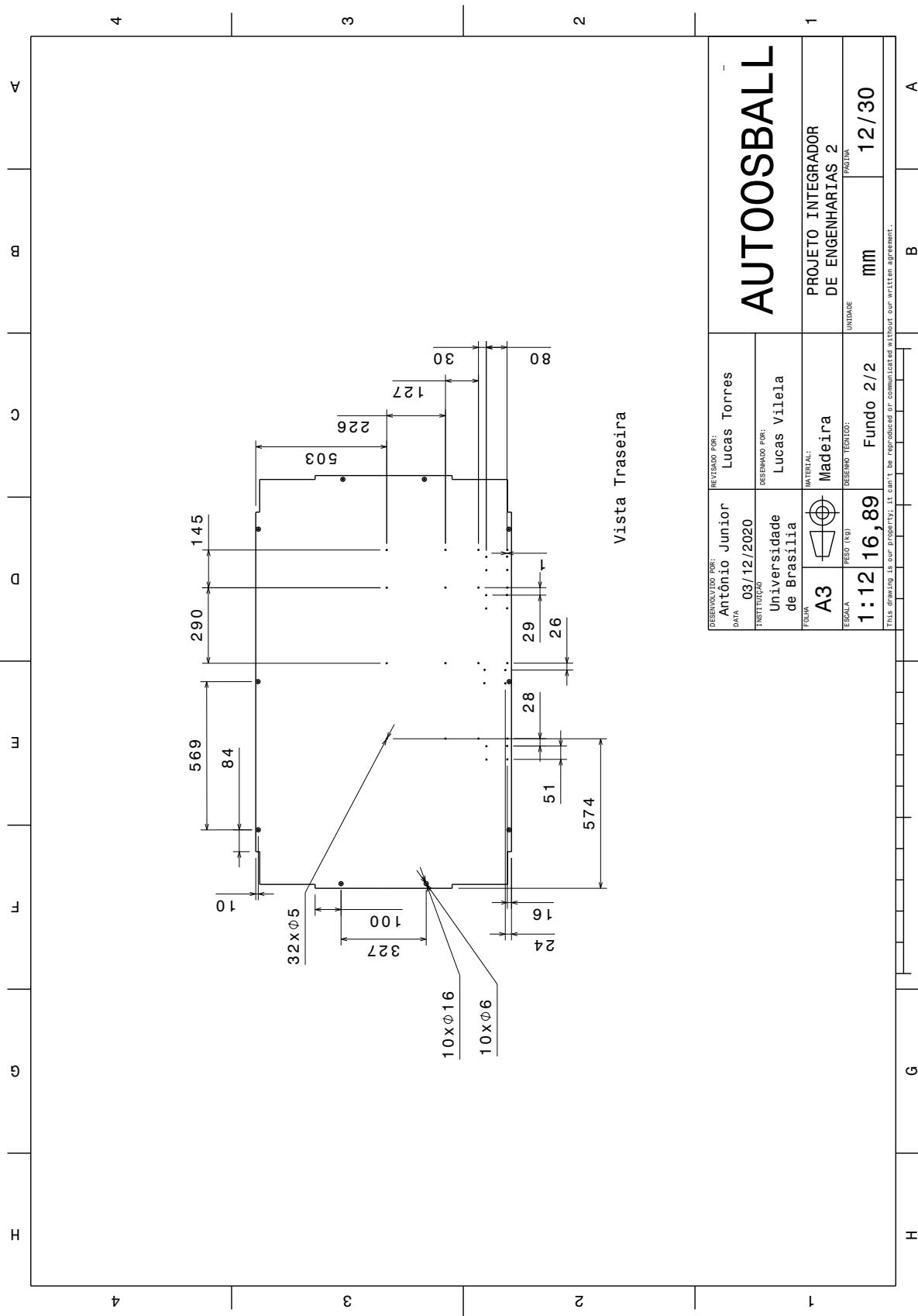


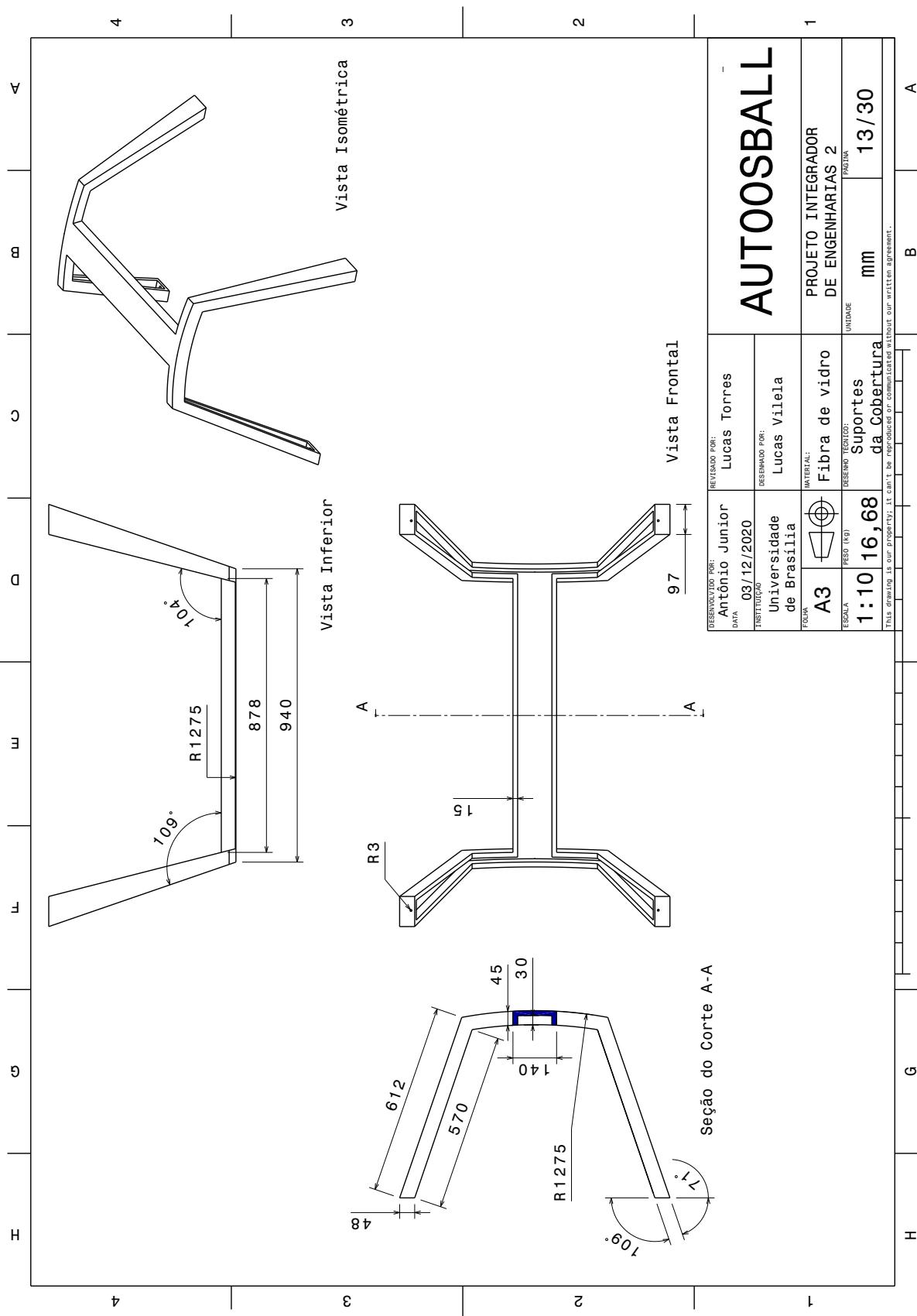


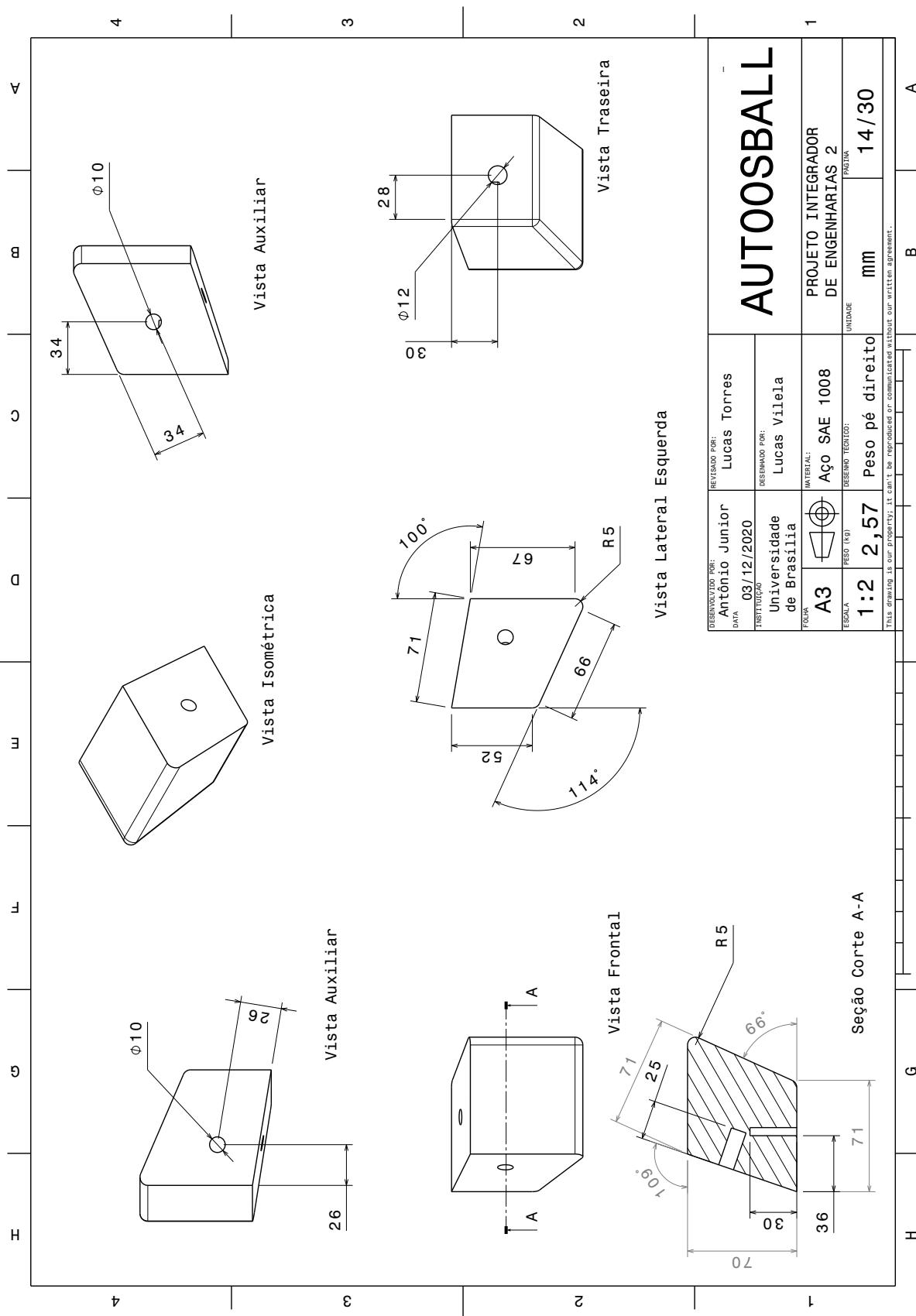


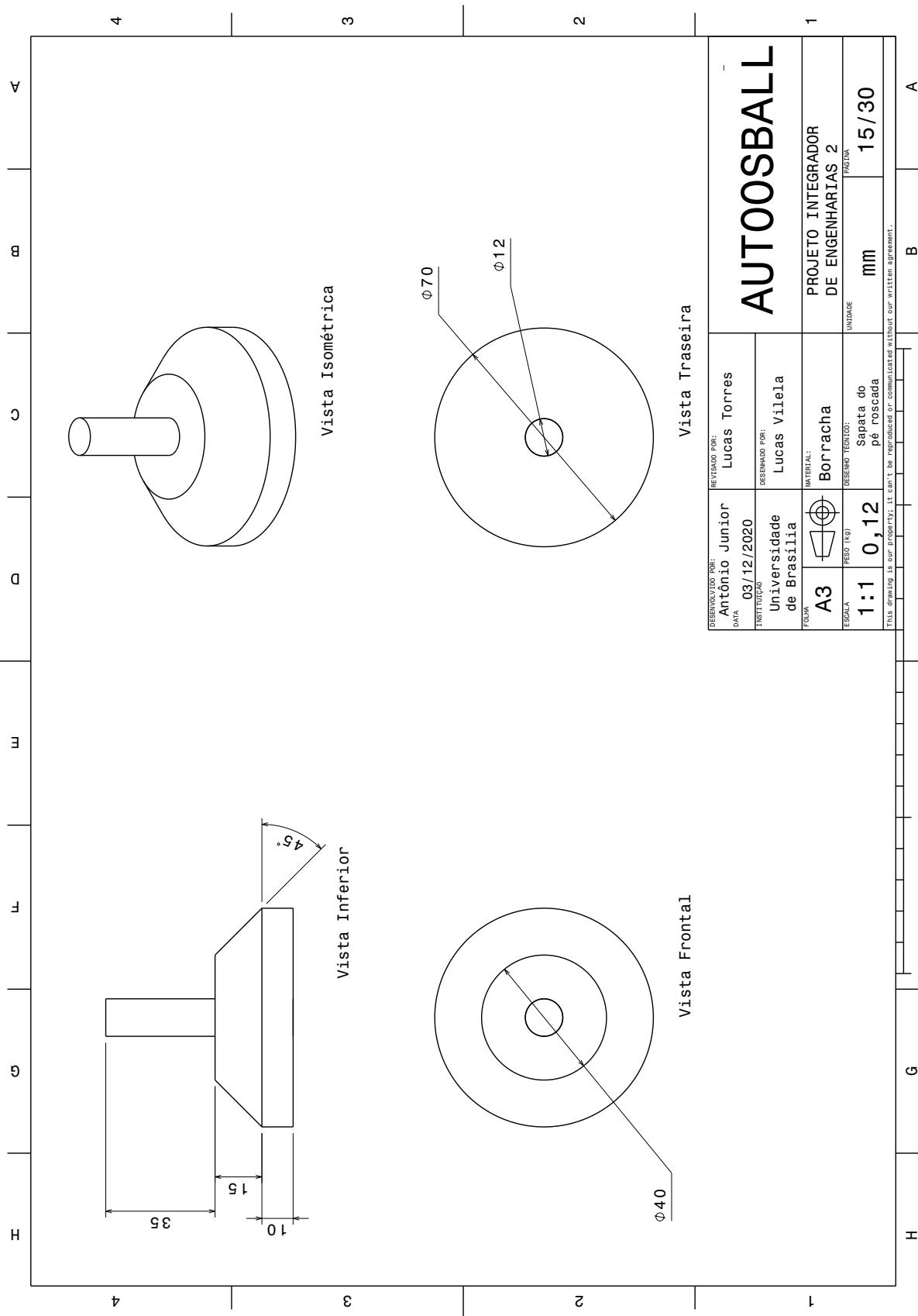


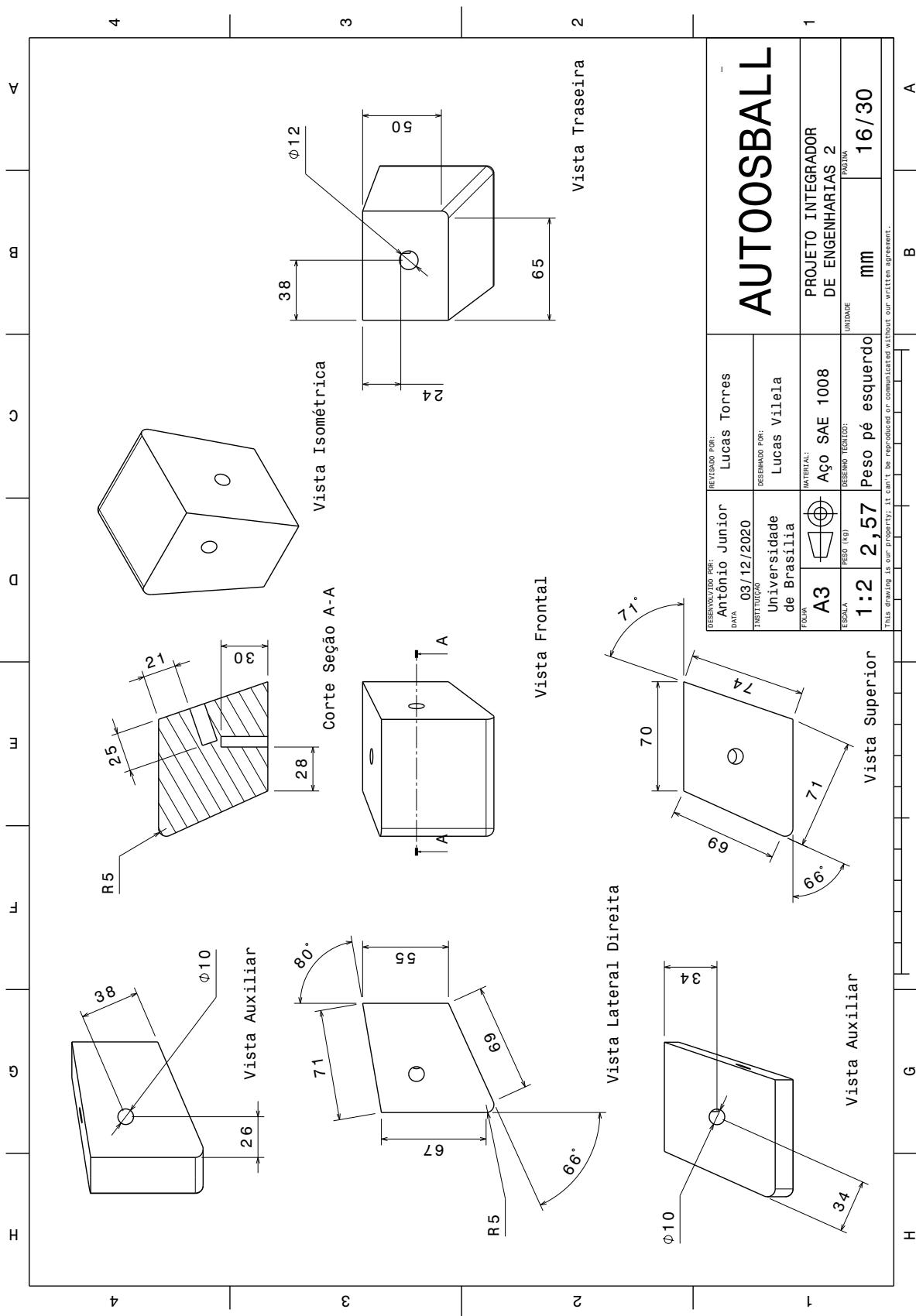


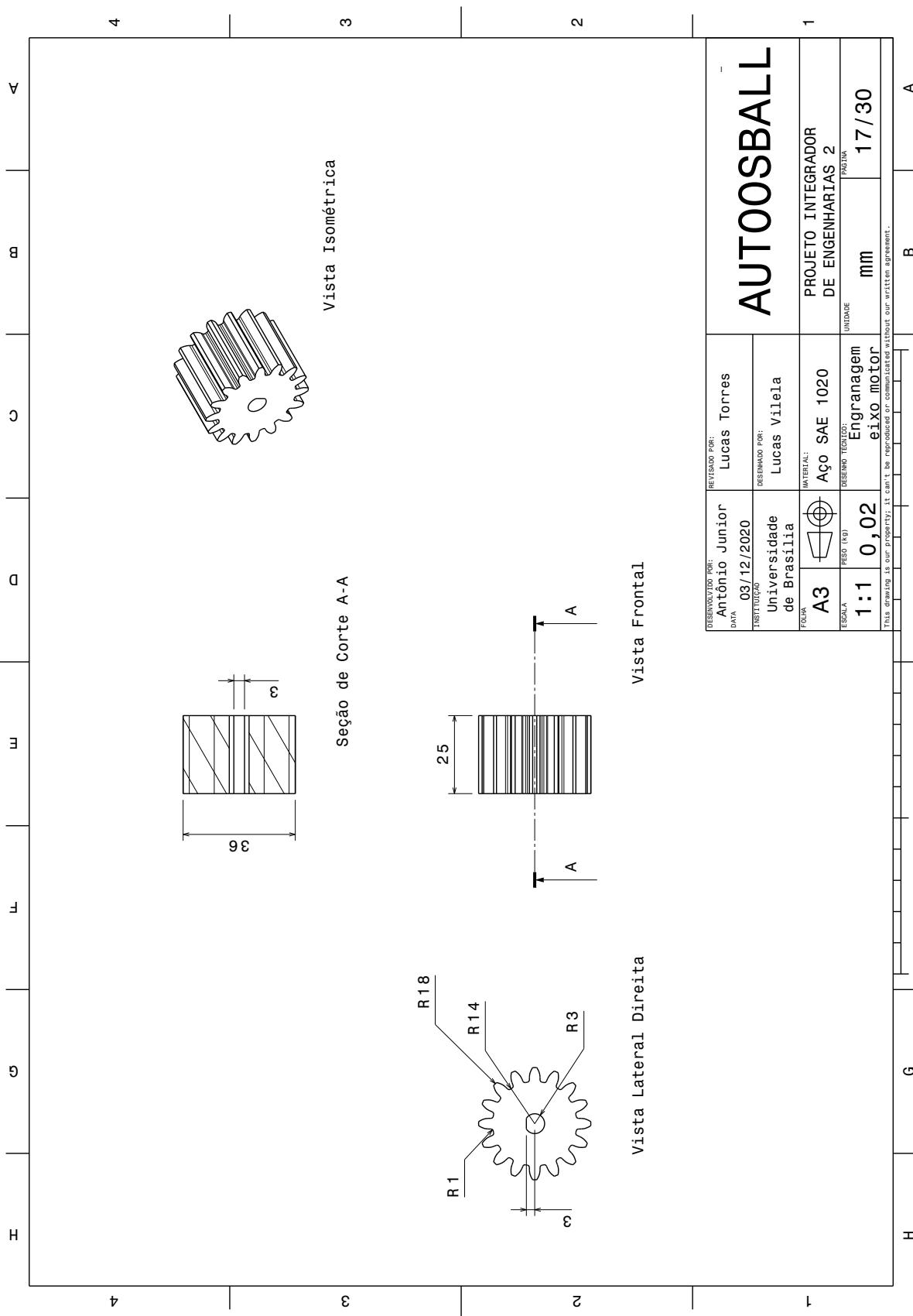


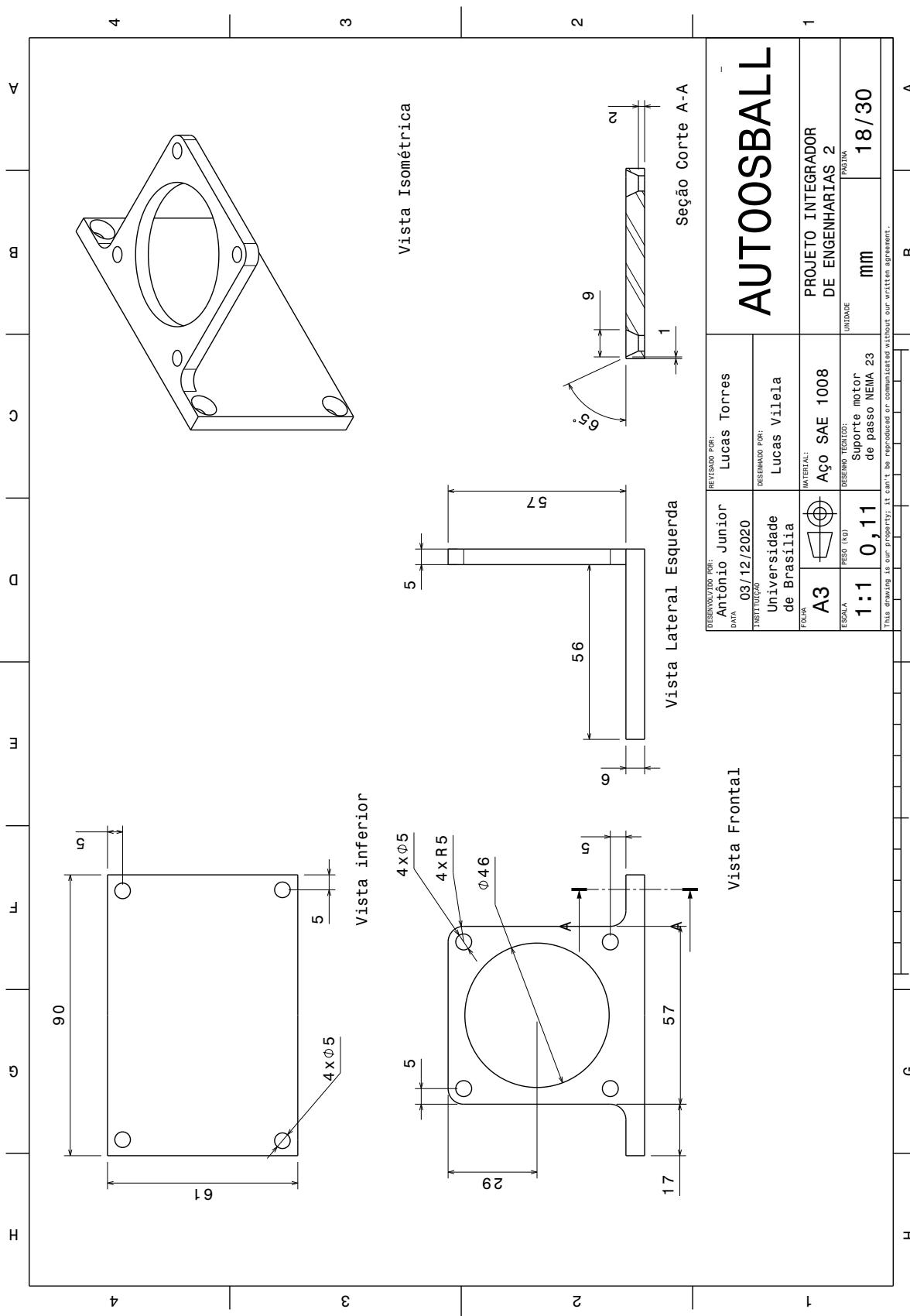


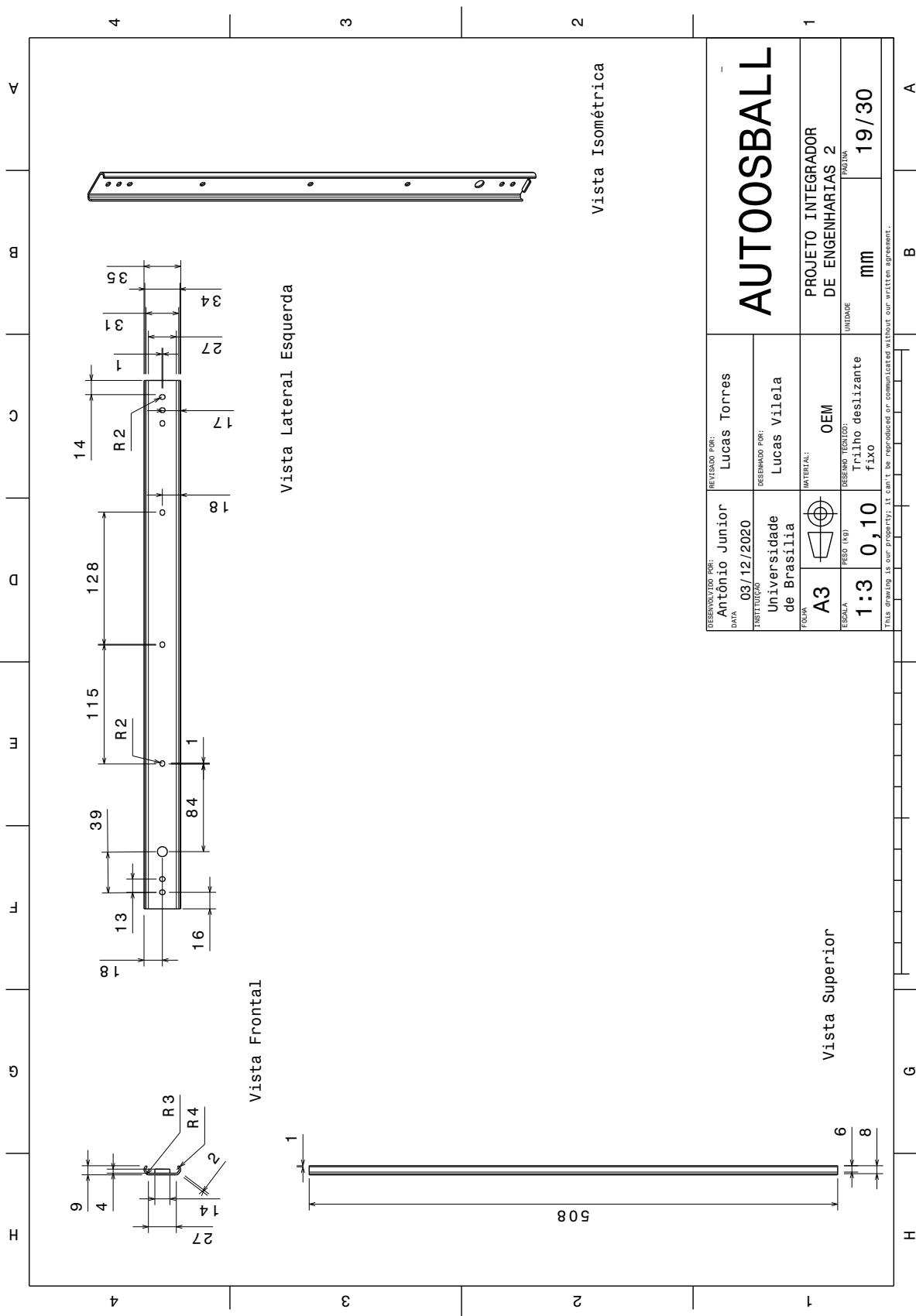


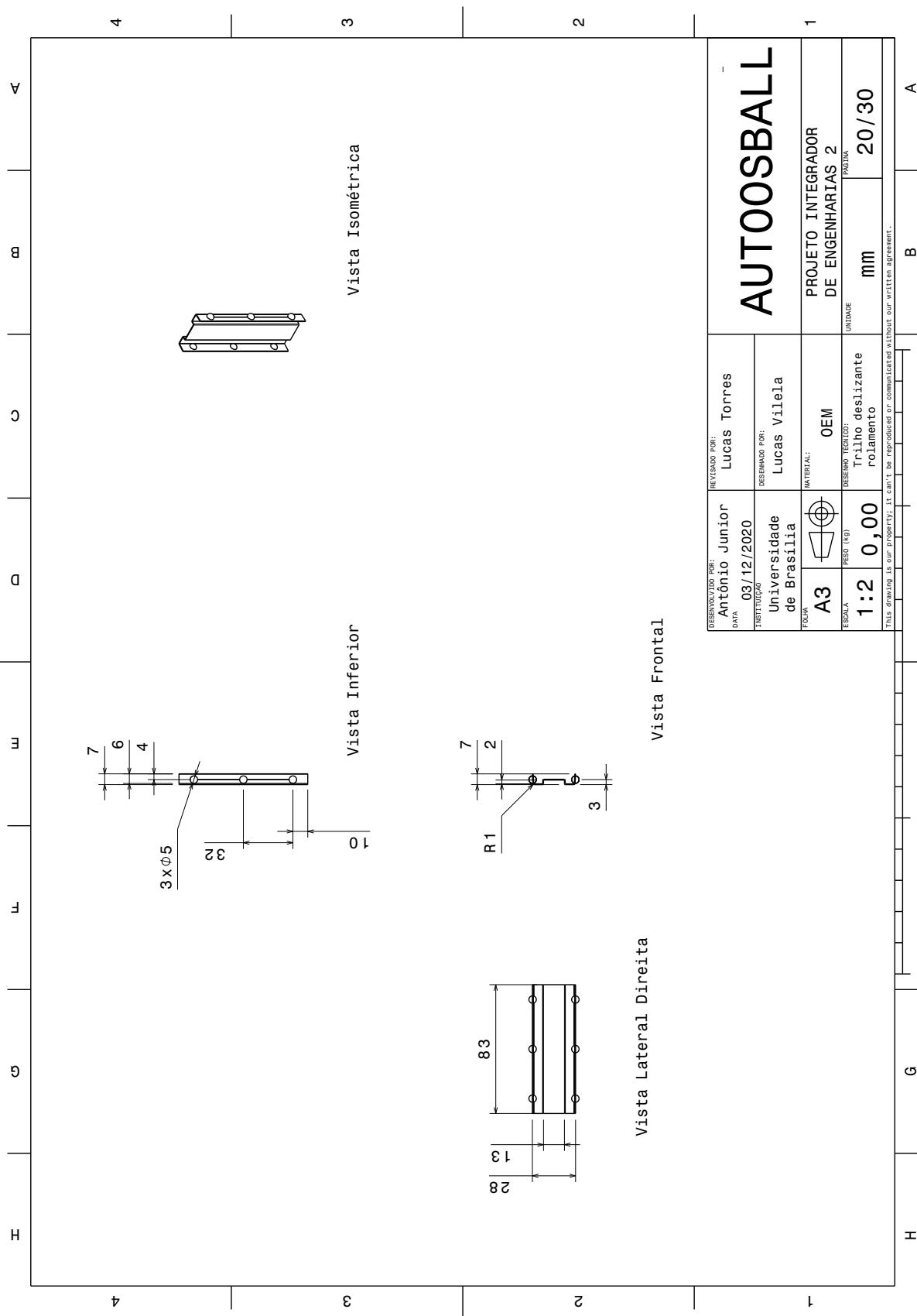


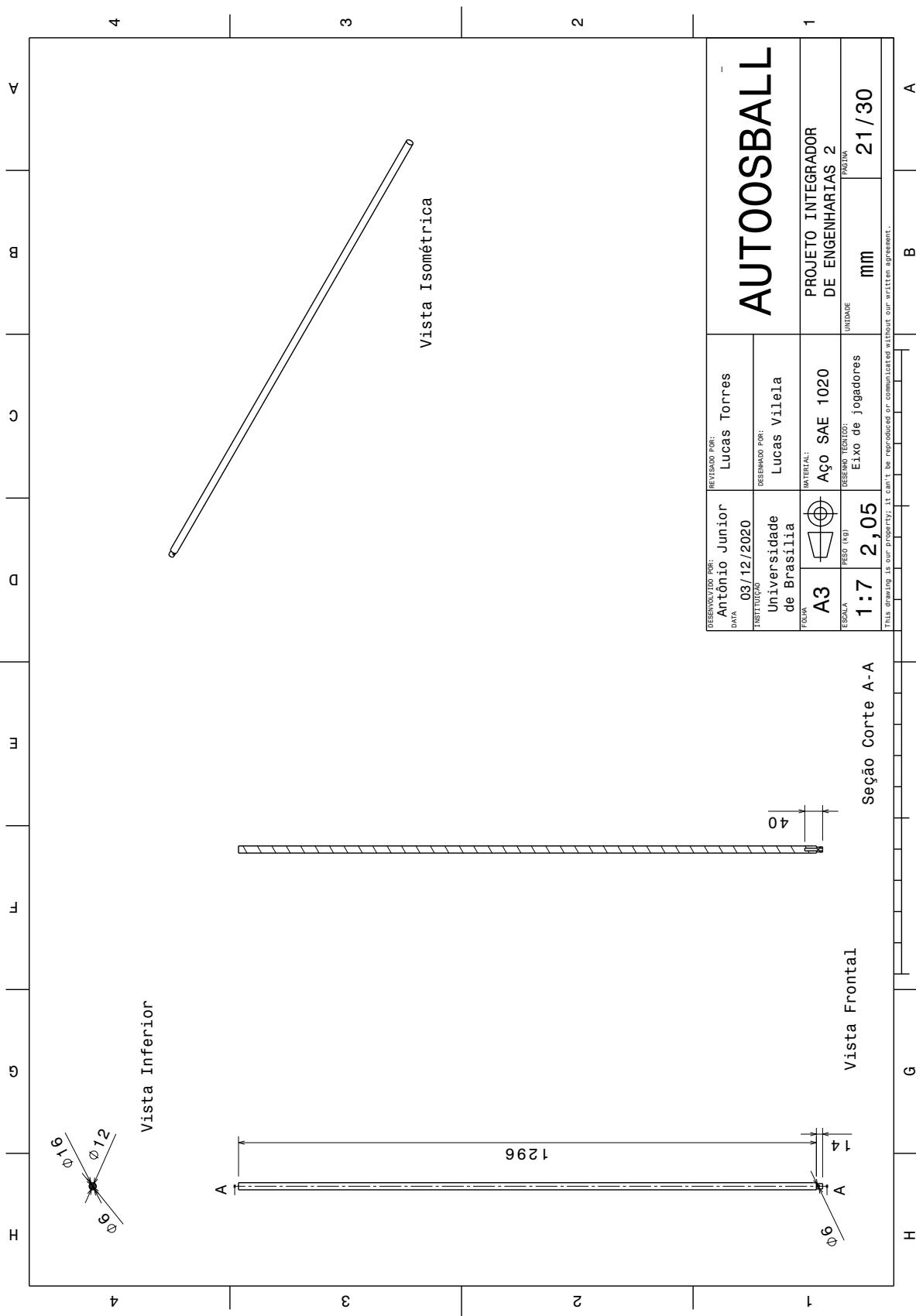


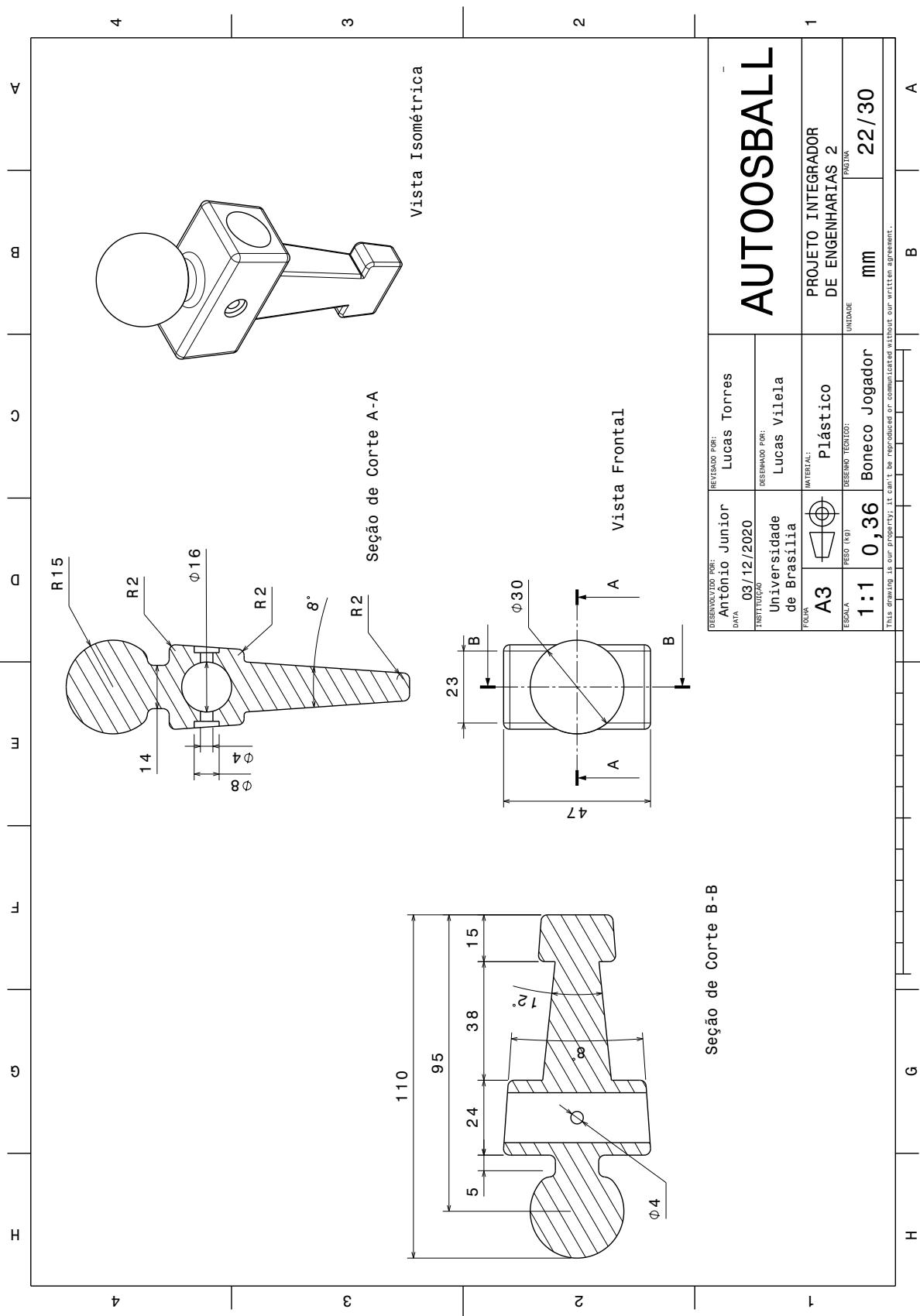


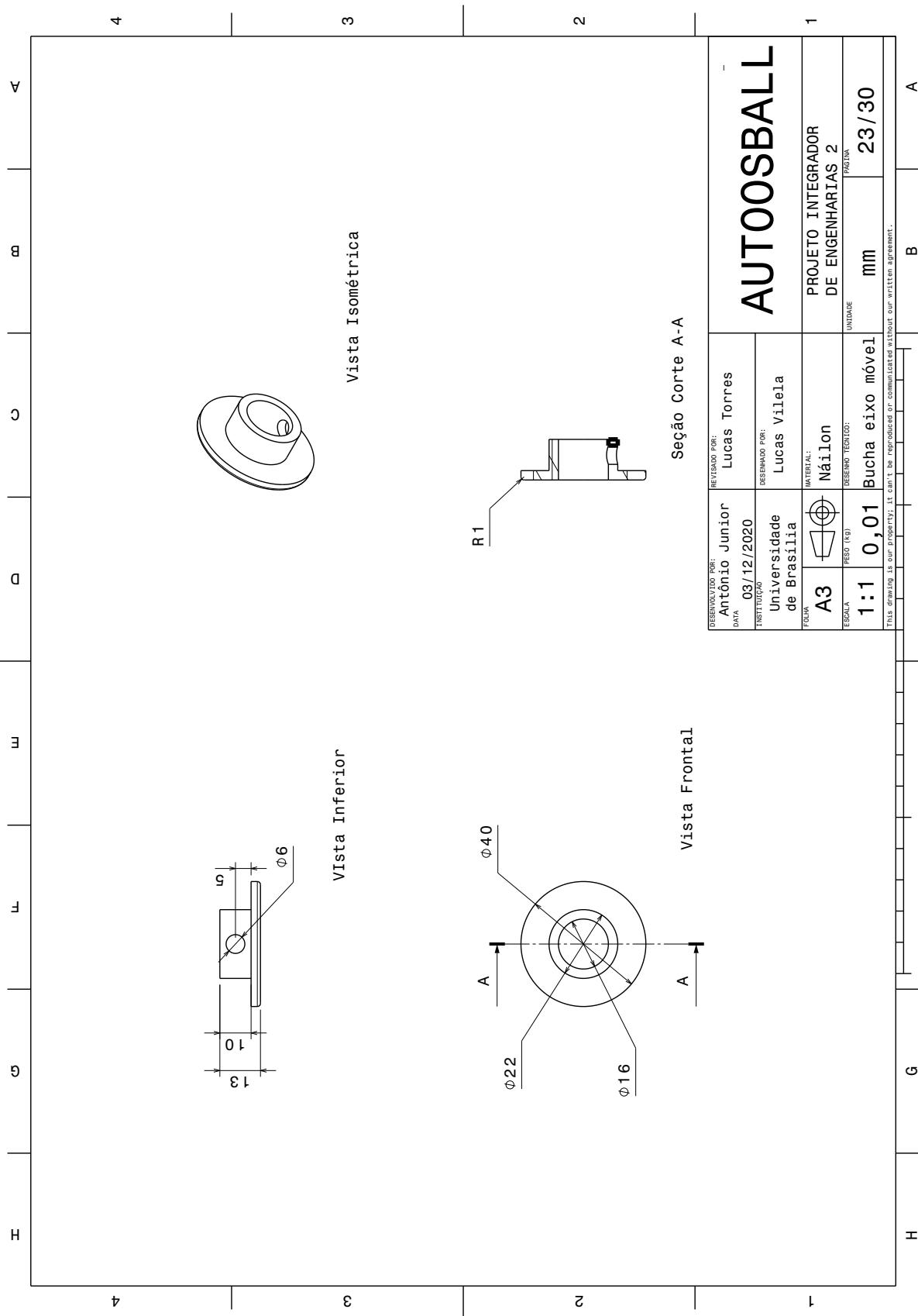


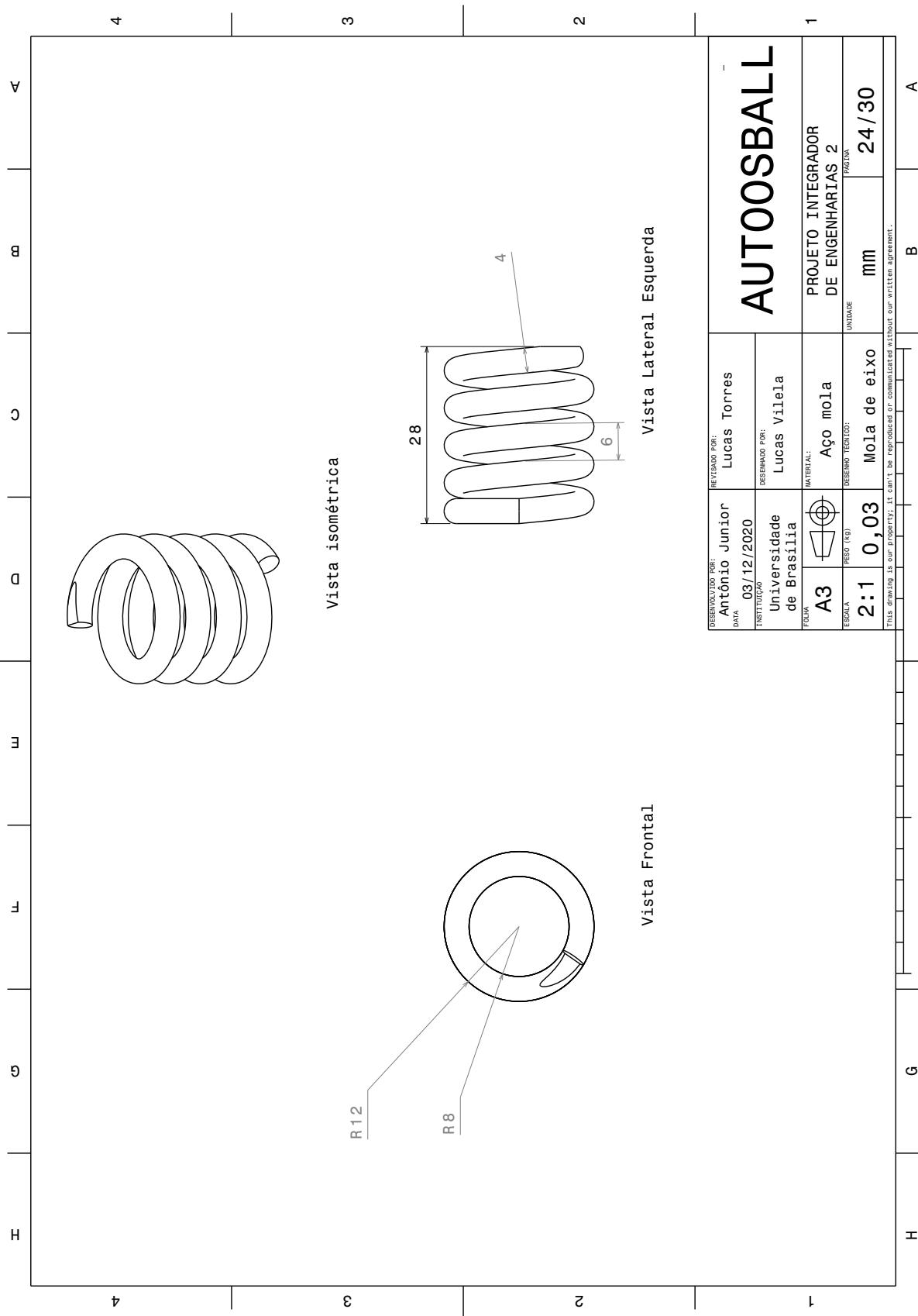


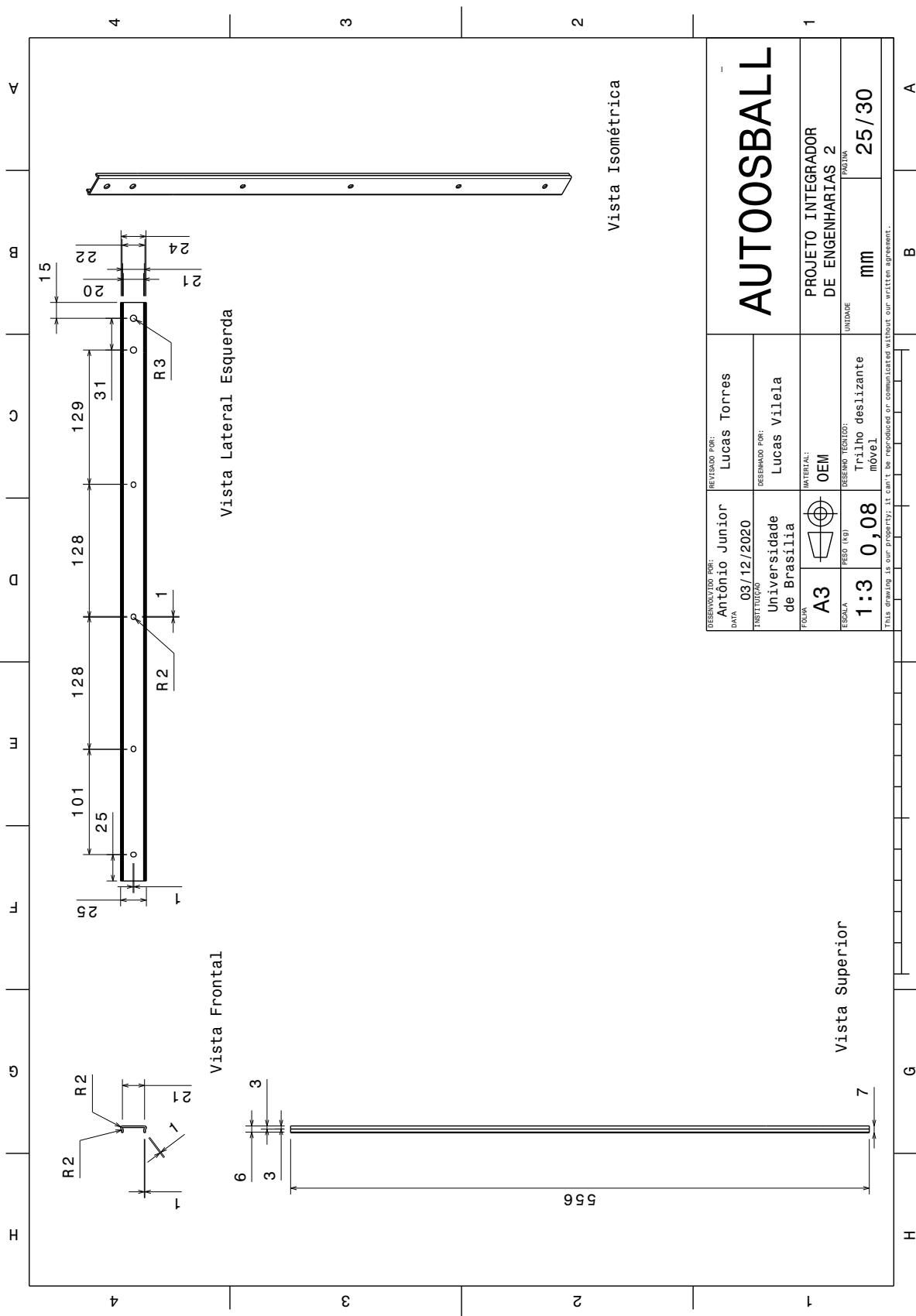


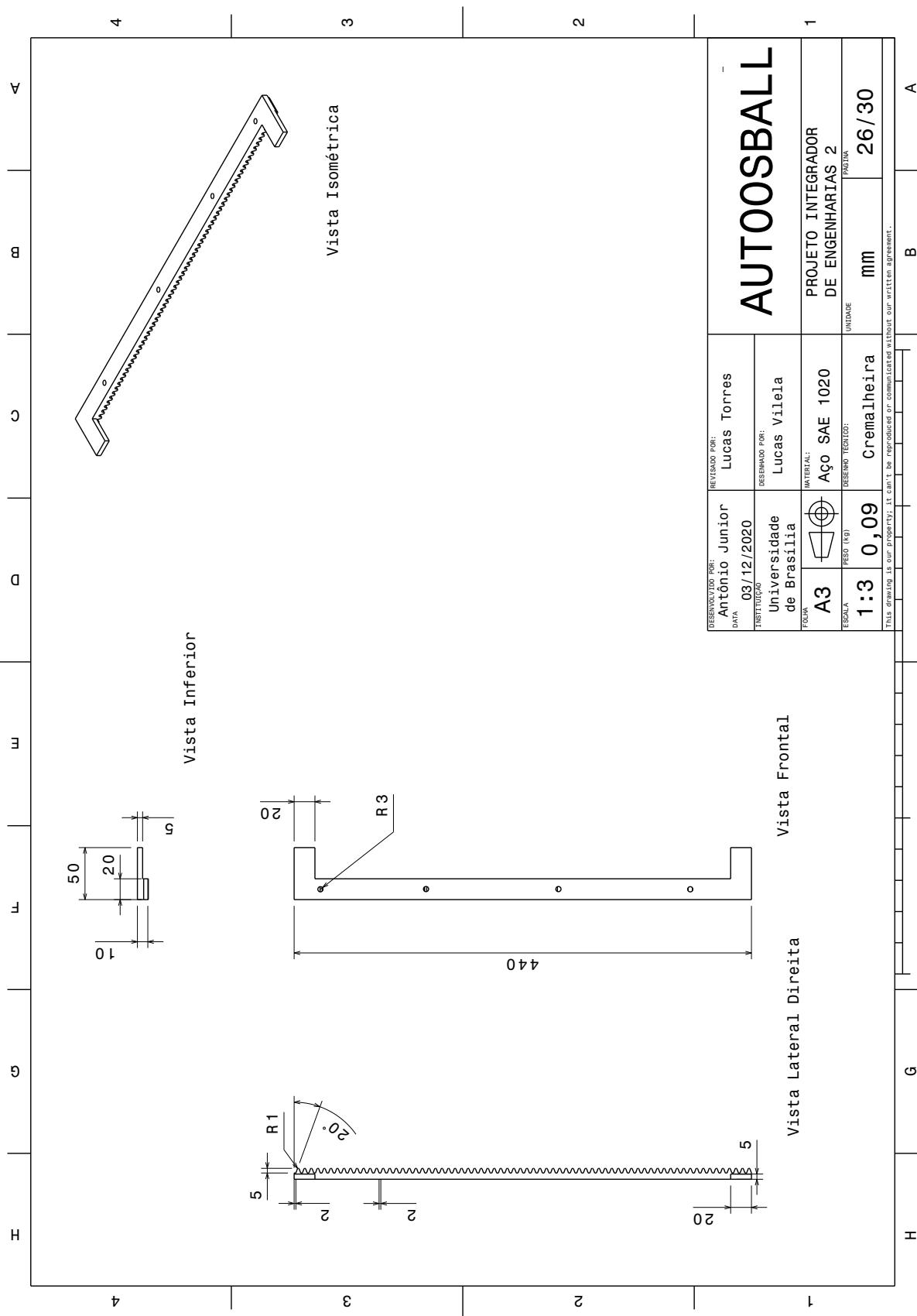


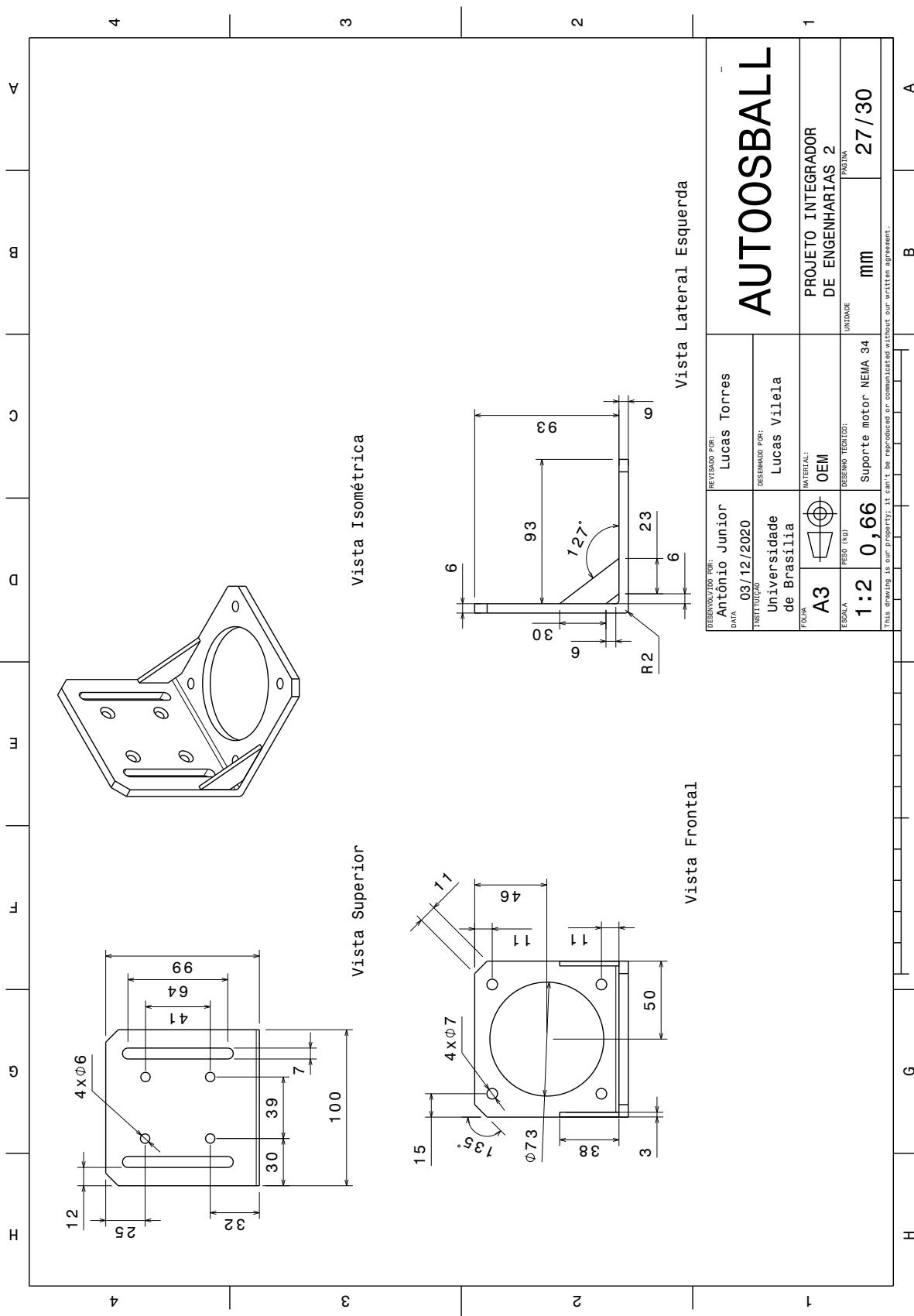


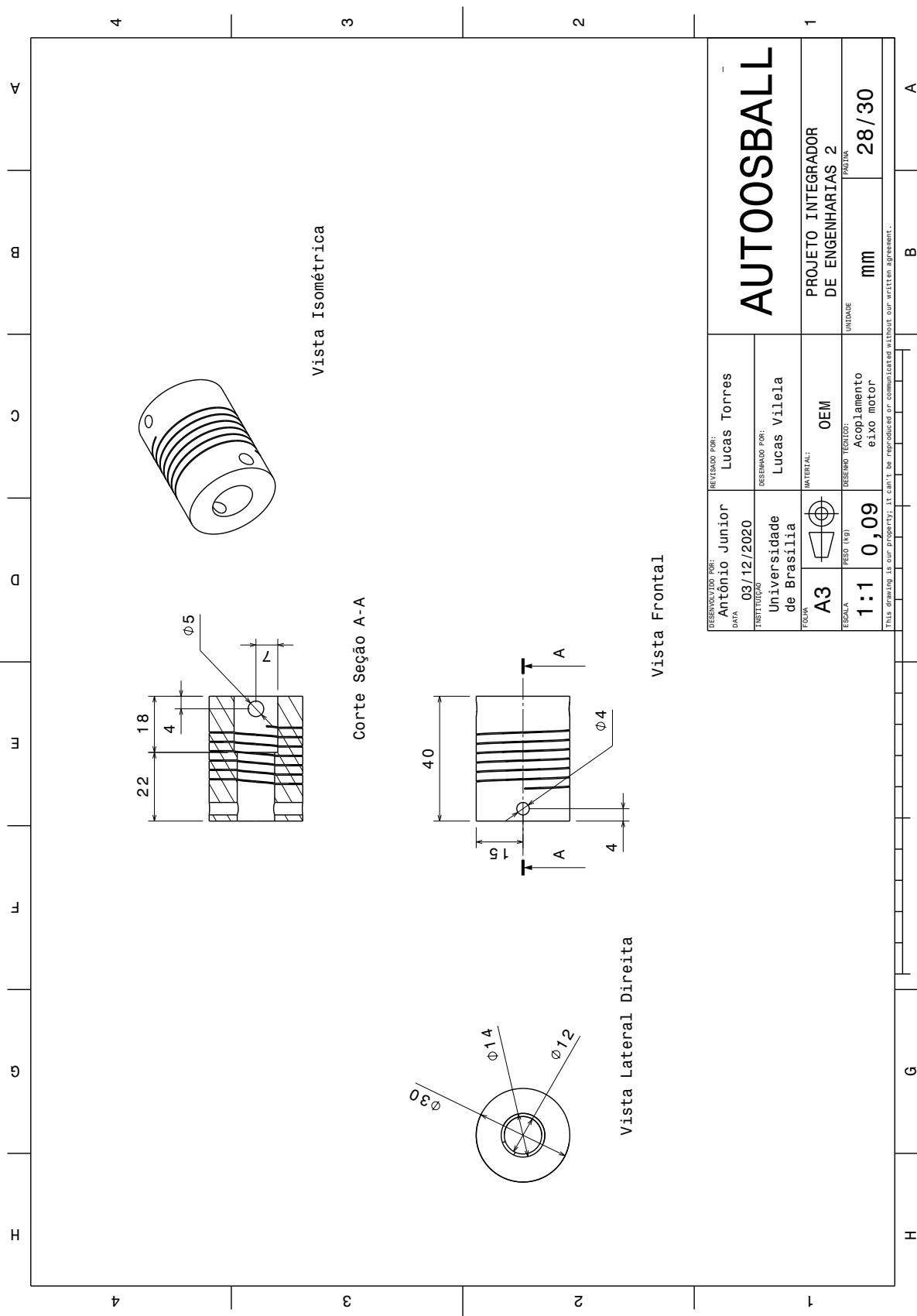


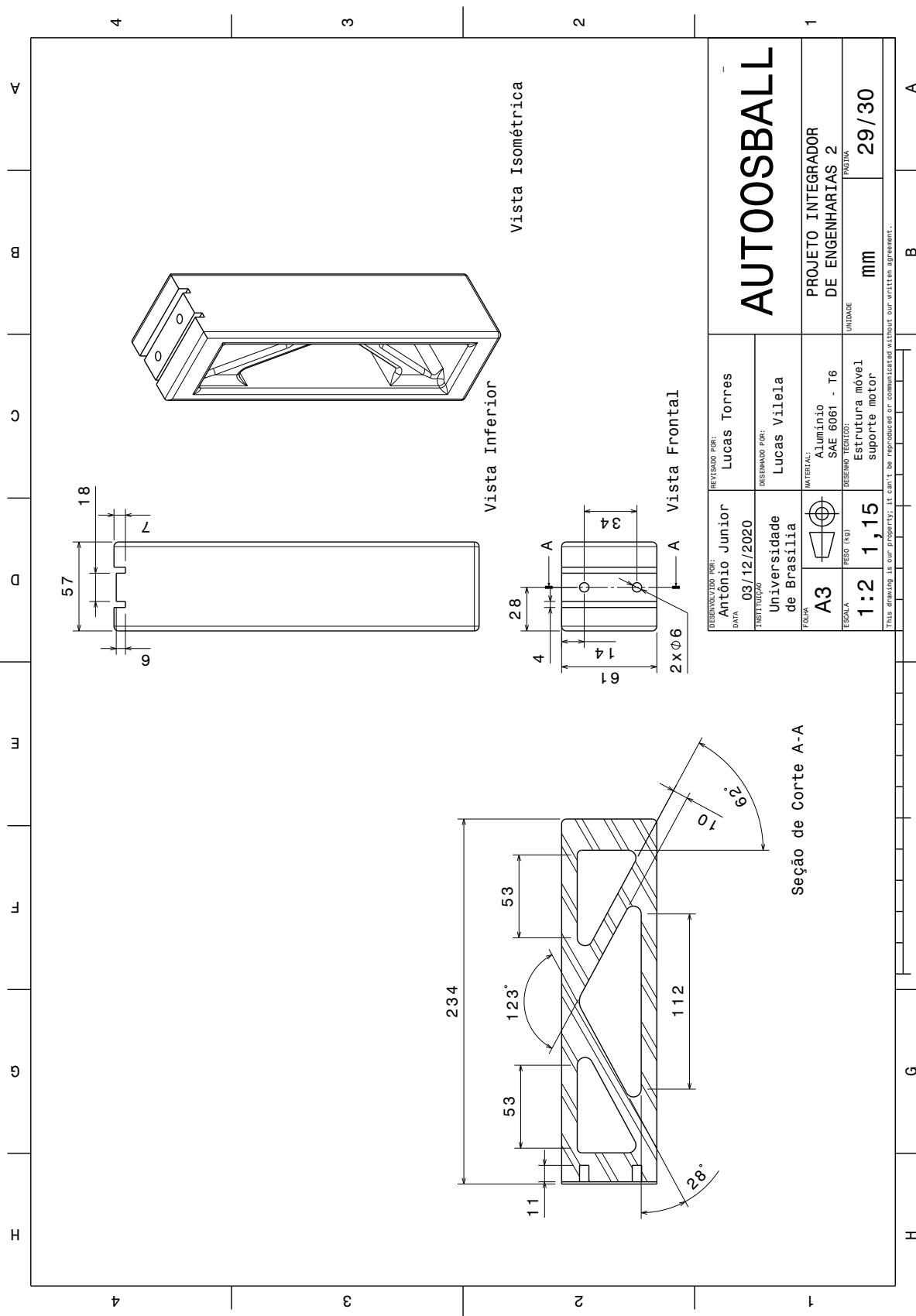


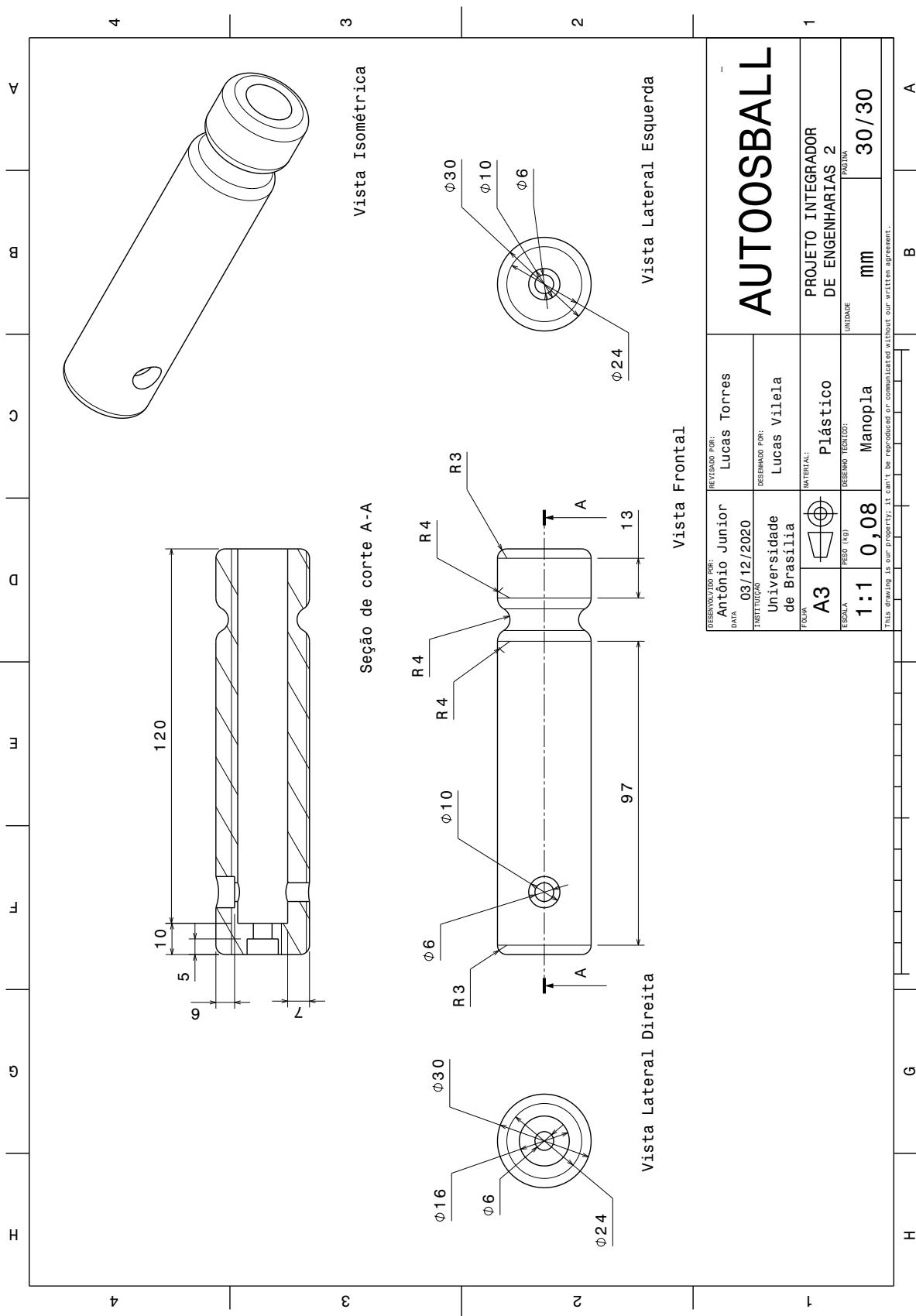












APÊNDICE D – Software

D.1 Código Fonte da Solução de Sofware

