UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA E MECÂNICA
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

PAULO CESAR MARTINS DA SILVA
PAULO HENRIQUE GARCIA DE SÁ
RODRIGO DE CARVALHO YWATA

MANIPULADOR PARALELO RUS COM CONTROLE DE POSIÇÃO PID

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2018

PAULO CESAR MARTINS DA SILVA
PAULO HENRIQUE GARCIA DE SÁ
RODRIGO DE CARVALHO YWATA

MANIPULADOR PARALELO RUS COM CONTROLE DE POSIÇÃO PID

Trabalho de Conclusão de Curso de Graduação, apresentado ao Curso Superior de Tecnologia em Mecatrônica Industrial, dos Departamentos Acadêmicos de Eletrônica e Mecânica, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Luís Paulo Laus

CURITIBA
06/2018

TERMO DE APROVAÇÃO

PAULO CESAR MARTINS DA SILVA
PAULO HENRIQUE GARCIA DE SÁ
RODRIGO DE CARVALHO YWATA

MANIPULADOR PARALELO RUS COM CONTROLE DE POSIÇÃO PID

Este trabalho de conclusão de curso foi apresentado no dia 09 de julho de 2018, como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial, outorgado pela Universidade Tecnológica Federal do Paraná. Os alunos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Milton Luiz Polli
Coordenador de Curso
Departamento Acadêmico de Mecânica

_____
Prof. M.Sc. Sérgio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____
Prof. Dr. Luiz Carlos de Abreu Rodrigues
UTFPR

_____
Prof. Dr. Gilson Yukio Sato
UTFPR

_____
Prof. Dr. Luís Paulo Laus
Orientador - UTFPR

"A Folha de Aprovação assinada encontra-se na Coordenação do Curso"

RESUMO

MARTINS, Paulo Cesar; SÁ, Paulo H. Garcia; YWATA, Rodrigo C. **Manipulador paralelo RUS com controle de posição PID**. 2018. 75 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Mecatrônica Industrial), Departamentos Acadêmicos de Eletrônica e Mecânica, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

Este trabalho apresenta o desenvolvimento de uma plataforma robótica com seis graus de liberdade controlada via *smartphone*. Seu objetivo é oferecer uma ferramenta para atividades práticas na área de mecatrônica complementares à teoria apresentada em sala de aula. O dispositivo consiste em um robô paralelo atuado por seis servomotores controlados por uma placa Arduino. A interface desenvolvida para *smartphones* com sistema operacional Android permite que qualquer aluno com este sistema possa interagir com a plataforma via *bluetooth*. A execução do projeto exigiu a integração multidisciplinar de conhecimentos em robótica, eletrônica, projeto de PCI, servo acionadores, programação de microcontroladores, entre outros. Esta característica é fundamental na mecatrônica. A experiência evidenciou que a união entre teoria e prática pode ser um diferencial no processo de aprendizagem. Todo conteúdo estudado foi posto em prática e testado, possibilitando melhor compreensão e assimilação do conhecimento.

Palavras chave: Arduino. Cinemática. Manipulador paralelo. Mecatrônica. Robótica.

ABSTRACT

MARTINS, Paulo Cesar; SÁ, Paulo H. Garcia; YWATA, Rodrigo C. **Parallel manipulator RUS with PID position control**. 2018. 75 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Mecatrônica Industrial), Departamentos Acadêmicos de Eletrônica e Mecânica, Universidade Tecnológica Federal do Paraná. Curitiba, 2018.

This work presents the development of a robotic platform with six degrees of freedom controlled via smartphone. Its objective is to offer a tool for practical activities in the area of mechatronics complementary to the theory presented in the classroom. The device consists of a parallel robot operated by six servomotors controlled by an Arduino board and the interface developed for smartphones with Android operational system that allows any student with this system to interact with the platform via Bluetooth. The execution of the project required the multidisciplinary integration of knowledge in robotics, electronics, PCB design, servo drives, programming of microcontrollers, among others. Note that this characteristic is fundamental in mechatronics. Experience has shown that the union between theory and practice can be a differential in the learning process. All subjects studied during the course have put into practice and tested, allowing better understanding and assimilation of knowledge.

Keywords: Arduino. Kinematics. Parallel manipulator. Mechatronics. Robotics

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Esta seção apresenta o problema motivador deste trabalho, sua contextualização e justificativa.

1.1 TECNOLOGIA

O avanço cada vez mais acelerado da tecnologia para suprir as necessidades de mercado e de consumo, sejam elas de métodos e sistemas mais eficientes de produção ou de novos produtos, tem trazido grandes desafios às empresas que investem em pesquisa e desenvolvimento. Em razão disto, há uma maior demanda por profissionais qualificados e aptos a responder de forma dinâmica e eficiente às constantes mudanças no setor de tecnologia. Essa nova realidade exige das instituições de ensino superior uma maior atenção aos métodos de ensino praticados, para que o aluno saia preparado para o mercado de trabalho.

O ensino de tecnologia, especialmente na área de mecatrônica industrial, mostra fundamental necessidade de agregar atividades práticas à teoria aplicada em sala de aula, proporcionando ao aluno um contato prévio com situações com as quais poderá se deparar no exercício da profissão. Além de facilitar a compreensão e assimilação do conteúdo apresentado. Porém, mesmo nas melhores universidades, ainda se encontram algumas situações deficitárias, onde conteúdos potencialmente atrelados às experiências práticas são apresentados sem este caráter prático, deixando de proporcionar ao estudante uma fase de fixação de conhecimentos essencial à formação de um conhecimento sólido e perene.

Com o objetivo de contribuir para o processo de ensino, este trabalho apresentará uma estrutura robótica microcontrolada que permitirá ao estudante visualizar e experimentar vários conceitos relacionados à Mecatrônica. Dentre eles a robótica, teoria de controle, microcontroladores, eletrônica, programação e servoacionadores. Essa integração entre diferentes áreas de conhecimento em um sistema mecatrônico proporciona, além da prática em si, uma visão multidisciplinar dos conteúdos relacionados à área, o que é um fundamento essencial da Mecatrônica e enriquece o aprendizado.

1.2 PROBLEMA

Alguns autores já discorreram sobre a importância da conciliação entre teoria e aplicações reais nas práticas de ensino, sendo que uma não pode ser entendida como independente da outra. É possível observar isso na afirmação “a teoria que não responde às demandas da prática requer ser revista” (MEDEIROS; CABRAL, 2006, p. 4).

A Universidade Tecnológica Federal do Paraná possui plantas para demonstração de sistemas de controle em malha fechada voltados para processos contínuos com controle de variáveis como: temperatura, vazão e pressão. Porém, não há plantas correspondentes para controle de variáveis mecânicas, como posição e velocidade, o que consideramos ser um importante complemento prático para disciplinas relacionadas à teoria de controle e robótica.

O sistema proposto, de certa forma, é diferente das aplicações vistas na disciplina de “controle de processos contínuos”, onde a planta já estava pronta e o controlador PID já estava desenvolvido, faltando apenas a sintonia dos parâmetros de controle. Este projeto envolve a implementação de um controlador PID de forma discretizada, além de equações de cinemática e integração com os periféricos do sistema.

1.3 JUSTIFICATIVA

A importância do controle nas áreas tecnológicas pode ser evidenciada em afirmações como:

> O controle automático é essencial em qualquer campo da engenharia e da ciência. O controle automático é um componente importante intrínseco em sistemas de veículos espaciais, sistemas robóticos, modernos sistemas de manufatura e quaisquer operações industriais que envolvam o controle de temperatura, pressão, umidade, viscosidade, vazão, etc. (OGATA, 2010, p.1).

Sendo a teoria de controle de extrema importância para aplicações diversas, é imprescindível que os alunos desenvolvam interesse pelo aprendizado e compreensão do assunto. Conforme citado anteriormente, a prática deve ser apresentada junto à teoria, visando complementar o conteúdo apresentado em sala de aula. O objetivo é proporcionar a prática através da reprodução de um sistema

similar ao “*Ball and Plate PID control with 6 DOF Stewart platform*” desenvolvido pelos estudantes Tyler Kroymann e Robert Dee da instituição San Jose State University, conforme Figura 1.

Espera-se que a maneira com que o sistema apresenta o funcionamento de um controlador PID e a integração deste com os atuadores, além da possibilidade de controlar variáveis mecânicas fora de simuladores, desperte o interesse por parte dos alunos naquilo que estão estudando.

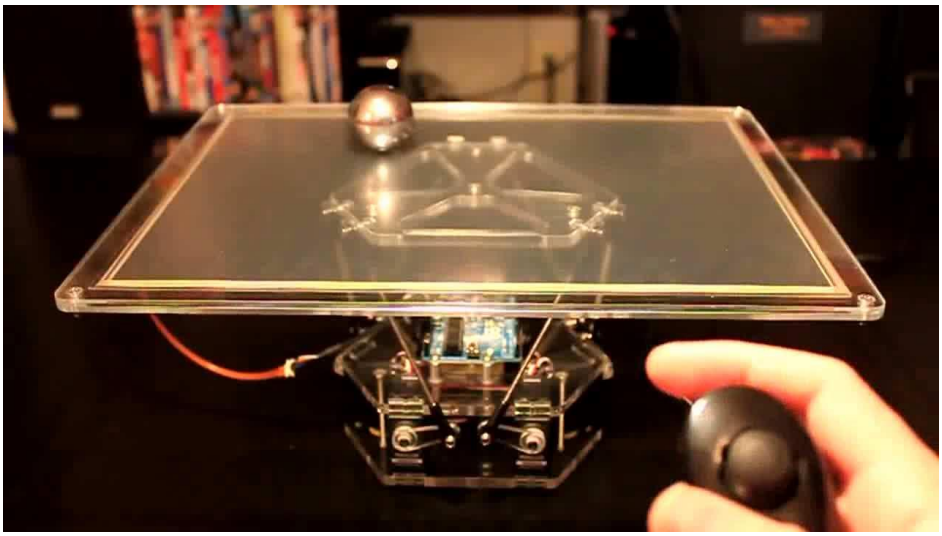


Figura 1 – Projeto base
Fonte: Full Motion Dynamics (2006, p. 54).

2 OBJETIVOS

Para execução do projeto foram definidos objetivo geral e objetivos específicos, buscando apresentar uma solução alternativa ao problema apresentado.

2.1 OBJETIVO GERAL

Implementar um sistema de controle PID em um Manipulador Robótico Paralelo tipo RUS de seis graus de liberdade com supervisão via *software* de computador.

2.2 OBJETIVOS ESPECÍFICOS

- Levantar e realizar o estudo dos conteúdos pertinentes às áreas de conhecimento envolvidas;
- Projetar e confeccionar componentes mecânicos que não são comerciais, como por exemplo, a base e a plataforma;
- Montar um manipulador paralelo RUS, a partir de componentes disponíveis no mercado;
- Escolher os componentes que não serão confeccionados, como motores, Arduino e sensor (tela *touchscreen*);
- Desenvolver *firmware* do sistema de controle PID utilizando a plataforma Arduino, a partir do estudo de sistemas já existentes;
- Elaborar sistema supervisório que seja capaz de controlar a plataforma;
- Projetar os circuitos eletrônicos responsáveis pelo fornecimento de tensão, filtragem e distribuição de sinais para os diversos elementos do sistema.

3 METODOLOGIA

O levantamento de dados pertinentes ao embasamento teórico do projeto foi realizado a partir da pesquisa em materiais como: livros, repositórios eletrônicos, dissertações, teses e artigos. Antes da aquisição de itens comerciais, foi realizada pesquisa e especificação técnica através materiais dos fabricantes, como catálogos e *datasheets*. Após a aquisição, eles foram testados e sua aplicabilidade foi validada. Em paralelo foram iniciados os processos de: projeto e confecção dos demais componentes e elaboração de *firmware* para teste e validação do conjunto.

4 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentada a base teórica utilizada para desenvolvimento de validação do projeto.

4.1 BREVE HISTÓRICO DA TEORIA DE CONTROLE

Norman S. Nise (2002) acredita que os primeiros sistemas com realimentação desenvolvidos por seres humanos foram implementados pelos gregos por volta de 300 a.C em um relógio de água criado por Ktesibios. O controle era feito através da medição do nível de água em um reservatório para garantir que a taxa de gotejamento do líquido fosse constante e então funcionar como um contador de tempo. Porém o autor enfatiza que os sistemas de controle realimentados são mais antigos que a própria humanidade afirmando que “Diversos sistemas de controle biológicos foram formados nos primeiros habitantes do nosso planeta”.

Os autores Ogata (2010) e Nise (2002) afirmaram que a partir da necessidade de controlar a velocidade de uma máquina a vapor, James Watt deu início ao que conhecemos como controle automático, com a construção de um regulador centrífugo, no século XVIII. Desde então vários trabalhos contribuíram para o desenvolvimento e aprimoramento das práticas de controle para chegarmos ao panorama atual, como por exemplo: Nicolas Minorsky, que contribuiu para a automatização na direção de navios, a partir da representação do sistema através de equações diferenciais; Hendrik Wade Bode e Harry Nyquist, no desenvolvimento da análise de amplificadores com retroação; Harold Hazen, introduzindo o termo servomecanismos para os sistemas de controle.

A teoria de controle foi inicialmente voltada aos estudos de métodos de resposta de frequência e lugar das raízes. O resultado obtido a partir da aplicação destes métodos era aceitável, porém, não completamente satisfatórios aos requisitos dos projetos. Apenas no fim da década de 50 estudos a respeito de teoria de controle começaram a ser direcionados para a obtenção de um dos muitos sistemas possíveis que seja ótimo para determinados pré-requisitos de projeto.

4.1.1 Aplicações Contemporâneas

Atualmente, é vasta a aplicação de sistemas de controles em áreas da indústria, com controle de variáveis de processos (temperatura, vazão, concentração química de determinado produto, nível, pressão, etc.) e mercado automobilístico com controles de tração, velocidade, temperatura de motor e até mesmo nos sistemas de pilotagem automática de veículos. Ainda na área industrial, é importante ressaltar o crescimento da utilização de robôs em linhas de montagem e confecção de componentes eletrônicos. Segundo Nise (2002), os sistemas de controle também possuem ampla aplicação em navegação, orientação e controle de mísseis, veículos espaciais, navios e aviões. Sobre ônibus espaciais, Nise explica um pouco sobre o funcionamento dos sistemas de orientação e posição:

> No espaço, o sistema de controle de voo gira os motores do sistema de manobra orbital (OMS – *orbital maneuvering system*) para uma posição que fornece um impulso na direção comandada para manobrar a nave. Na atmosfera terrestre, a nave é manobrada por comandos enviados do sistema de controle de voo às superfícies de controle, como por exemplo os elevons. (NISE, 2002, p.5).

Além da aplicação em orientação e posição, ônibus espaciais também possuem sistemas de controle complexos geradores de energia de célula de combustível que convertem oxigênio e hidrogênio em eletricidade e água, dando suporte a vida da tripulação a partir destas reações.

Para não limitar a aplicações que, de certa forma, estão um pouco distantes da maioria, é válido ressaltar que o controle está presente no dia a dia em sistemas de aquecimentos residenciais (comumente encontrado em países onde predominam temperaturas mais baixas) e aquecimento de água a gás, além de aplicações automobilísticas já citadas anteriormente (OGATA, 2010; NISE, 2002).

A área da medicina e saúde também vem impulsionando o crescimento de sistemas de controle. Um exemplo disso é a exigência no Brasil, desde 2012, da norma ABNT NBR IEC 60601-1-10 que possui o título “Equipamento eletromédico — Parte 1-10: Requisitos gerais para segurança básica e desempenho essencial — Norma colateral: Requisitos para o desenvolvimento de controladores fisiológicos em malha fechada”. Esta norma em questão foi elaborada para auxiliar durante do desenvolvimento de equipamentos médicos que utilizem sistemas de controles que

envolvem variáveis fisiológicas dos pacientes (temperatura, glicemia, hemodinâmica). A norma também se aplica a alguns tipos de controle que não serão detalhados neste trabalho: controle adaptativo, *fuzzy* e redes neurais lineares e não lineares.

Outra grande novidade um tanto quanto recente do conjunto robótica e controle é a utilização de robôs em centros cirúrgicos, onde foi possível que médicos pudessem operar remotamente seus pacientes através de *joysticks*. O sistema robótico de cirurgia mais conhecido é originalmente denominado *The da Vinci Surgical System*.

4.1.2 Malha Aberta

Quando o sinal de saída não exerce ação de controle sobre o sistema, ele é classificado como Sistema de Controle de Malha Aberta, em geral as ações de controle são diretamente em função do tempo. Ou seja, o sinal de saída (também chamado de variável) não será medido e tampouco comparado com a entrada (também chamada de referência). Como não existe comparação entre os sinais, a precisão depende de uma calibração realizada periodicamente. Máquinas de lavar roupas e torradeiras são exemplos de malha aberta, onde as primeiras apenas possuem controle do tempo das operações (lavar, enxaguar, secar) sem medirem se as roupas já estão de fato limpas e, as torradeiras em geral, além do tempo, possuem no máximo um termostato que desliga o aquecimento quando uma determinada temperatura é atingida, porém não tem controle sobre o estado do alimento, podendo este já estar até mesmo queimado. (OGATA, 2010; NISE, 2002).