O código fonte para a solução de software é organizado por meio do software de controle de versão Git. Os repositórios Git do projeto são disponibilizado de forma aberta na organização no GitHub¹ criada para este projeto.

Nesta organização os repositórios utilizados são:

- **PUDM**: Contém a documentação relacionada ao PUDM;
- **pebolim-sim**: Contém o código relacionado à simulação na Unity;
- **foosball_decision_server**: Contém o código do servidor de decisão.

Na Tabela 38 estão elencadas as descrições de cada um dos eventos do modelo de dados PUDM.

Descrição	Link
Evento de registro da mesa	RegisterEvent
Evento de atualização do estado da mesa	StatusUpdateEvent
Evento de ação da mesa	ActionEvent

Tabela 38 – Descrição dos eventos do PUDM no repositório

Na Tabela 39 estão elencadas as funcionalidades do Servidor de Decisão.

Descrição	Link
Interpreta os dados gerados pelo PUDM	event_controller.py
Realiza a calibragem do campo, obtenção das coordenadas da bola, decodificação e processamento da imagem	image_controller.py
Define a ação a ser tomada por cada uma das hastes da mesa dependendo da direção e da posição da bola	decision_controller.py
Implementa a máquina de estados	machine_state_controller.py

Tabela 39 – Descrição das funcionalidades do Servidor de Decisão no repositório

¹ <https://github.com/pi2-2020-1-pebolim>

Na Tabela 40 estão elencadas as descrições de cada uma das funcionalidades do repositório da simulação da Unity.

Descrição	Link
Implementar o modelo de dados PUDM para comunicação entre os serviços	PUDM
Prover um campo virtual de pebolim com dimensões adequadas	Scenes
Implementar física para os objetos do campo virtual que sejam adequadas a uma mesa de pebolim real	Kicker
Simular o comportamento de motores na mesa	MotorMove
Implementar uma câmera virtual	CameraGrabber
Enviar o estado da mesa para o Servidor de Decisão via PUDM	StatusUpdateEvent
Executar ações recebidas do Servidor de Decisão via cliente PUDM	ActionEvent