4.1.3 Malha Fechada

Ogata (2010) e Nise (2002) destacaram que, em contrapartida ao sistema de malha aberta, o Sistema de Controle de Malha Fechada possui sua saída exercendo controle sobre o sistema. Esse sistema é frequentemente chamado de sistema com realimentação, uma vez que a realimentação é utilizada com a finalidade de diminuir o erro (diferença entre saída e o valor de referência).

4.1.4 Comparativo entre Malhas

Em resumo ao que salientaram Ogata (2010) e Nise (2002), a principal vantagem do sistema em malha fechada é que mesmo com perturbações externas ao sistema, o valor desejado é ajustado automaticamente e muitas vezes a supervisão contínua de um operador não é necessária.

Porém, quando o sistema é isento de distúrbios e a entrada é previamente conhecida, recomenda-se a malha aberta, devido às questões de menor custo e complexidade para implantação e manutenção da planta, em virtude do número de componentes e potência que são necessários para o bom funcionamento de um sistema em malha fechada. Um bom desempenho, visando o sistema como um todo, também pode ser obtido a partir da utilização o de uma combinação entre o controle de malha aberta e de malha fechada.

4.1.5 Ação Proporcional

Bega (2003) descreve que a Ação Proporcional é caracterizada por fornecer uma correção proporcional ao sinal de erro (desvio). Esta apresenta velocidade de resposta rápida, porém não eficiente em regime estacionário, uma vez que não possui capacidade de zerar o erro residual (*offset*). O erro diminui conforme o ganho proporcional aumenta. Logo, é correto afirmar que o ganho proporcional deve ser ajustado no maior valor permitido pelo processo. Em contrapartida, um ganho excessivo poderá ocasionar instabilidade no sistema.

4.1.6 Ação Integral

Bega (2003) afirma que ao contrário da ação proporcional, a ação integral é eficiente em eliminar o erro residual variando lentamente e proporcionalmente à integral do sinal de erro em função do tempo. Enquanto houver diferença entre os valores medido e desejado, a ação integral estará tentando eliminar o erro de *offset*. Devido ao fato da ação integral agir de forma lenta, perturbações rápidas no sistema

não serão corrigidas. A correção deste tipo de ação só ocorre quando o sentido do desvio for alterado, caso contrário o sistema poderá tornar-se instável.

4.1.7 Ação Derivativa

Diferente das duas ações anteriores, a ação derivativa não reage de acordo com a amplitude do desvio, e sim de acordo com a velocidade deste. Ela tem como principal característica a tentativa de antecipação da correção em relação ao desvio, melhorando o tempo de atuação. A instabilidade para esse tipo de sistema ocorre porque a ação puramente derivativa induz movimentos rápidos no elemento final de controle, já que não leva em consideração a amplitude do desvio. Sendo assim, a ação derivativa não deve ser utilizada sozinha, sendo ela acoplada ao controlador proporcional ou Proporcional Integral Derivativo (PID) (BEGA, 2003).

4.1.8 Ação Proporcional Integral Derivativa

As ações de controle, quando combinadas, caracterizam o controlador Proporcional Integral Derivativo (PID), e tendem a combinar as vantagens de cada ação de controle (velocidade, antecipação e eliminação de erro) em um único controlador. Conforme citado a seguir:

> Em sistemas de controle em malha fechada, a ação proporcional elimina as oscilações, a integral elimina o *offset*, enquanto a derivativa fornece ao sistema uma ação antecipativa, evitando previamente que o desvio se torne maior quando o processo se caracteriza por ter uma correção lenta, comparada com a velocidade do desvio, como ocorre em alguns sistemas de controle de temperatura (AMOROSO, 2013).

4.2 DISPOSITIVO DE CONTROLE: ARDUINO

O Arduino consiste em uma placa de desenvolvimento e prototipagem eletrônica, que utiliza os microcontroladores Atmel AVR. Desenvolvida na Itália, busca tornar a eletrônica mais acessível aos usuários e interessados pelo assunto. Possui hardware e softwares livres, ou seja, quem possui capacidade e interesse pode

fabricar uma placa similar sem que seja necessário pagar direitos autorais aos desenvolvedores. Apresenta modelos com diferentes tipos de microcontroladores, sendo capazes de auxiliar no desenvolvimento eletrônico desde projetos simples de cunho educacional até mesmo projetos complexos que envolvem robótica (SOARES, 2013).

4.2.1 Arduino Mega 2560

O Mega 2560 é uma placa Arduino que possui todos os circuitos necessários para o suporte e funcionamento do microcontrolador ATmega 2560. Nesta placa são encontradas: 54 entradas/saídas digitais (sendo que 15 destas podem ser usadas como saídas de sinal de modulação por largura de pulso, mais conhecida por PWM), 16 entradas analógicas, quatro portas seriais, um cristal de frequência 16 MHz, conexão USB, plugue de alimentação, conexão ICSP, e um botão *reset* (ARDUINO, 2015).

4.3 MANIPULADOR: ROBÔ PARALELO

Gonçalves (2009) descreve uma arquitetura paralela como uma cadeia cinemática fechada com seguimentos, ou conjuntos de seguimentos articulados, que unem simultaneamente a base ao elemento terminal. Há várias formas construtivas onde os acionadores podem ser instalados, na base fixa ou perto da base, tornando-os mais leves. A existência de várias cadeias cinemáticas, de complexidades variadas entre base e elemento terminal, caracterizam uma arquitetura paralela. Já para Merlet (1997) “um robô paralelo é composto de um efetuador com *n* graus de liberdade e de uma base fixa, ligados entre si por, pelo menos, duas cadeias cinemáticas independentes. O acionamento ocorre por meio de *n* atuadores simples”.

4.3.1 Graus de Liberdade

Sistemas mecânicos que possibilitam que um corpo rígido (efetuador) se mova em relação à uma base fixa desempenham importante papel em inúmeras aplicações. A capacidade de mover-se no espaço, de várias maneiras, através de translação ou rotação determinam os graus de liberdade do efetuador (gdl). O número máximo de graus de liberdade para um corpo rígido é seis, o que equivale a três movimentos de translação ao longo de eixos mutuamente ortogonais e três rotações em torno destes eixos. A posição e orientação de um efetuador pode ser descrita através de coordenadas generalizadas de um de seus pontos e dos ângulos que definem sua orientação, ou através de qualquer conjunto de parâmetros que permitam definir com exclusividade sua posição final. Isso permite que se controle vários graus de liberdade através de um sistema mecânico. Logo, este sistema pode ser definido como um robô (MERLET, 1997).

4.3.2 Plataforma de Stewart

A plataforma de Stewart é uma estrutura cinemática paralela capaz de executar movimentos em seis graus de liberdade, tem várias aplicações em manufatura, simuladores de movimento, tarefas de precisão, entre outras. O mecanismo consiste em uma base fixa conectada a uma plataforma móvel através de seis segmentos, formados por duas juntas posicionadas nos extremos do atuador (ACUÑA, 2009). Desde a publicação do trabalho de Stewart (1965) várias estruturas paralelas foram propostas, dentre elas o manipulador paralelo *Hexa*, uma estrutura robótica de arquitetura paralela com seis graus de liberdade capaz de executar movimentos com alta velocidade e precisão (MERLET, 2006). Há várias arquiteturas possíveis o robô *Hexa*. A Figura 2 apresenta uma arquitetura semelhante a que será utilizada neste trabalho.

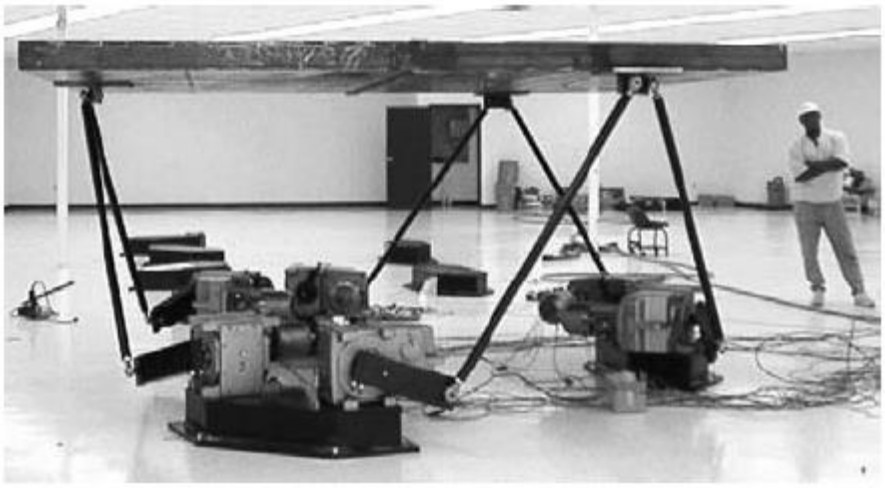


Figura 2 – Exemplo de manipulador paralelo com 6 gdl tipo RUS
Fonte: Merlet (2006, p. 54).

Esta arquitetura é do tipo RUS com seis gdl, da notação literal, significa que possui juntas rotativa (R), universal (U) e esférica (S), respectivamente, em sua cadeia cinemática, partindo da base ao elemento terminal. Assim como todas as estruturas paralelas em geral, ela apresenta grandes vantagens sobre outros tipos construtivos em velocidade de operação, rigidez, precisão e capacidade de carga. Embora tenha um espaço de trabalho menor que arquiteturas seriais, a possibilidade de ter sua estrutura instalada sobre o posto de trabalho é vantajosa por não ocupar o espaço no chão de fábrica (GONÇALVES, 2009). Aliada às vantagens naturais relacionadas ao tipo de estrutura, a utilização de servomotores como atuadores traz um ganho ainda maior ao sistema, devido ao alto grau de precisão e velocidade que proporcionam, além de exigir um sistema de controle mais simples (MELLO, 2011).

Neste trabalho, a diferença para o modelo supracitado refere-se às juntas utilizadas nas cadeias cinemáticas. A junta universal foi substituída por uma junta esférica, o que mudaria a nomenclatura de RUS para RSS. Porém, devido à larga utilização da sigla RUS por estudantes e curiosos sobre o assunto, independentemente da junta utilizada, optou-se por mantê-la no título do trabalho. O efeito da substituição da junta é o aumento de um gdl para cada uma das cadeias cinemáticas do manipulador, que possibilita a rotação das hastes que ligam a plataforma às hastes dos motores, em torno do próprio eixo. Por outro lado, na prática, este movimento é anulado pelos demais, não tendo nenhuma influência na movimentação da plataforma.

4.3.3 Cinemática

Utilizada para descrever as relações entre o movimento das juntas de um robô e dos segmentos que o compõem, a cinemática cria um mapeamento entre as variáveis de junta e a posição e orientação desejadas. Ela se divide em dois tipos, cinemática direta, que permite calcular posição e orientação conhecendo-se as variáveis de junta, e a cinemática inversa, com a qual se calcula o valor das variáveis de junta a partir da posição e orientação desejadas. Neste trabalho, por se tratar de um robô paralelo, a cinemática inversa se mostra mais vantajosa e eficaz na definição dos valores das variáveis de junta, já que oferece uma solução única para uma dada posição e orientação desejada. Além de ter um equacionamento relativamente simples do ponto de vista computacional. Já a cinemática direta se mostra mais complexa na definição das variáveis e também leva a múltiplas soluções, o que torna o processamento mais moroso e torna inviável uma solução algébrica fechada para o problema (LAUS, 2017).

4.4 ACIONADORES: SERVOMOTORES

Foram escolhidos como atuadores para plataforma servomotores utilizados em aeromodelismo, devido às suas vantagens em relação à disponibilidade, custo, torque e precisão de movimento.

4.4.1 Servomotor

Segundo Kosow (1993), os servos motores de corrente contínua (CC) são acionados por corrente proveniente de amplificadores eletrônicos ou amplificadores CA, reatores saturáveis, amplificadores retificadores. As principais características de servomotores, sejam CA ou CC, são o torque de saída que apresenta uma certa proporcionalidade à tensão de alimentação, e sentido do torque, definido pela polaridade da tensão de controle. Os servos motores são utilizados quando se deseja movimentar um objeto e ter precisão no posicionamento. Eles têm a capacidade de

realizar um movimento e manter-se nesta posição mesmo sofrendo algum tipo de esforço contrário.

Um servomotor é composto pelos componentes listados a seguir e apresentados na Figura 3:

- circuito de controle: responsável por monitorar o potenciômetro e acionar o motor;
- potenciômetro: ligado ao eixo de saída, monitora a sua posição;
- motor: movimenta a engrenagem;
- engrenagem: responsável pela redução da rotação do motor, obtendo assim um maior torque no eixo de saída;
- caixa servo: caixa para acomodar e posicionar as peças internas conforme Figura 3.

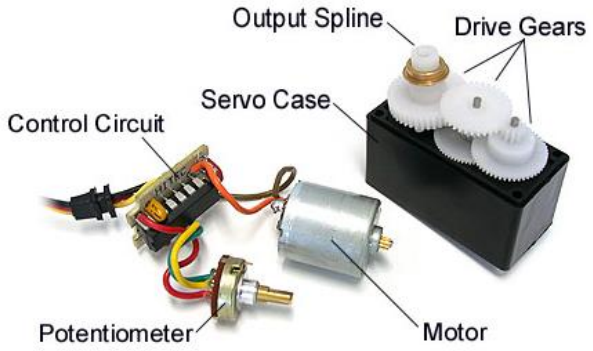


Figura 3 – Componentes do servo motor
Fonte: SERVOCITY(2018).

4.4.2 Controle do servomotor

O servomotor é um mecanismo eletromecânico que trabalha em sistema de controle de malha fechada, recebe um sinal do controle, compara com sua posição atual e realiza o movimento. Os servos motores possuem três fios, sendo dois de alimentação e um para controle. Esse controle é feito através de um sinal PWM (*Pulse With Modulation*), que segundo Rashid (1999) é obtido através de chaveamento (liga/desliga) durante um semiciclo, controlando a tensão através da largura do pulso. O sinal recebido para controle do servo é de 0V ou 5V. O circuito de controle monitora este sinal em intervalos de tempo de 20 ms. Assim que há uma alteração na largura do sinal, ele altera a posição do potenciômetro para que ele coincida com o sinal recebido.

A Figura 4 ilustra o sinal PWM com larguras de pulso diferentes e as respectivas posições do servomotor. O circuito de controle recebe um sinal de 1ms, por exemplo, que comparado ao valor do potenciômetro gera um sinal para o servomotor, que por sua vez altera a posição do eixo de acordo com o sinal recebido.

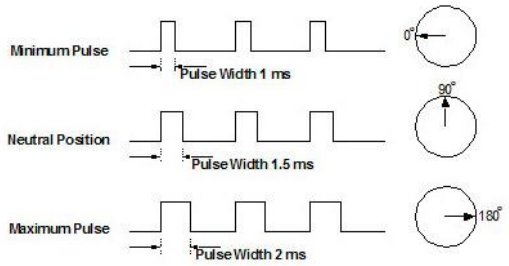


Figura 4 – Sinais de controle por PWM
Fonte: SERVOCITY (2018).

4.5 SENSOR: TELA TOUCH SCREEN

A idealização do primeiro dispositivo *touch screen* ocorreu em 1965 por E.A Johnson no Royal Radar Establishment em Malvern, Reino Unido. Originalmente o trabalho foi descrito em um artigo publicado na Electronics Letters, com o título "*Touch display – a novel input/output device for computer*". Nele é descrito um tipo de mecanismo hoje conhecido como toque capacitivo, atualmente usado em muitos *smartphones*. Já na década de 70 foi desenvolvido, por G. Samuel Hurst, o *touchscreen* de tipo resistivo ofuscando a ideia inicial por toque capacitivo. As primeiras tentativas de patente não tiveram sucesso pela aparente falta de aplicabilidade fora do ambiente de laboratório. Hust e um grupo de amigos, autointitulados "Elographics", persistiram na ideia trabalhando de forma independente desenvolvendo o projeto inicial, que contribuiu para o surgimento do que hoje é conhecida como tecnologia de toque resistivo (ION, 2013).

Ao longo dos anos, foram desenvolvidos diferentes tipos de *touchscreens*. Neste trabalho, será utilizada tela de toque resistivo. Este tipo foi escolhido por apresentar, como afirma Walker (2014), vantagens como baixo custo, baixo consumo de energia, resistência à contaminação externa, ampla disponibilidade de mercado, estrutura fina e leve, além de funcionar somente com o dedo ou objetos não afiados. A tela resistiva de 5 fios (Figura 5) consiste em duas camadas separadas por

pequenos pontos isolantes. A base se constitui de um substrato de vidro revestido por material resistivo, a superfície de toque é uma membrana revestida por material condutivo. A partir da flexão da membrana, decorrente do toque, acontece o contato elétrico entre os revestimentos resistivo e condutivo. A representação da coordenada do ponto de toque é obtida a partir da leitura da diferença de tensão equivalente à posição nos eixos X e Y (ELOTOUCH, 2015).