Controlar o movimento da bola de pebolim a partir do teclado	KeyboardMove
Controlar o movimento de chute a partir do teclado	KeyboardMove

Tabela 40 – Descrição das funcionalidades da Unity no repositório

D.2 Detalhes da Modelagem PUDM

A modelagem de dados está escrita em uma forma adaptada de JSON, com a adição de comentários (marcados pelo símbolo "/*") e no lugar dos valores se encontra o tipo de dado esperado para o atributo. Nessa descrição: "[]" indica uma lista, "{}" um objeto, "..." indica a continuação de outros objetos do mesmo tipo em uma lista.

Os atributos *timestamp*, *version* e *eventType* são comuns a todos os eventos, pois são eles que descrevem o tempo em que o evento foi enviado, a sua versão e o tipo de evento que foi enviado.

- Atributos comuns:

```
{
  "eventType": String
  "timestamp": int,
```

```

    "version": int,
}

• RegisterEvent: Enviado do cliente para o servidor por meio de uma requisição HTTP POST ao início de sua execução. Serve para identificar a mesa e suas características no servidor.

{
    "eventType": "register",
    "timestamp": int,
    "version": int,
    "fieldDefinition": { // Contem definições do campo de jogo
        "dimensions": [float, float], // comprimento e largura em cm
        "lanes": [ // Lista que contem a descrição de cada
            // linha controlada pela maquina (cada elemento
            // representa uma linha)
        {
            "laneID": int, // Id da linha, a linha mais próxima
            // do gol da maquina é a linha 0
            "xPosition": float // Posição X da linha
            // (fixo para a haste)
            "playerCount": int // Quantidade de jogadores na
            // linha (deve ser maior ou igual a 0)
            "playerDistance": float // Distancia em Y entre dois
            // jogadores na linha (ignorado
            // caso exista apenas 1 jogador)