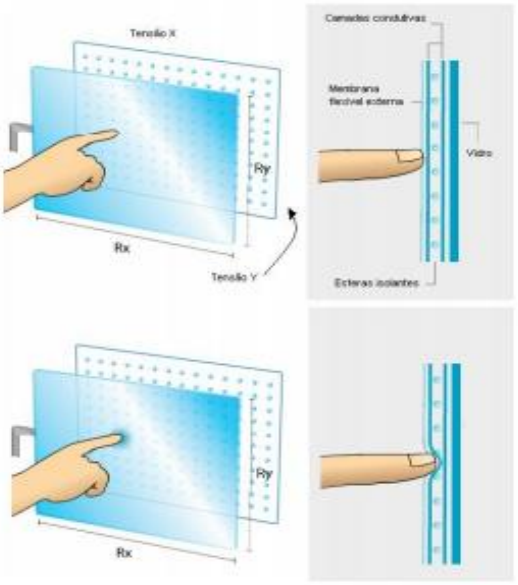


Figura 5 – Tela resistiva 5 fios.
Fonte: Elotouch (2015).

4.6 SISTEMA DE SUPERVISÃO

O sistema de medição e controle da plataforma, bem como a interface ao usuário, será desenvolvido através do *software* LabVIEW *da* National Instruments. É uma plataforma de programação gráfica que permite a integração entre software, IP e hardware através de ferramentas que proporcionam maior rapidez e eficiência na resolução de problemas. A escolha do *software* se deu pelo fato de que possui alto grau de integração com *hardwares* de aquisição de dados, bibliotecas de processamento de sinais e controles de interface ao usuário desenvolvidos para visualização dos dados de medição, que permitem o desenvolvimento de uma grande gama de atividades de forma bastante versátil e mais simplificada que outros *softwares* (NATIONAL INSTRUMENTS, 2015).

O sistema supervisório que foi desenvolvido para este projeto consistirá em uma interface gráfica, executada em um PC comum, que permite ao usuário visualizar valores de variáveis e gráficos correspondentes. Há também a possibilidade de definir determinados parâmetros e acionar funções pré-configuradas na plataforma. Inicialmente, o sistema deverá ter as seguintes funcionalidades:

- mostrar na tela o deslocamento linear e angular da plataforma em cada eixo;
- gráfico que mostre a posição de determinado corpo que esteja sobre a tela *touch screen* em tempo real;
- listas de seleção de funcionalidades da plataforma;
- opção para entrada de valores de posicionamento de juntas.

4.7 MIT APP INVENTOR

Desenvolvido no MIT (Instituto de Tecnologia de Massachussetts), o App Inventor é uma plataforma visual de programação para desenvolvimento de softwares para uso em *smartphones* com sistema operacional Android. Sendo de uso gratuito, possui bibliotecas já integradas e uma programação baseada em blocos muito intuitiva, o que possibilita em pouco tempo a criação de aplicativos com diversas funcionalidades. Pode ser utilizado via navegador *web* e possui dois ambientes de programação, designer (Figura 6) onde é possível editar o layout da tela e seleção de funcionalidades nativas do Android e o ambiente *blocks* (Figura 7) onde a lógica de funcionamento do *software* é elaborada.

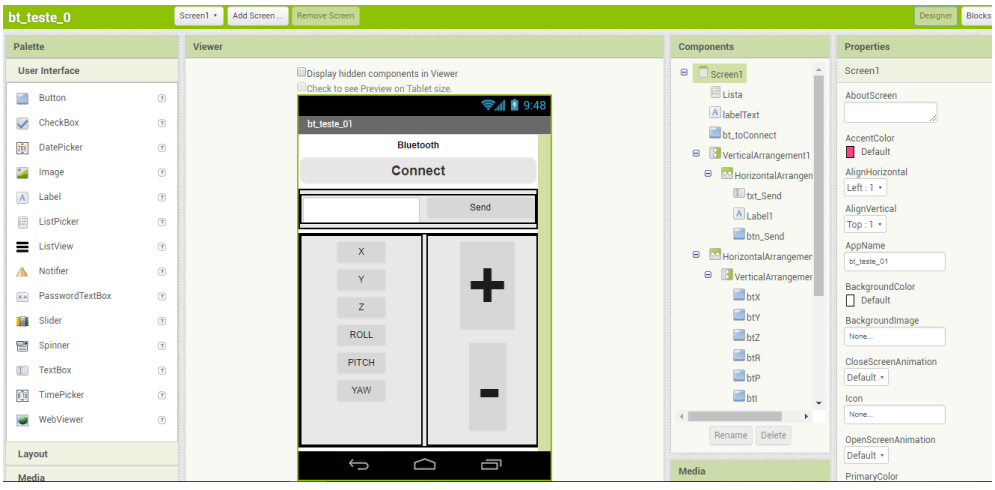


Figura 6 – Ambiente designer MIT APP Inventor.
Fonte: Autoria própria

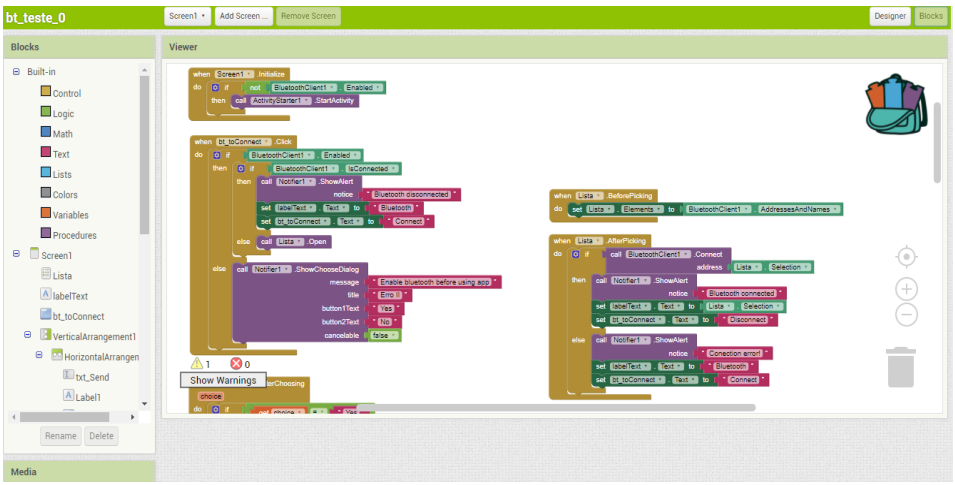


Figura 7 – Ambiente Block MIT APP INVENTOR
Fonte: Autoria própria.

4.8 COMUNICAÇÃO VIA BLUETOOTH®

Optou-se por não desenvolver um novo protocolo de verificação de integridade de comunicação entre o módulo HC-05 e o *smartphone* utilizado. Pois os *chips* que utilizam o sistema *bluetooth*® já possuem em sua arquitetura camadas responsáveis pela qualidade dos dados a serem transmitidos interna e externamente (Figura 8). Nestas camadas, protocolos de comunicação como: Radiofrequência (RF), Controle de Enlace de Dados (LC), Gerenciamento de Enlace de Dados (LM) e Controle e Adaptação Lógico de Dados (L2CAP) já foram implementados. (SIG, 2004).

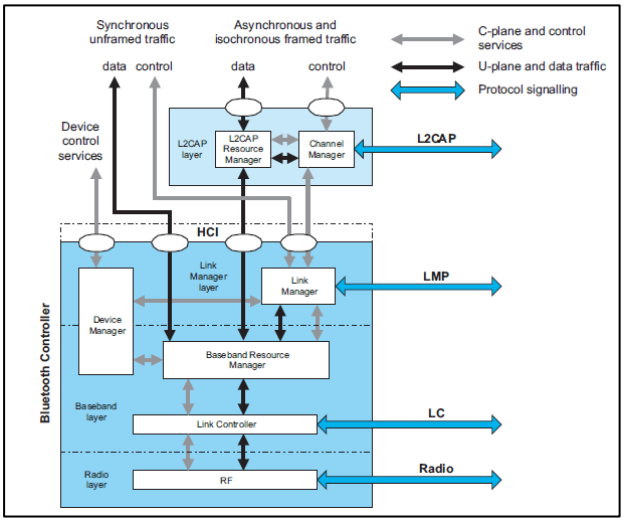


Figura 8 – Arquitetura interna *bluetooth* core.
Fonte: BLUETOOTH SPECIAL INTEREST GROUP, 2004.

A transferência de informações entre as camadas é feita de tal maneira que existem algoritmos responsáveis por analisar os pacotes de dados e caso sejam detectados erros, o próprio sistema se encarrega de eliminá-los. Durante estas verificações, uma das maneiras é a utilização do controle de dados com ARQ (*Automatic Repeat Request*), onde o receptor solicita a retransmissão da mensagem caso a leitura não tenha sido realizada conforme esperado. O emissor reenvia os dados e só enviará um novo pacote quando receber autorização por parte do receptor (SIG, 2004).

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Os detalhes da execução do projeto, problemas encontrados, ajustes e testes realizados serão apresentados ao longo desta seção.

4.9 ALTERAÇÃO DE ESCOPO

Durante o desenvolvimento do projeto surgiram dificuldades técnicas não previstas inicialmente que comprometeram a conclusão de alguns objetivos. A principal dificuldade foi encontrar um modelo cinemático já desenvolvido e pronto para ser implementado, já que o objetivo principal do projeto era elaborar um sistema de controle PID para plataforma. A consequência disto, dada a complexidade da implementação da robótica, foi a decisão de não fechar a malha do sistema com a tela *touchscreen* e trabalhar com o sistema em malha aberta, sem retroalimentação. Desta forma, o foco do projeto acabou sendo a robótica, em detrimento ao controle PID.

Optou-se também por alterar a plataforma utilizada para o desenvolvimento do sistema supervisório. Inicialmente foi utilizado o *software* LabView, porém, foram verificadas algumas desvantagens com relação a ele. A necessidade de licença de uso foi um fator bastante limitante, pois é acessível sem custo apenas em versão estudante, que, além de oferecer menos recursos que a versão comercial, limita sua utilização ao computador onde ele está instalado. Outra dificuldade foi a integração entre o Labview e o Arduino. Apesar de possuir bibliotecas específicas para o microcontrolador houve bastante ocorrência de erros de comunicação, que normalmente exigiam a reinicialização do sistema. A maior complexidade para programação em relação à plataforma escolhida, MIT App Inventor, também foi determinante nesta alteração.

Finalmente, com a adoção de uma interface ao usuário via aplicativo de celular, foi necessário adotar uma estrutura de comunicação que fosse acessível aos aparelhos. Foi adotada então a comunicação via *bluetooth*®. Para o Arduino foi utilizado o módulo *Bluetooth* HC-05, que já possui algumas aplicações para o microcontrolador.

4.10 ATUDORES: PROTEÇÃO E CONTROLE

O maior problema encontrado na utilização dos servos motores foi a baixa qualidade de fabricação, visto que, por questões de custo, foram todos importados da China. Durante os primeiros testes ocorreu a queima de componentes eletrônicos dos motores em diferentes rotinas. Após análise, constatou-se que, mesmo sem carga e longe dos limites de operação, ocorria travamento e/ou queima. Para diminuir o risco de dano, foi inserida uma proteção térmica (fusíveis autorresetáveis tipo PTC) na linha de alimentação dos motores. O modelo de fusível escolhido foi o “PFRA.090” que inicia o processo de interrupção do circuito interno em 0,9A, evitando a queima dos motores que trabalham com corrente nominal de no máximo 1,4A.

A necessidade de produção de uma placa para inserção das proteções térmicas, viabilizou também a colocação de proteção para supressão de surtos nas linhas de alimentação e comunicação serial. Para as linhas de alimentação foi escolhido o diodo modelo “P6KE12A” (tensão de acionamento em 12,6V, sendo 12V a tensão do circuito) e para comunicação serial o modelo “SMBJ6.5A” (tensão de acionamento em 6,5V, sendo 5V e 3,3V as tensões dos circuitos de comunicação). Vale ressaltar que uma pessoa pode facilmente acumular uma carga de mais de 10kV, e se não houver proteção nos circuitos eletrônicos, eles podem ser danificados com apenas um toque. Os diodos foram doados por uma empresa de montagem de placas eletrônicas, uma vez que para as tensões de trabalho não foram encontrados modelos disponíveis no mercado brasileiro com prazo de entrega viável.

A falta de qualidade dos motores MG995 ficou evidenciada também pela baixa repetibilidade dos movimentos em relação a largura de pulso aplicada para posicionamento do motor. Definida uma posição angular de referência, ao movimentar o motor para outra posição e em seguida enviá-lo novamente para posição de referência (largura de pulso), fisicamente houve variação na posição do eixo do motor. Essa diferença foi maior sempre que havia mudança no sentido do movimento, horário ou anti-horário.

Medidas realizadas apontaram divergências consideráveis entre os valores referenciados no *datasheet* do MG995. Para posicionar o motor nos ângulos de 0, 90 e 180 graus, as larguras de pulso esperadas eram de 1, 1.5 e 2 ms respectivamente. Porém, as Figuras 9, 10 e 11 mostram que a variação real era de 0,5 a 2,4 ms. Para melhorar a resolução dos movimentos dentro do espaço de trabalho, optou-se por não

utilizar a biblioteca para controle de servomotores, disponível no site oficial do Arduino. Os movimentos foram realizados alterando a largura de pulso através da manipulação direta dos registradores do microcontrolador ATMEGA2560, presente na placa Arduino utilizada neste projeto.

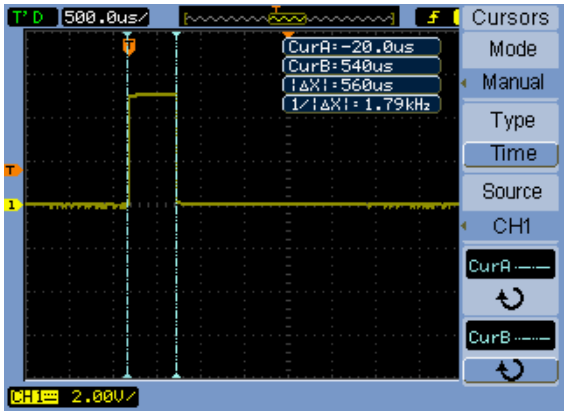


Figura 9 – Largura de pulso (0,54 ms) para 0º
Fonte: Autoria própria.

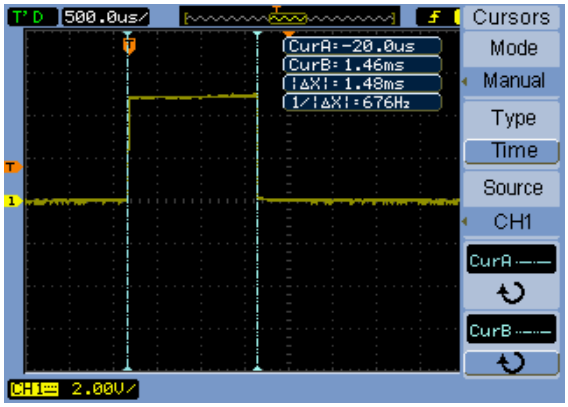


Figura 10 – Largura de pulso (1,48 ms) para 90º
Fonte: Autoria própria.

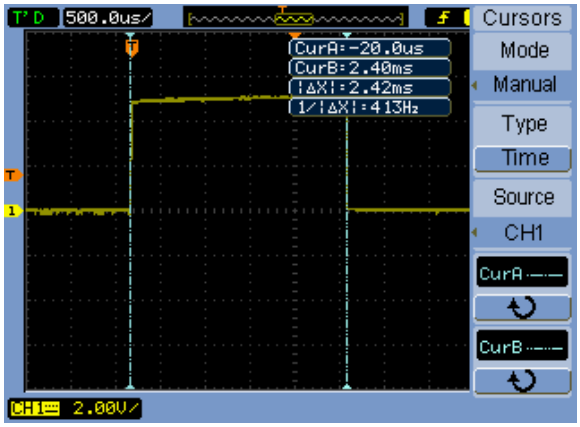


Figura 11 – Largura de pulso (2,42 ms) para 180º
Fonte: Autoria própria.

4.11 PLACA DE CIRCUITO IMPRESSO

A PCB foi projetada no *software* Cad Eagle, gratuito para aplicações não comerciais. Para facilitar a manutenção e utilização de futuros alunos interessados pelo projeto, a placa terá alguns barramentos de pinos compatíveis como a placa Arduino Mega 2560 e também com saídas disponíveis para expansão para outras aplicações. Primeiramente foi elaborado um esquemático e o *layout* da placa, conforme Figura 13 – Esquemático da placa no software Eagle. Figura 12, respectivamente. Neste esquemático é importante ressaltar:

1. F1 a F6 - fusíveis autorresetáveis citados na Seção 3.2 para proteção individual dos motores;
2. D1 a D5 - diodos de supressão de surtos, também conhecidos como diodos TVS ou diodos avalanche, para proteção contra surtos de energia;
3. SV14 – barramento de pinos para conexão com módulo *bluetooth* HC-05;
4. SV15 – barramento de pinos para selecionar qual das quatro UARTS irá se comunicar com o módulo *bluetooth*;

Com auxílio do *software* SolidWorks para projetos mecânicos, foi possível gerar uma visualização 3D do projeto da placa, conforme Figura 14.

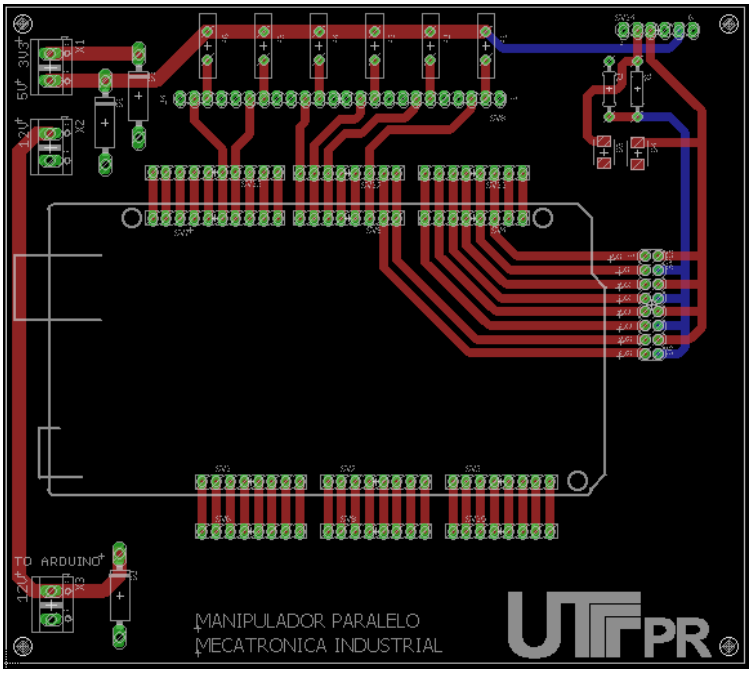


Figura 12 – Layout da placa de circuito impresso.
Fonte: autoria própria.

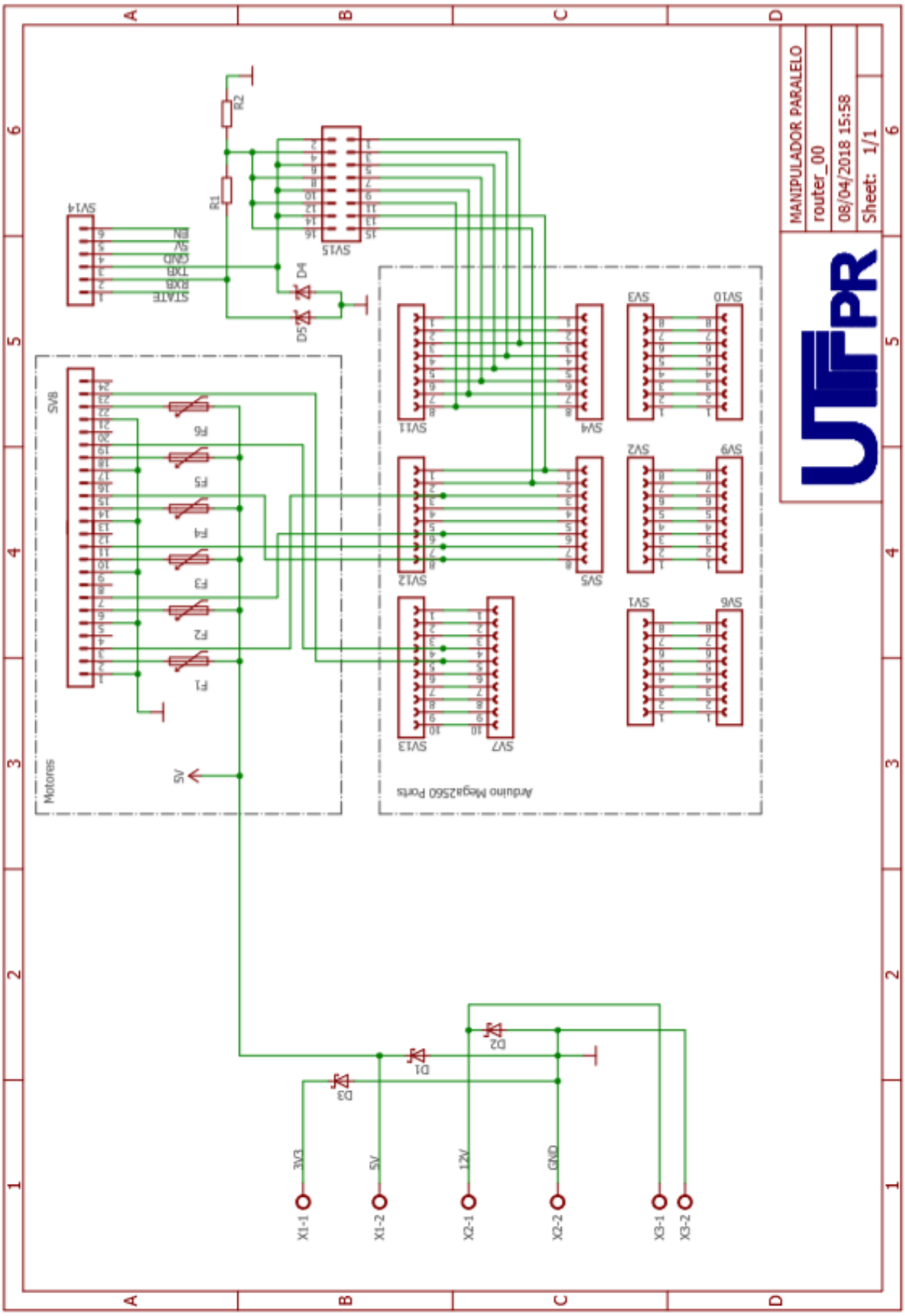


Figura 13 – Esquemático da placa no *software* Eagle.
Fonte: Autoria própria.

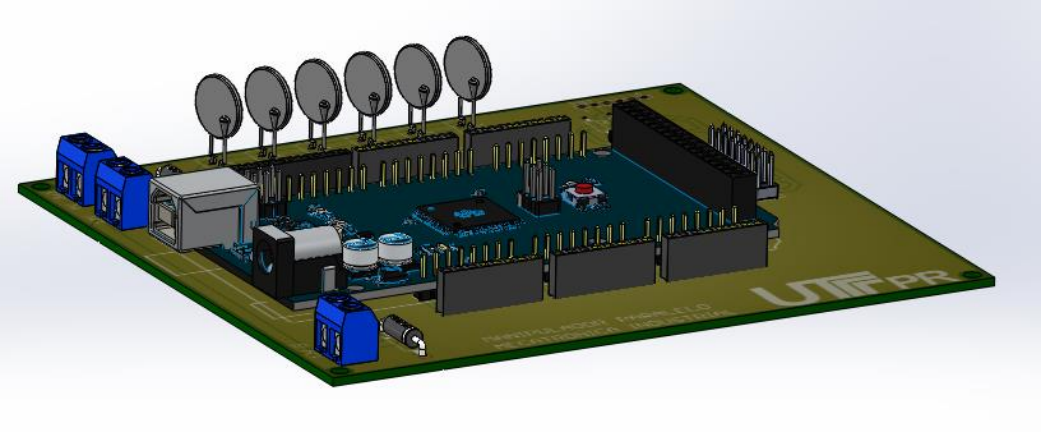


Figura 14 – Pré-visualização 3D da placa de circuito impresso.
Fonte: Autoria própria

4.11.1 Falha de fabricação.

As placas foram fabricadas por uma empresa especializada, mesmo assim houve falhas durante o processo de fabricação. Uma delas foi um curto circuito entre uma das pistas da comunicação serial. Mas o que gerou maior dificuldade foi fabricação da placa com todos os furos iguais e com diâmetro muito inferior ao especificado (Figura 15). Como o prazo para fabricação de uma nova placa seria de aproximadamente 20 dias, a solução foi ajustar a furação manualmente, ao fazer isso a metalização de todos os furos se rompeu e todas as ligações elétricas tiveram que ser refeitas com *jumpers* (Figura 16).

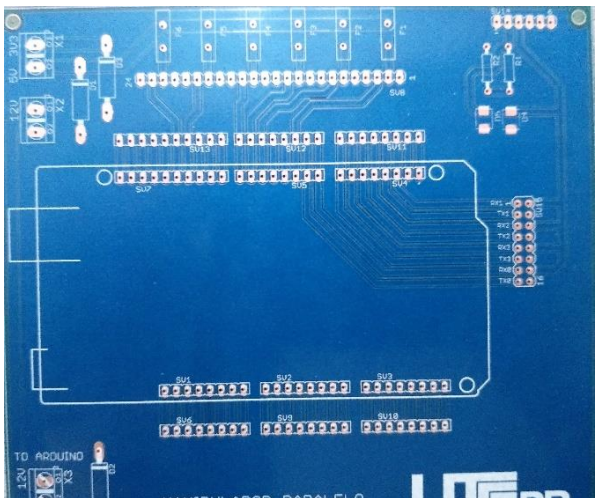


Figura 15 – Placa de circuito impresso com furos de diâmetros menores.
Fonte: Autoria própria.

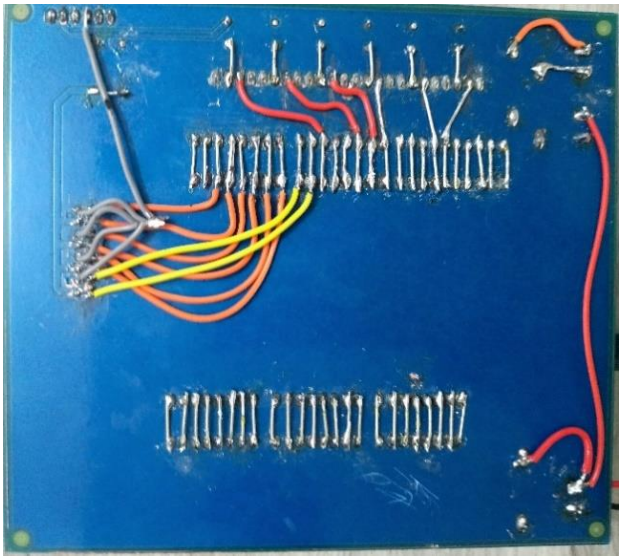


Figura 16 – Retrabalho para ligações elétricas.
Fonte: Autoria própria.

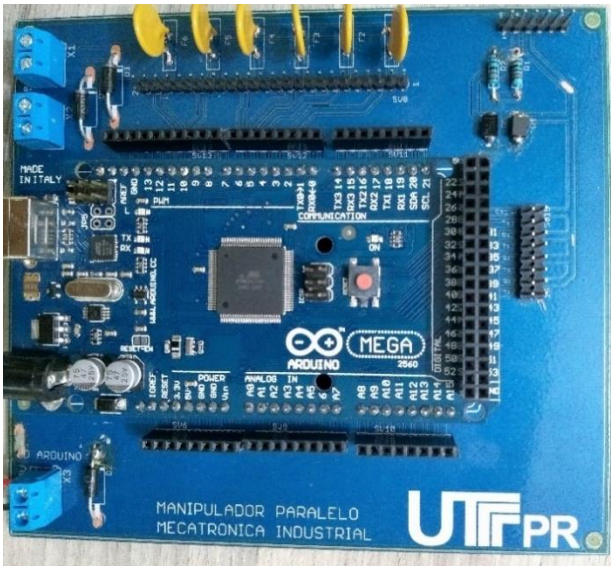


Figura 17 – Montagem final da placa de circuito impresso.
Fonte: Autoria própria.

4.12 FONTE DE ALIMENTAÇÃO

Para suprir a demanda de alimentação de todo circuito (motores, placa Arduino, módulo *bluetooth*) escolhemos uma fonte ATX de 230 watts de potência utilizada em computadores, modelo “Blu 230”. Durante os testes, notou-se que esporadicamente a fonte entrava em modo de segurança e as saídas de tensão eram zeradas.

A primeira análise foi com relação ao consumo de corrente do sistema em relação a fonte. Medições mostraram que a fonte estava sem sobrecarga, uma vez que todo sistema estava consumindo entre 1,0 e 3,0 amperes (sendo que aproximadamente 2,4 amperes são da saída 5,0 VDC que supre os motores) e a fonte suporta até 14 amperes nesta saída, conforme Figura 18, retirada do *datasheet* disponível no site do fabricante.

ENTRADA	TENSÃO (AC)				CORRENTE		FREQUÊNCIA		
	115/230V				5/3A		50/60 Hz		
SAÍDA DC	+3.3V	+5V	+12V$_1$	+12V$_2$	-12V	+5VSB	PS-ON	POK	COM
CORRENTE MÁX.	16A	14A	16A	16A	0.3A	2.0A			
POT. COMB.	86W		144W				REMOTE	P.G	RETURN
POT. COMB. OPERACIONAL	230W								

Figura 18 – Capacidade da fonte.
Fonte: Datasheet da fonte.

Em segunda análise, verificou-se que o acionamento dos motores gerava ruído intenso na tensão alimentação. A Figura 19 mostra que durante a movimentação da plataforma os níveis de tensão que deveriam ser de 5 VDC chegaram a 3,66 e 6,22 VDC, além de um ruído em uma frequência de aproximadamente 50Hz.

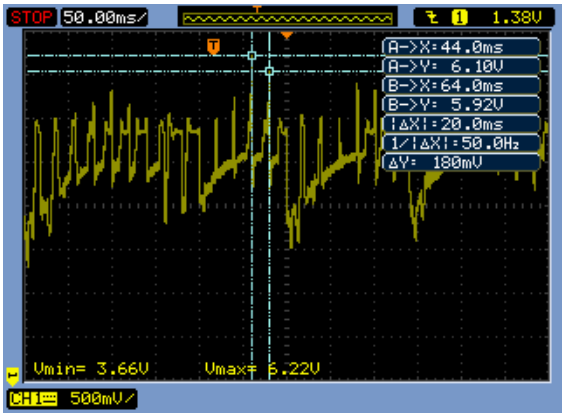


Figura 19 – Medição na alimentação dos servomotores durante movimentação.
Fonte: Autoria própria.

Para tentar minimizar este ruído, foram feitos alguns ajustes listados a seguir. É importante ressaltar que os testes foram feitos acionando apenas um motor, devido aos componentes de “baixa” potência disponíveis durante os testes e ligação deles em paralelo com o motor. Os testes realizados não tiveram sucesso efetivo.

1. Teste com acionamento do motor sem ajuste nenhum (Figura 20).

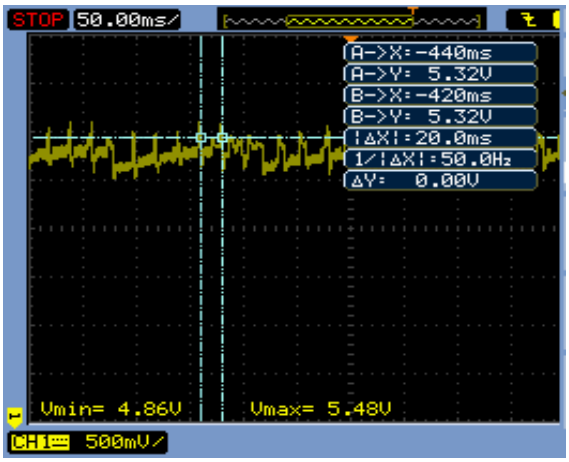


Figura 20 – Teste 1.
Fonte: Autoria própria.

2. Teste com acionamento do motor e capacitor de poliéster de 100nF (Figura 21).

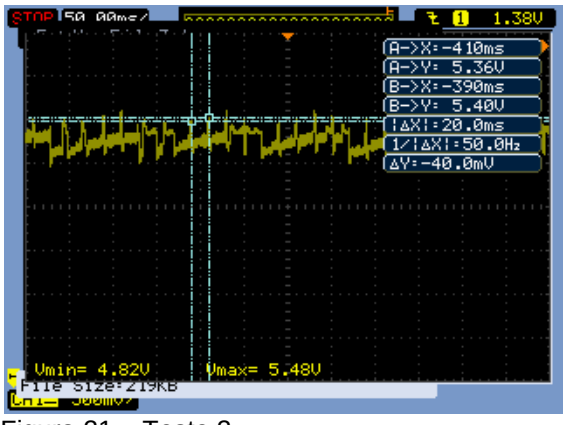


Figura 21 – Teste 2.
Fonte: Autoria própria.