            "movementLimit": float // Distancia de movimentação
            // possível no eixo Y da haste
            // em relação à posição inicial
        },
        ...
    ],
    "cameraSettings": { // Informações da câmera integrada a mesa
        "framerate": int, // Em frames por segundo
        "resolution": [int, int] // comprimento e largura em pixels
    }
}

```

- StatusUpdateEvent: Evento enviado do cliente para o servidor para informar do estado atual da mesa, junto da imagem mais recente da camera.

```

{
    "eventType": "status_update",
    "timestamp": int,
    "version": int,
    "camera": {
        "image": String // bytes da imagem codificados
                    // em uma string por meio de um encoder base64
    },
    "lanes": [ // lista contendo o estado de cada linha
                // controlada pela maquina, cada elemento representa
                // uma linha
    {
        "laneID": int, // id da linha
        "currentPosition": float // posição Y atual da linha,
                                // em relação a posição inicial
        "rotation": float // rotação atual da linha, sendo 0º
                        // o boneco na posição inicial
    },
    ...
]
}

```

- ActionEvent: Evento enviado a partir do servidor para o cliente com o resultado do processamento da inteligência artificial.

```

{
    "eventType": "action",
    "timestamp": int,
    "version": int,
    "desiredState": [ // estado que o PUDMServer deseja para a mesa,
                    // cada elemento é um comando para uma linha
                    // não é obrigatório ter um elemento para
                    // cada linha
    {
        "laneID": int, // id da linha que este comando afeta
        "position": float, // posição para onde a linha deve se
                           // mover, em Y, sendo 0
                           // a posição inicial
        "kick": Boolean // informa se a linha deve iniciar
    }
}

```

```
// um comando de chute
},
...
]
}
```

D.3 Distribuição

O Servidor de Decisão necessita de compatibilidade com o sistema operacional Snappy Ubuntu Core (escolhido como parte da solução eletrônica do projeto), uma versão enxuta do Tradicional Ubuntu para desktop. Esta versão para Raspberry possui a qualidade de ser um sistema operacional leve e projetado especificamente para aplicações de sistemas embarcados e IoT (Internet of Things, internet das coisas) ([BERMEJO et al.](#),).

Para alcançar este objetivo, o software foi empacotado como um pacote Snap. Formato utilizado para a entrega de pacotes no sistema operacional citado. A partir de um Snap pode ser criada uma imagem de instalação especializada, que instalará o Ubuntu Core, o Servidor de Decisão e as dependências necessárias.

Os detalhes do empacotamento podem ser vistos no repositório de código do Servidor de Decisão.