3. Teste com acionamento do motor e capacitor de poliéster de 470nF (Figura 22).

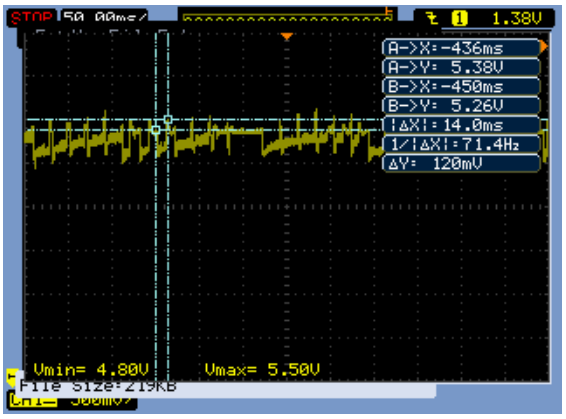


Figura 22 – Teste 3
Fonte: Autoria própria.

4. Teste com acionamento do motor com diodo modelo UF4007, com intuito de funcionalidade como diodo de roda livre (Figura 23).

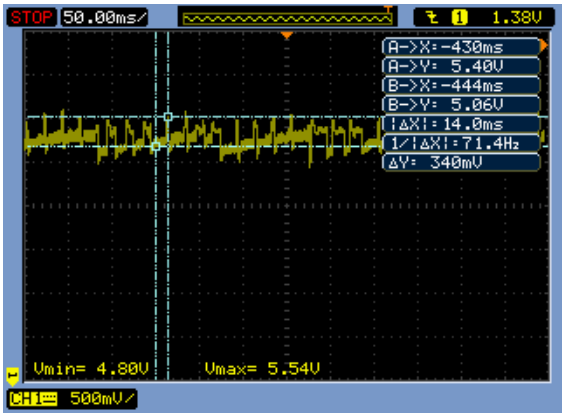


Figura 23 – Teste 4.
Fonte: Autoria própria.

5. Teste com acionamento do motor e capacitor eletrolítico de 10µF (Figura 24).

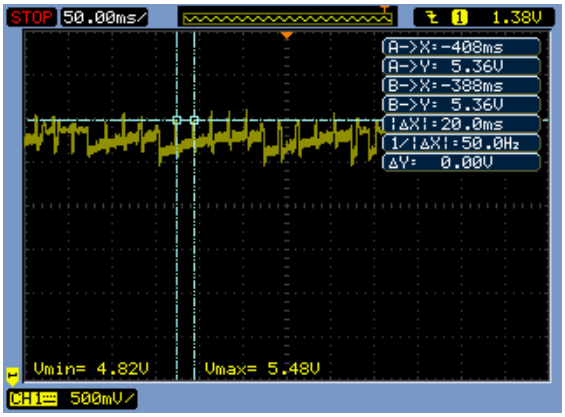


Figura 24 – Teste 5.
Fonte: Autoria própria.

6. Teste com acionamento do motor e capacitor eletrolítico de 47µF (Figura 25).

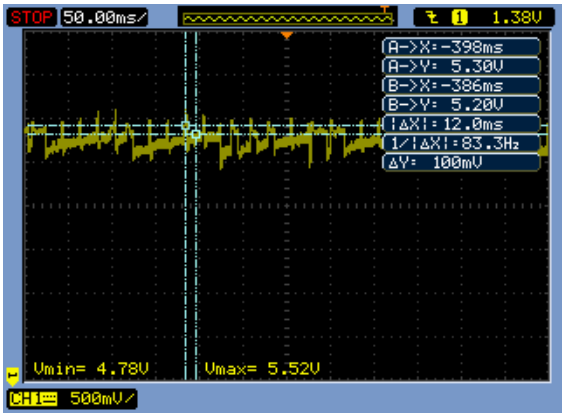


Figura 25 – Teste 6.
Fonte: Autoria própria.

4.13 FIRMWARE E SOFTWARE

Para funcionamento e integração do sistema foi necessário desenvolver o *firmware* do microcontrolador (apêndice K) e o *software* de interface ao usuário.

4.13.1 Firmware

A programação do sistema envolveu duas etapas distintas, desenvolvimento do firmware de controle para o Arduino, responsável pelo cálculo e controle da posição dos motores e a estrutura de controle do aplicativo Android, que oferece a interface ao usuário para acessar as funcionalidades da plataforma. A integração entre os dois é realizada via *bluetooth.* As principais estruturas definidas para o Arduino são a recepção e decodificação do comando recebido do aplicativo, a função de cinemática inversa para o cálculo das variáveis de junta e as funções de trajetória e de controle de movimento da plataforma, que comandam os motores de acordo com as variáveis calculadas. O diagrama da Figura 26 mostra um esquema simplificado do funcionamento do sistema.

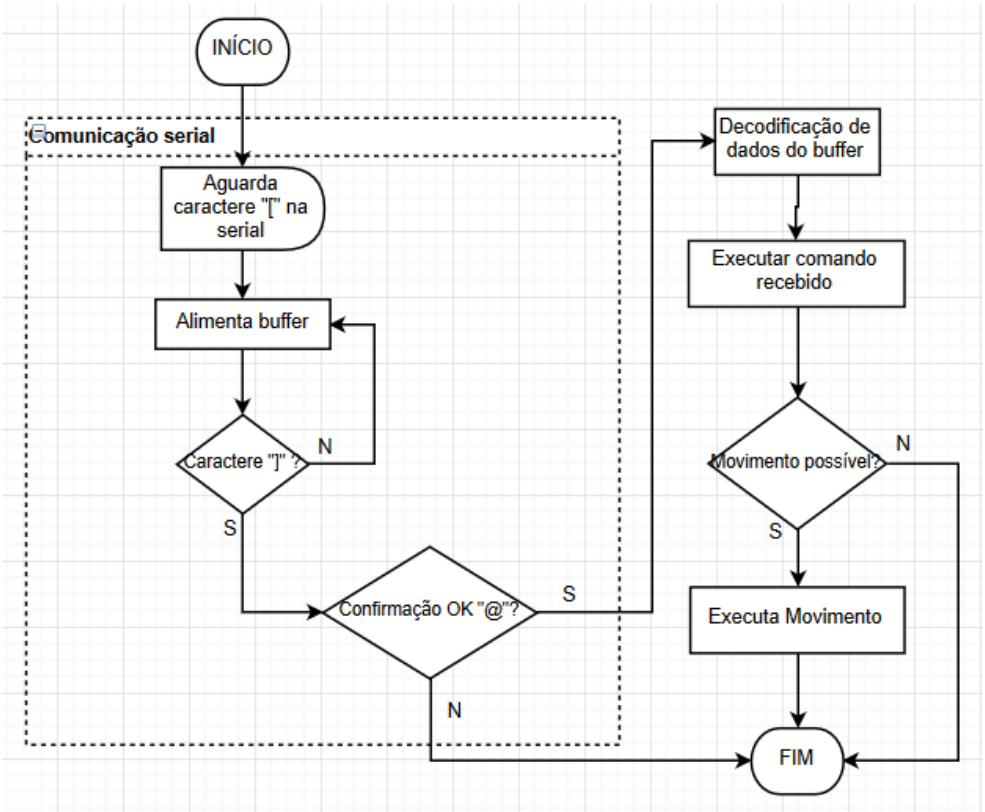


Figura 26 - Fluxograma de funcionamento do sistema.
Fonte: Autoria própria.

No Arduino a execução do programa ocorre na função *loop*, que a cada ciclo identifica se há dado recebido na porta serial e realiza o tratamento se verdadeiro. Quando o caractere “[” é identificado ele é gravado em um vetor, assim como os caracteres subsequentes até que se receba o caractere “]”, que encerra a mensagem. Uma vez gravada, ela é reenviada ao aplicativo, que compara os dados recebidos com os enviados e retorna uma confirmação ao Arduino pelo caractere “@” quando há consistência. A mensagem recebida define a execução de uma das seguintes opções: controlar as variáveis espaciais da plataforma manualmente, de forma individual e com reprodução simultânea ou acionar trajetórias pré-definidas para demonstração.

Definido o movimento a ser realizado, é necessário calcular as variáveis de junta através da equação de cinemática, sendo definida como uma função que é executada sob demanda das funções de trajetória e controle de movimento. Estas funções foram definidas a partir de equações básicas, que simulam a variação das coordenadas de posição e orientação do ponto central da plataforma de acordo com o movimento desejado. O protocolo de comunicação é composto por uma *string* de tamanho variável, o que permite a utilização de várias estruturas de mensagens. As estruturas utilizadas neste trabalho são apresentadas nas Tabelas 1 e 2.

Tabela 1 - Estrutura de Mensagens do Aplicativo para o Arduino

Mensagem Aplicativo para Arduino	**Sintaxe**
Comando de *Slider*	[X;+99.9;+99.9;+999.9;+99.9;+99.9;+99.9;X]
Comando de botão de demonstração	[X;X;X]
Confirmação	[@]

Fonte: Autoria própria.

Tabela 2 - Mensagens Enviadas do Arduino para o Aplicativo.

Mensagem Arduino para Aplicativo	**Palavra**
Em movimento	mov
Em espera	esp

Fonte: Autoria própria.

4.13.2 Software

O aplicativo de controle da plataforma, desenvolvido através do MIT App Inventor, utiliza uma composição de blocos de comandos pré-definidos, que possuem estruturas lógicas de repetição, funções matemáticas, dentre outras. Elas são ativadas a partir do toque do usuário nos objetos inseridos na interface do aplicativo. A interface consiste em duas telas, a primeira para controle das coordenadas espaciais da plataforma através de *sliders,* que permitem a variação de cada coordenada dentro de uma faixa pré-determinada com reprodução simultânea. A segunda tela possui botões que ativam as funções de trajetórias pré-definidas no Arduino. A Figura 27 apresenta a imagem das duas telas disponíveis.

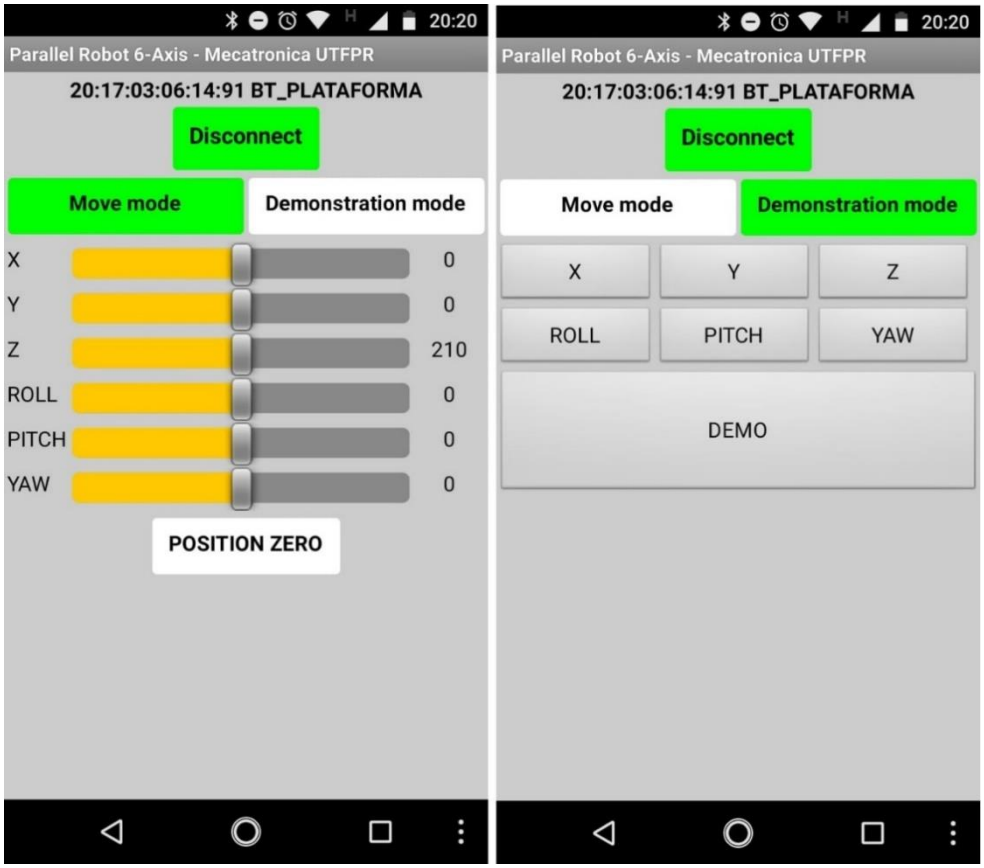


Figura 27 – Telas do aplicativo Andoid.
Fonte: Autoria própria.

4.14 PROJETO E MONTEGEM DO PROTÓTIPO

O projeto do protótipo construído foi realizado com auxílio do *software* SolidWorks (Figura 28). O primeiro passo foi a escolha de itens comerciais como: motores, hastes (apêndice D), *links* (apêndice E), fonte e distanciadores (apêndice G). Definidos os componentes, as plataformas (apêndices B, C e F) foram projetadas com base nos modelos estudados, reproduzindo as formas e disposição dos motores. As medidas foram arbitrárias, pois as dificuldades com relação à robótica inviabilizaram o cálculo do espaço de trabalho do manipulador, que deveria ter sido realizado na fase inicial de projeto. A fabricação foi feita sob encomenda, utilizando placas de policarbonato submetidas a um processo de corte a *laser*. O gabinete (apêndice H) foi fabricado artesanalmente em policarbonato e madeira revestida com material sintético para acabamento.

Figura 28 – Imagem renderizada do projeto CAD.
Fonte: Autoria própria.

Para testes iniciais, a montagem da plataforma foi executada com as hastes plásticas originais dos servomotores, juntas esféricas plásticas e, como hastes da plataforma, “almas” de eletrodos revestidos. Esta escolha foi feita para prevenir possíveis danos aos componentes finais, em decorrência de falhas no controle da plataforma. De fato, ocorreram falhas que poderiam ter causado danos críticos nas hastes de fibra de carbono, escolhidas para montagem final, devido a sua maior fragilidade em relação ao material dos eletrodos. Esta estrutura foi mantida até a validação da estrutura lógica de comando da plataforma. Nela foram inseridas restrições nas variáveis de junta para que não ultrapassem os valores de segurança. A integração dos componentes eletrônicos do sistema foi realizada conforme diagrama funcional apresentado na Figura 29.

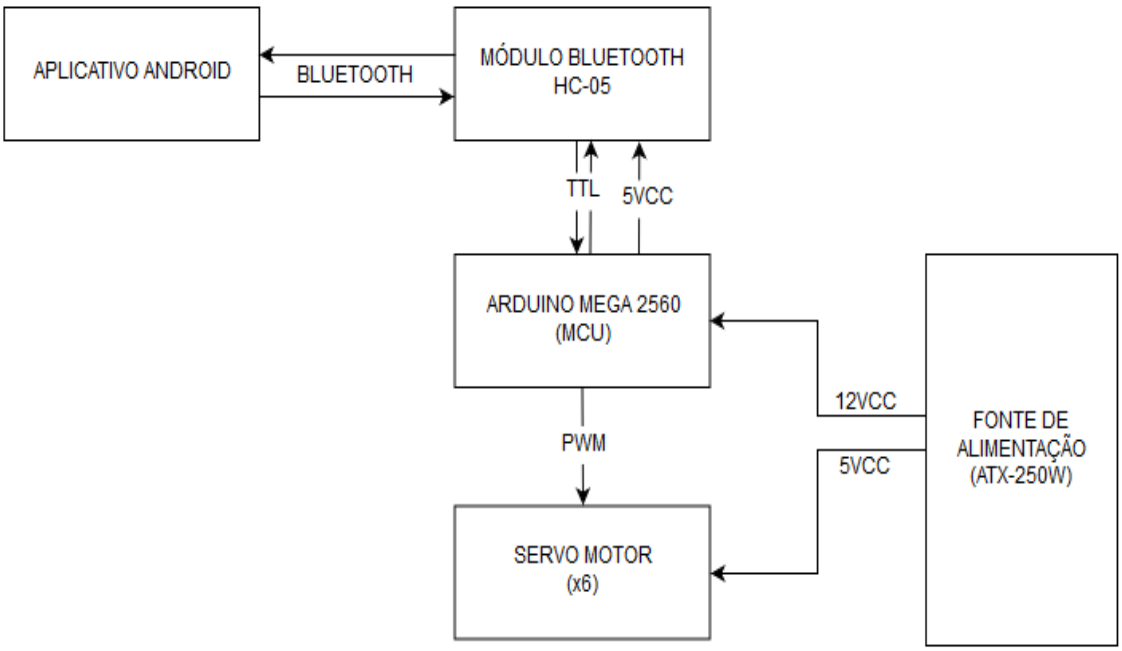


Figura 29 – Diagrama Funcional Eletrônico do Protótipo.
Fonte: Autoria própria.

Para a montagem final, foram utilizados os componentes que foram importados da China. Os primeiros itens testados foram os motores e as articulações em alumínio, para verificar se as furações estavam corretas (Figura 29). Todos foram fixados com parafusos “M3” nos suportes de acrílico e não foram necessários retrabalhos nestes elementos. Foram fixadas então as hastes, juntas e espaçadores. As furações e roscas estavam corretas, porém, as hastes de fibra de carbono chegaram com rachaduras nas extremidades conectadas às juntas, sendo necessário reforça-las com cola e abraçadeiras de *nylon* (Figura 30).

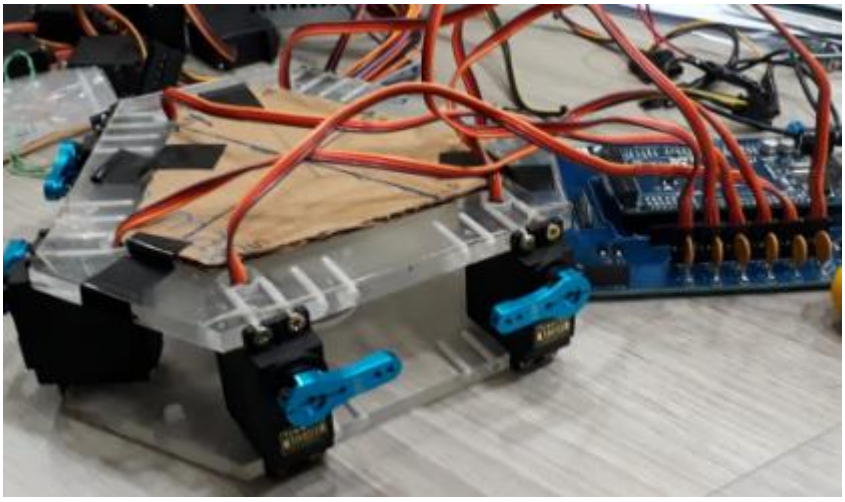


Figura 30 – Fixação das hastes de alumínio nos motores.
Fonte: Autoria própria

Figura 31 – Reforço alternativo das hastes de fibra de carbono.
Fonte: Autoria própria.

Uma vez montada a estrutura mecânica do manipulador, foi construído o gabinete de madeira. Ele serve como base para plataforma e acomoda a fonte de alimentação e os componentes eletrônicos do sistema. Suas partes foram montadas utilizando cola e parafusos autoatarraxantes. A placa eletrônica foi interligada mecanicamente ao gabinete através de distanciadores (com rosca) de tamanhos comerciais e parafusos “M3”. Com gabinete e manipulador montados, o último processo foi a ligação entre os dois, que foi realizada na parte de acrílico utilizando

como fixação parafusos e porcas de tamanho “M4”. O produto final pode ser visto nas Figuras 32, 33 e 34. Os custos do protótipo estão detalhados na tabela 3.

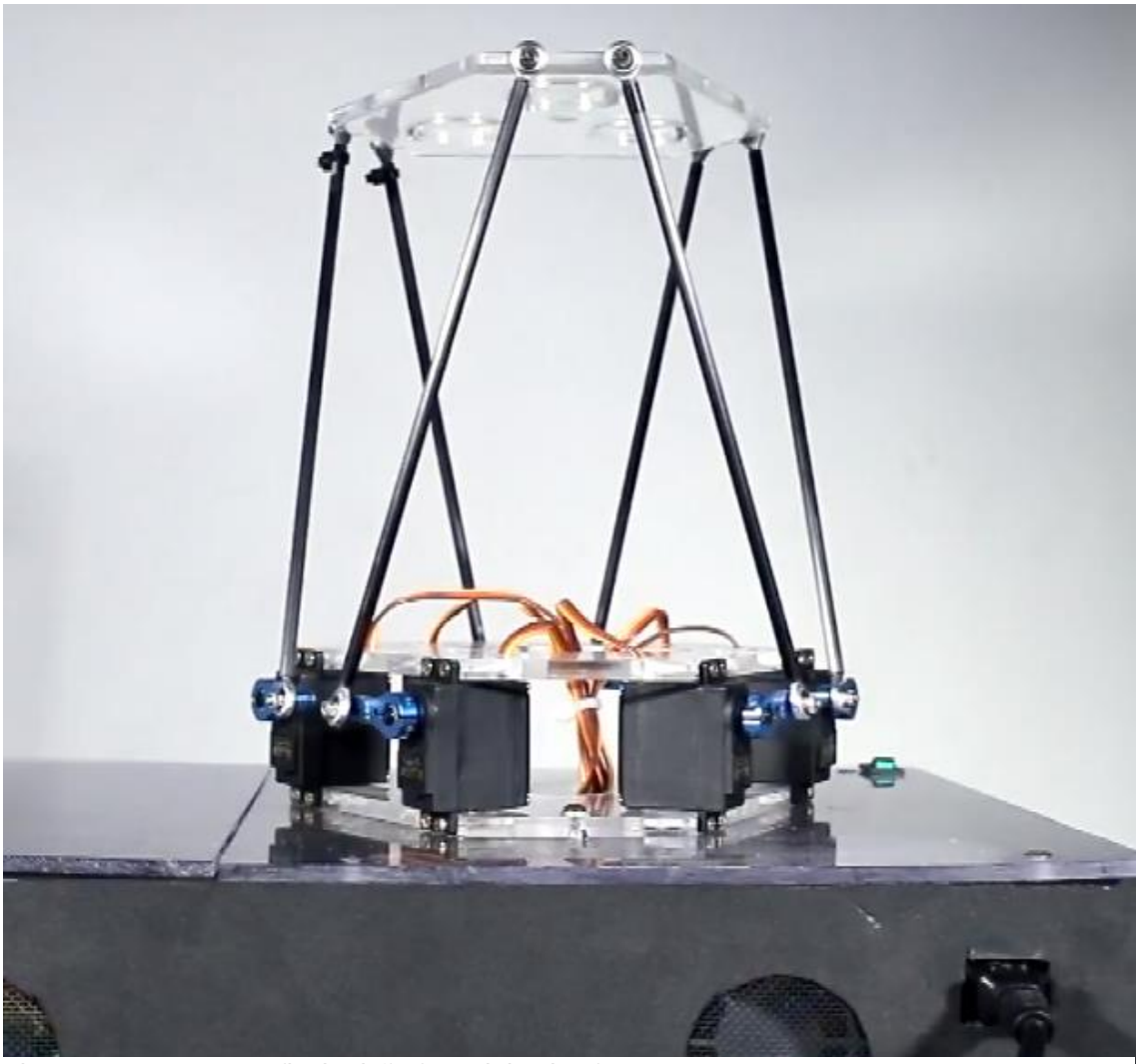


Figura 32 – Montagem final: Visão frontal da plataforma.
Fonte: Autoria própria.

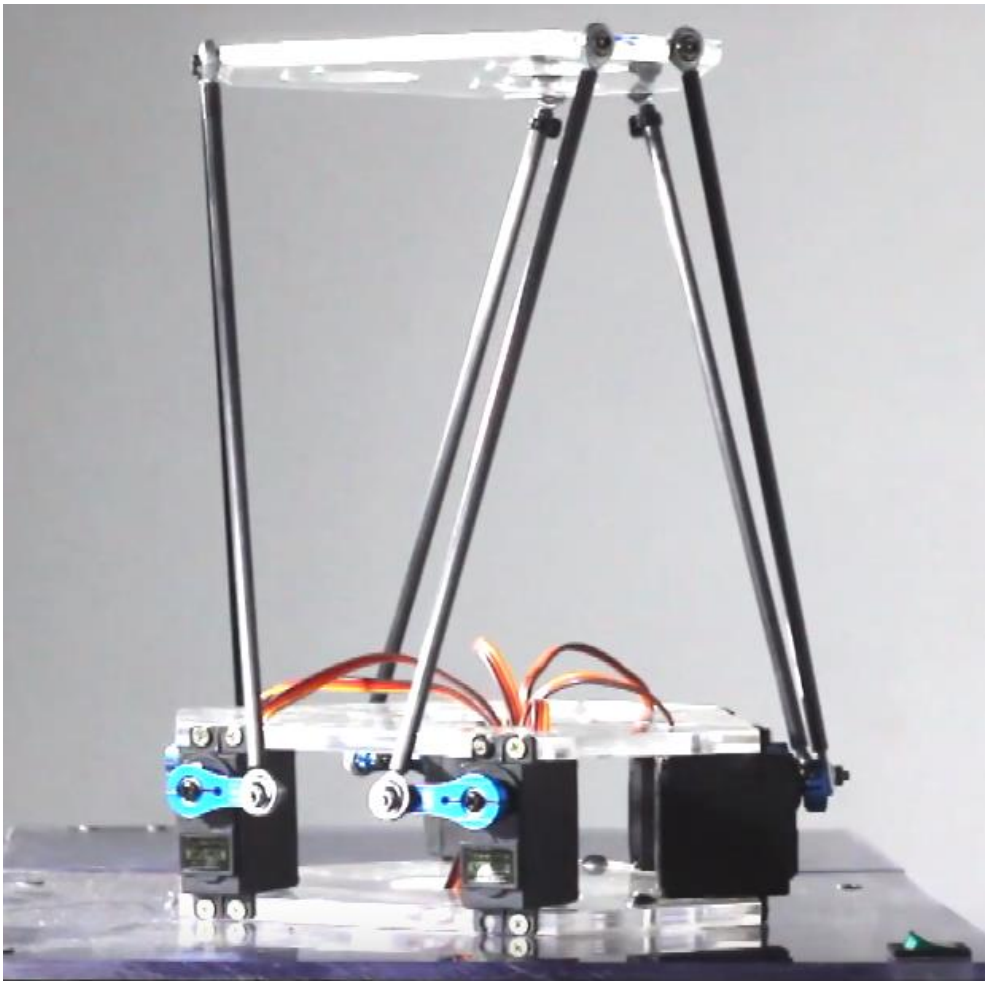


Figura 33 – Montagem final: Visão lateral da plataforma.
Fonte: Autoria própria.

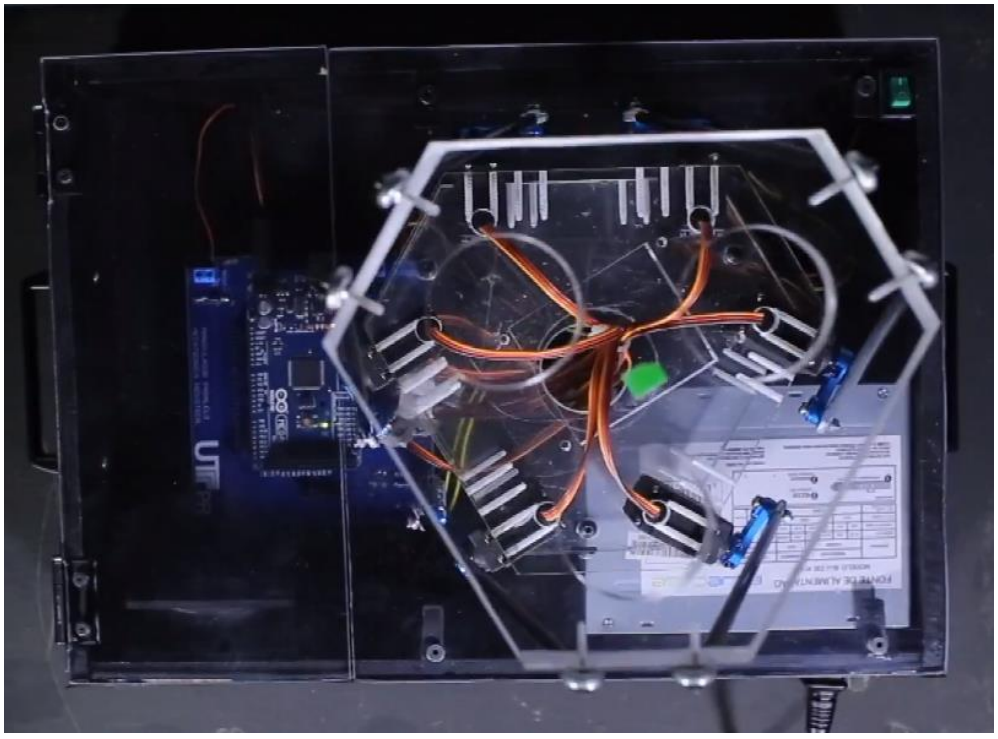


Figura 34 – Montagem final: Visão superior da plataforma.
Fonte: Autoria própria.

Tabela 3 – Custos dos componentes da plataforma.

COMPONENTE	QTD	PREÇO UNITÁRIO	PREÇO CONJUNTO
HASTE PLATAFORMA	6	R$ 10,32	R$ 61,92
MOTOR	6	R$ 13,05	R$ 78,30
HASTE MOTOR	6	R$ 7,28	R$ 43,68
FONTE	1	R$ 50,00	R$ 50,00
ARDUINO MEGA 2560	1	R$ 63,90	R$ 63,90
PLACA SINAIS	1	R$ 70,00	R$ 70,00
COMPONENTES PLACA	1	R$ 65,00	R$ 65,00
PLATAFORMAS ACRILICO	3	R$ 60,00	R$ 180,00
GABINETE	1	R$ 60,00	R$ 60,00
		TOTAL	**R$ 672,80**

Fonte: Autoria própria.

6 CONSIDERAÇÕES FINAIS

O protótipo construído apresenta um sistema mecatrônico bastante completo, permitindo o estudo e demonstração prática de todos os elementos que o compõe, do projeto à construção e funcionamento. Este trabalho oferece também possibilidades de estudos futuros para melhoria em controle e expansão funcional. A proposta de desenvolver uma ferramenta didática para atividades práticas foi cumprida, apresentando um custo relativamente baixo. Além de utilizar recursos de software de fonte aberta, gratuitos e de fácil acesso para estudantes. Apesar de um dos itens iniciais ter sido retirado do projeto, o controle PID em malha fechada, o objetivo geral foi atingido.

Durante o desenvolvimento do protótipo, algumas dificuldades encontradas remetem à baixa qualidade dos componentes comerciais. Os motores, hastes e juntas foram importados da China por questão de custo, já que os componentes encontrados no Brasil são fruto da mesma importação a um custo maior. Componentes de qualidade comprovada apresentaram custos inviáveis. Esta opção trouxe consequências como: problemas com repetibilidade no posicionamento dos motores, características de funcionamento diferentes entre itens do mesmo lote, baixa resistência nas hastes da plataforma, que ocasionaram rachaduras, e roscas espanadas. Houve também problemas na placa de circuito eletrônico produzida sob encomenda, o que quase comprometeu o funcionamento do protótipo.

Apesar dos percalços construtivos, a maior dificuldade do projeto foi com relação às limitações de conhecimento técnico em relação à robótica, só resolvidas graças ao professor orientador do trabalho, Luís Paulo Laus. A escolha inicial de desenvolver um sistema supervisório utilizando o *software* LabView também gerou alguns transtornos. Optar por desenvolver um aplicativo para Android foi uma saída interessante, pela facilidade em desenvolvê-lo no MIT App Inventor e a possibilidade de utilizá-lo em qualquer telefone celular com este sistema operacional e comunicação *bluetooth*. Ele também dispensa o uso licença, proporcionando mobilidade e maior interatividade. Outro ponto determinante foi a dificuldade encontrada na elaboração do *firmware* do microcontrolador, a falta de experiência e conhecimentos mais refinados em lógica e linguagem de programação, em geral, o que acabou travando o desenvolvimento do projeto em várias ocasiões. Porém, o aprendizado decorrente das limitações iniciais é bastante válido.

Ao longo de todo processo de estudo, pesquisa e desenvolvimento, motivados pelas necessidades e problemas que a prática proporcionou, foi possível perceber que o aprendizado para os integrantes da equipe foi mais aprofundado e permitiu uma maior retenção do conhecimento, se comparado a estudos puramente teóricos vistos durante a graduação. Isso nos permite crer que, além do desenvolvimento de uma ferramenta didática, a ideia de reforçar o conhecimento teórico através da prática traz resultados positivos. Foi justamente esta ideia que motivou a escolha do tema deste trabalho de conclusão de curso, a partir de dificuldades vivenciadas em disciplinas teóricas ao longo da graduação. Ao final deste trabalho, identificamos também algumas oportunidades para trabalhos futuros:

- fechar a malha do sistema para possibilitar a implementação do controle PID de posição da plataforma;
- desenvolver um sistema de calibração eficiente dos motores;
- melhorar o sistema de supervisão e controle;
- otimizar o espaço de trabalho do manipulador paralelo;
- desenvolver fonte de alimentação ou circuito para supressão dos ruídos gerados no acionamento dos motores;
- implementar um interpolador de trajetória para suavizar a movimentação do manipulador.

REFERÊNCIAS

ACUÑA, Hernán Gonzalez. **Projeto mecatrônico de uma plataforma Stewart para simulação dos movimentos nos navios**. 2009. 112f. Dissertação (Mestrado em Engenharia Mecânica) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

AMOROSO, Anderson Levati. Notas de aula no curso de Tecnologia em Mecatrônica Industrial, disciplina de Teoria de Controle. Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

ARDUINO. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardMega2560>. Acesso em: 24/fevereiro/2015.

BEGA, Egídio Alberto. **Instrumentação aplicada ao controle de caldeiras 3ª edição**. Rio de Janeiro. Interciência. 2003.

BLUETOOTH SPECIAL INTEREST GROUP (SIG). **BLUETOOTH SPECIFICATION Version 2.0 + EDR [vol 0]**. Disponível em: <https://www.bluetooth.com/specifications/qualification-test-requirements>. Acesso em: 24/abril/2018.

ELOTOUCH SOLUTIONS. Como funciona um Touchscreen AccuTouch. Disponível em <http://www.elotouch.com.br/Produtos/Touchscreens/AccuTouch/accworks.asp>. Acesso em: 27/agosto/2015.

GONÇALVES, Rogério Sales. **Estudo de rigidez de cadeias cinemáticas fechadas.** 2009. 239 f. Tese (Doutorado em Engenharia Mecânica) – Universidade Federal de Uberlândia, Uberlândia, 2009.

ION, Florence. **From touch displays to the surface: a brief history of touchscreen technology**. Disponível em: <http://arstechnica.com/gadgets/2013/04/from-touch-dis:plays-to-the-surface-a-brief-history-of-touchscreen-technology/1/>. Acesso em: 27/agosto/2015.

LAUS, Luís Paulo. **Robótica: cinemática de robôs.** Notas de aula no curso de Tecnologia em Mecatrônica Industrial, disciplina de Robótica. Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

SOARES, Karla. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2013/10/o-que-e-um-arduino-e-o-que-pode-ser-feito-com-ele.html>. Acesso em: 24/fevereiro/2015.

MEDEIROS, Marinalva Veras; CABRAL, Carmen Lúcia de Oliveira. **Formação docente: da teoria à prática em uma abordagem sócio-histórica. Revista e-curriculum**, ISSN 1809-3876, São Paulo, v.1, n.2, junho de 2006. Disponível em: <http://www.pucsp.br/ecurriculum>. Acesso em 11/novembro/2014.

MELLO, Camilla Bacellar. **Controle de trajetória de uma plataforma Stewart para simulação de transferência de carga fora de porto**. 2011. 114f. Dissertação (Mestrado em Engenharia Mecânica) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2011.

MERLET, Jean-Pierre. **Parallel Robots**. 2.ed. Sophia-Antipolis: Springer, 2006.

NATIONAL INSTRUMENTS. Disponível em: <http://www.ni.com/labview/pt/>. Acesso em: 07/maio/2015.

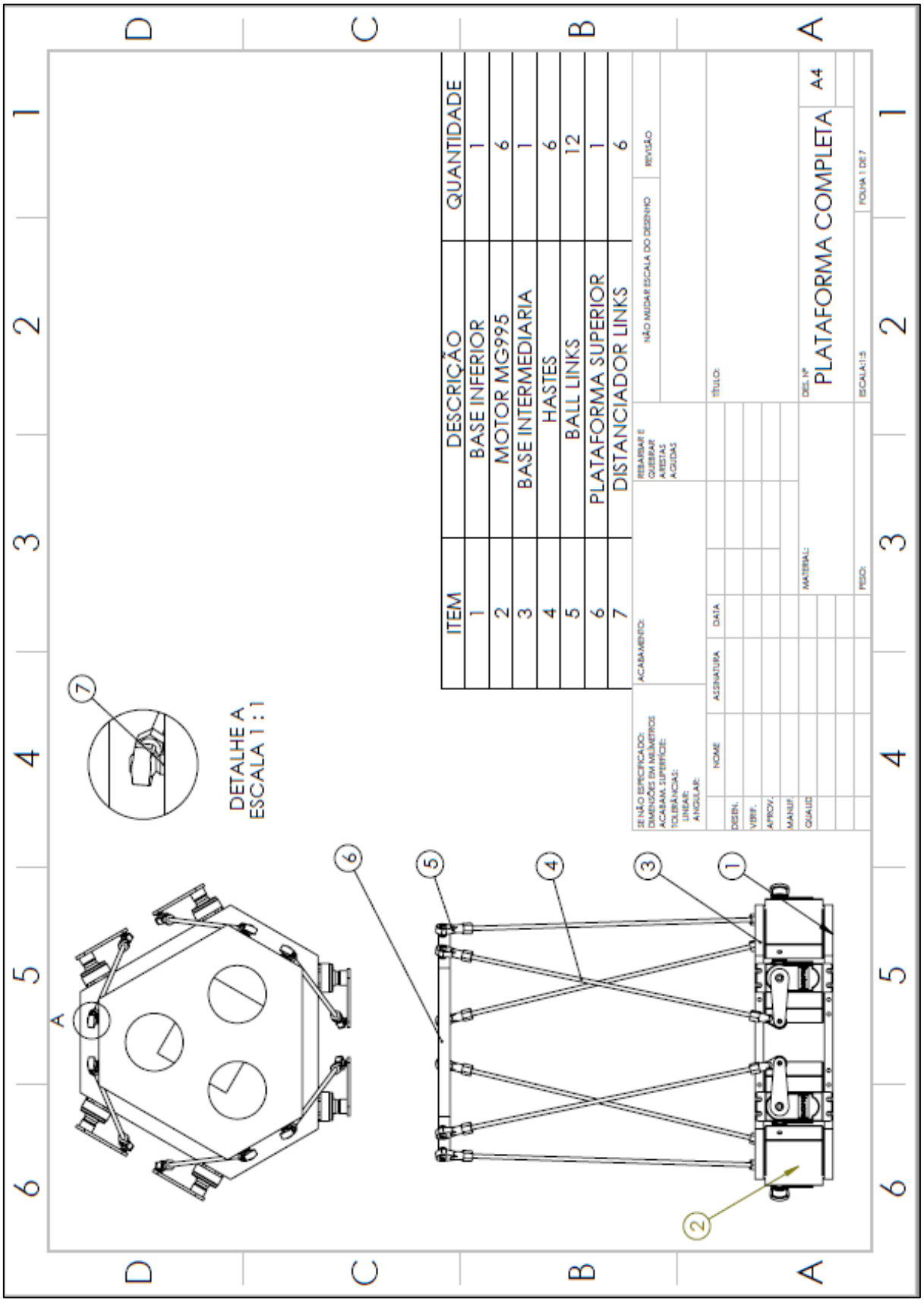
NISE, Norman S. **Engenharia de sistemas de controle 3ª edição**. Rio de Janeiro: LTC. 2002.

OGATA, Katsuhiko; tradutora Heloísa Coimbra de Souza; revisor técnico Eduardo Aorun Tannuri. **Engenharia de controle moderno 5ª edição**. São Paulo. Pearson Prentice Hall. 2010.

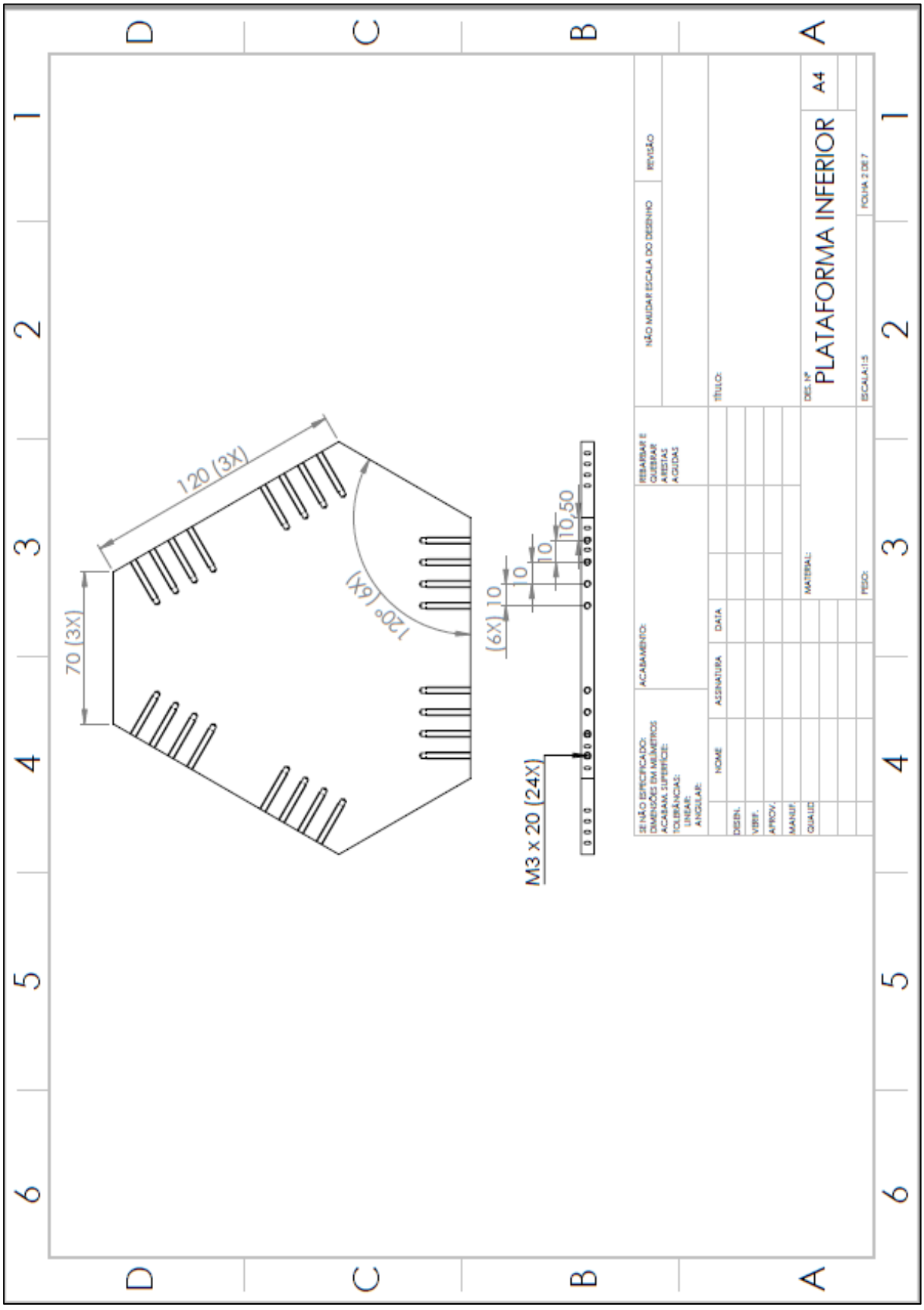
SERVOCITY. *How Does a Servo Work?* Disponível em: <http://www.elotouch.com.br/Produtos/Touchscreens/AccuTouch/accworks.asp>. Acesso em: 27/agosto/2015.

WALKER, Geoff. **Part 1: Fundamentals of Projected-Capacitive Touch Technology**. Disponível em: <http://walkermobile.com/Touch_Technologies_Tutorial_Latest_Version.pdf> Acesso em: 27/agosto/2015.

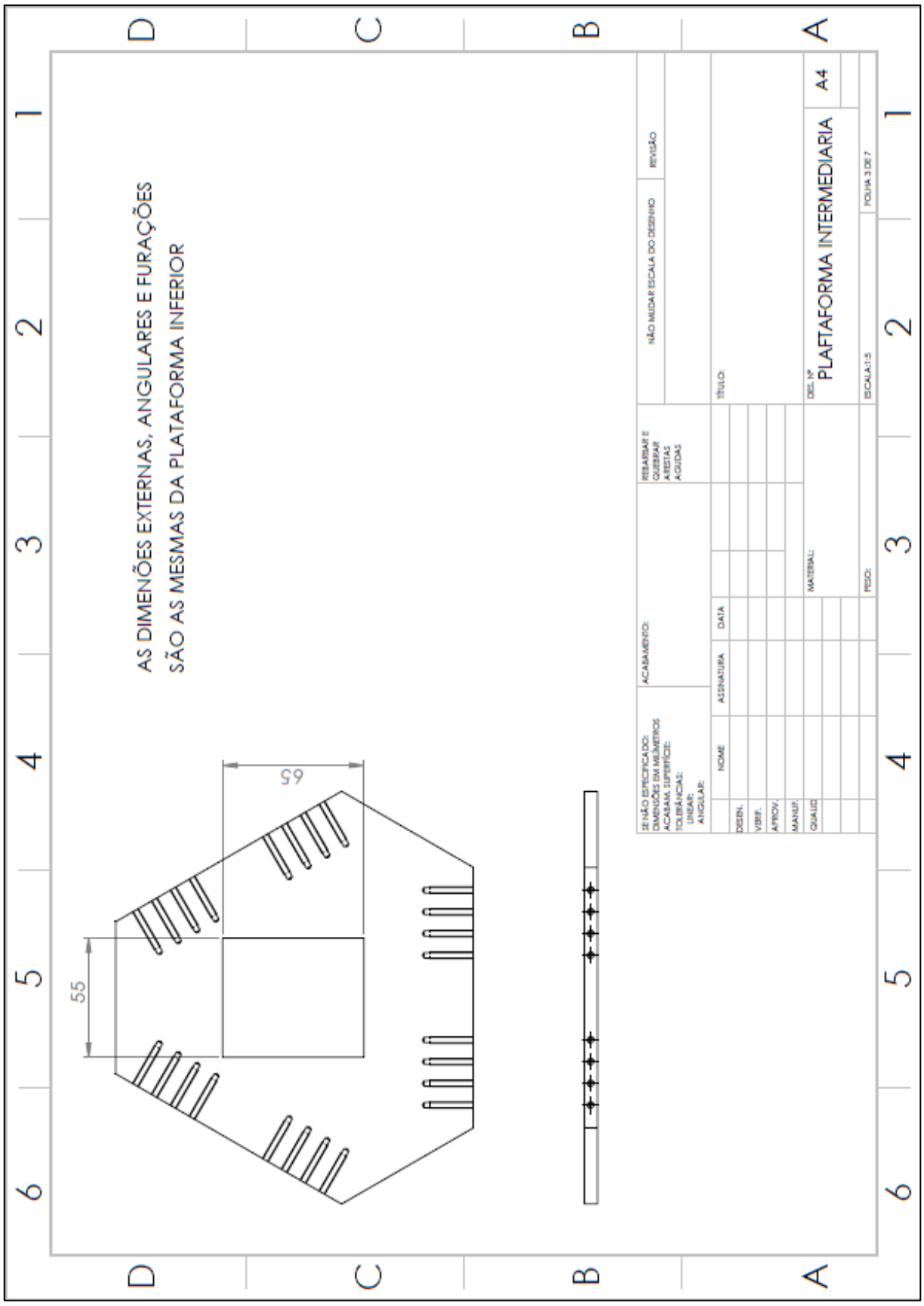
APENDICE A – DESENHO PLATAFORMA COMPLETA



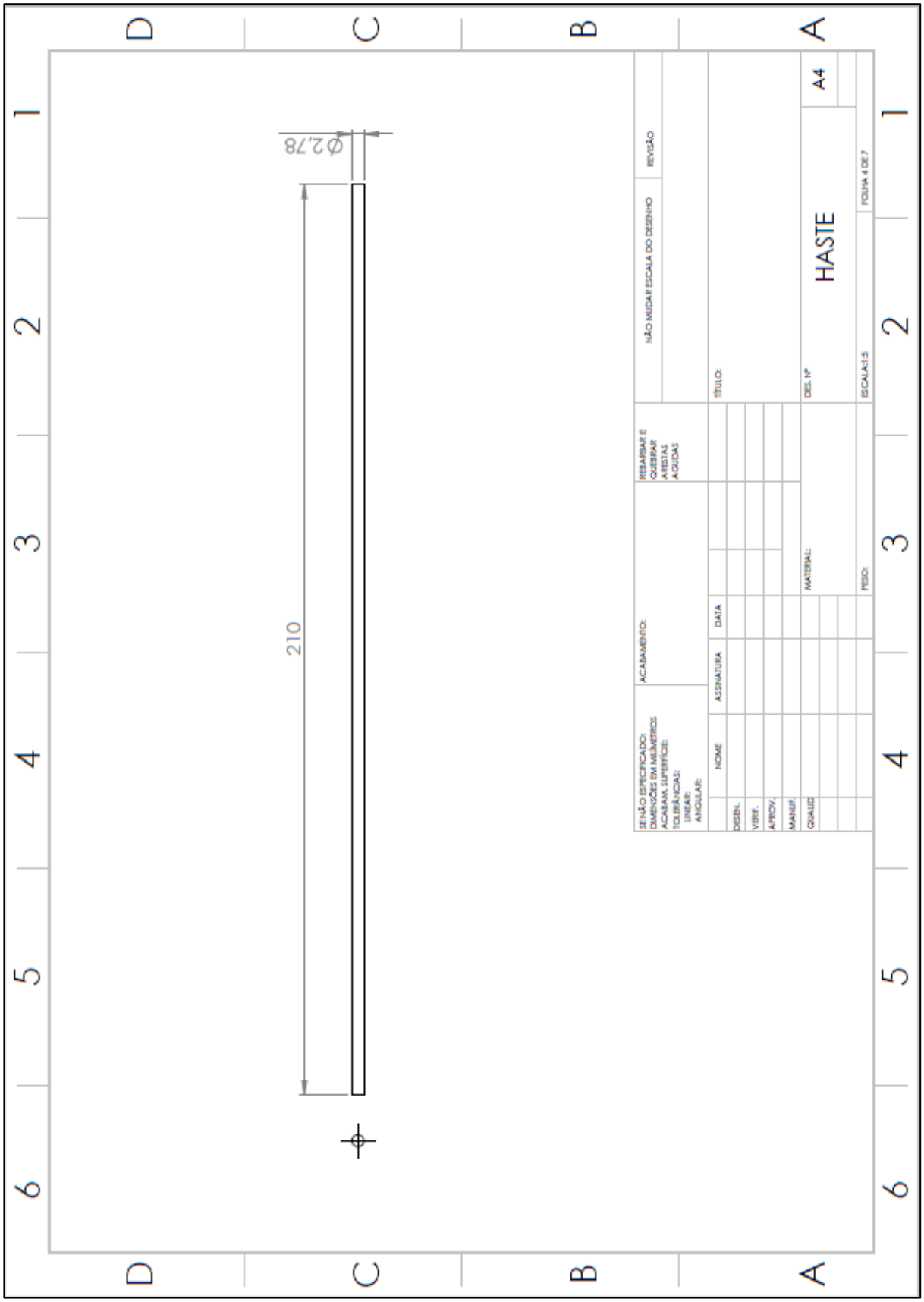
APENDICE B – DESENHO PLATAFORMA INFERIOR



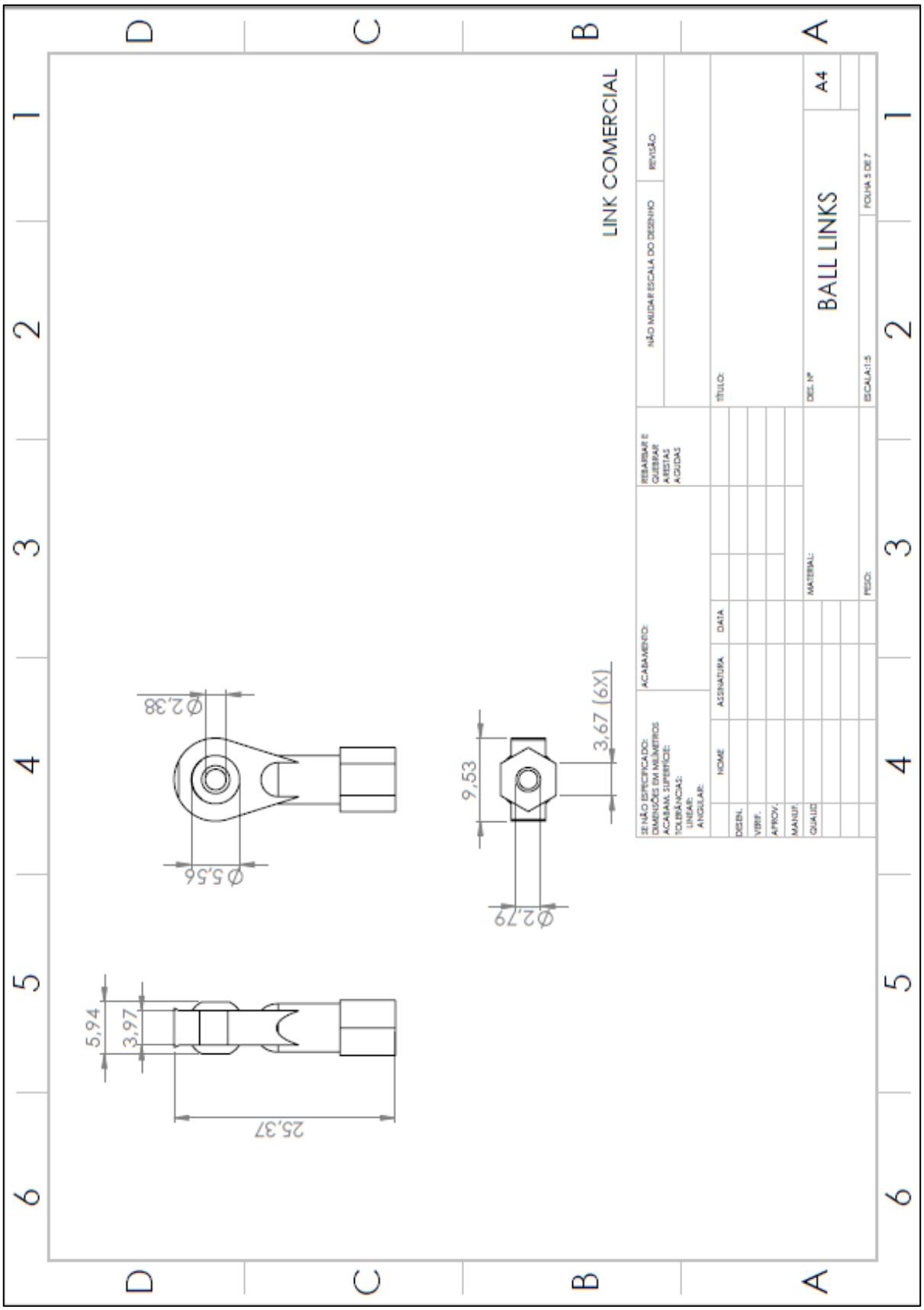
APENDICE C – DESENHO PLATAFORMA INTERMEDIÁRIA



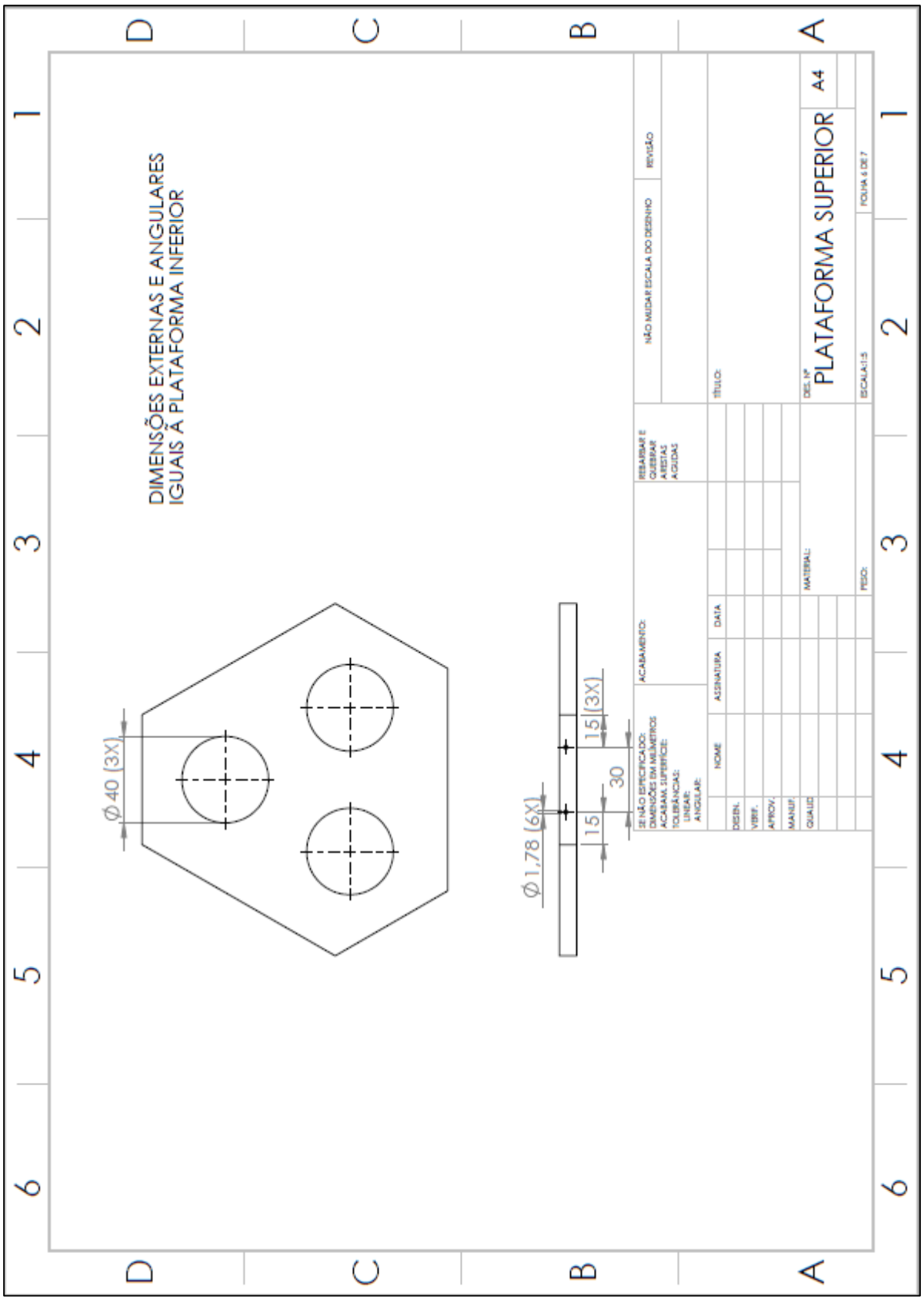
APENDICE D – DESENHO HASTE



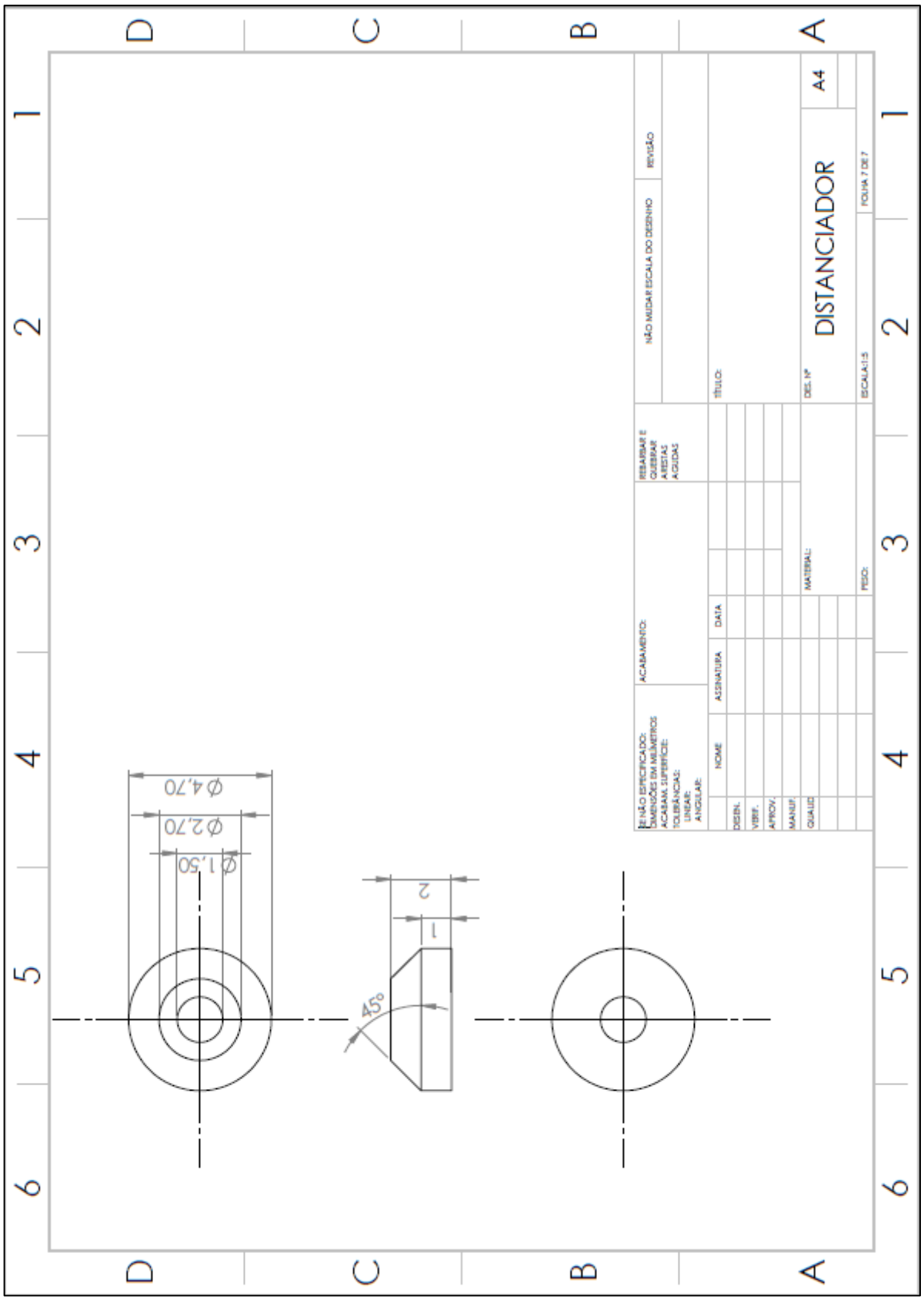
APENDICE E – DESENHO *BALL LINKS*



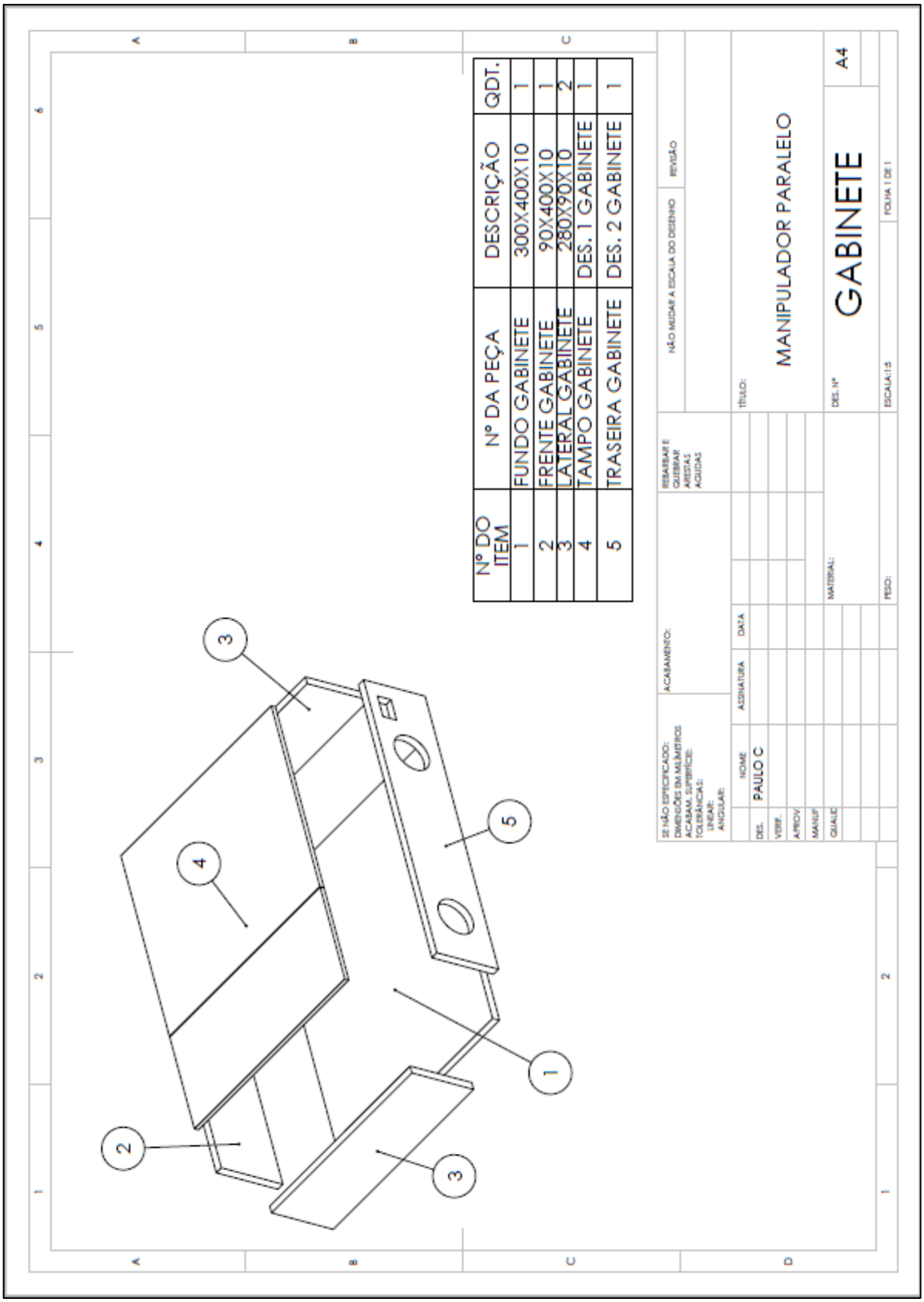
APENDICE F – DESENHO PLATAFORMA SUPERIOR



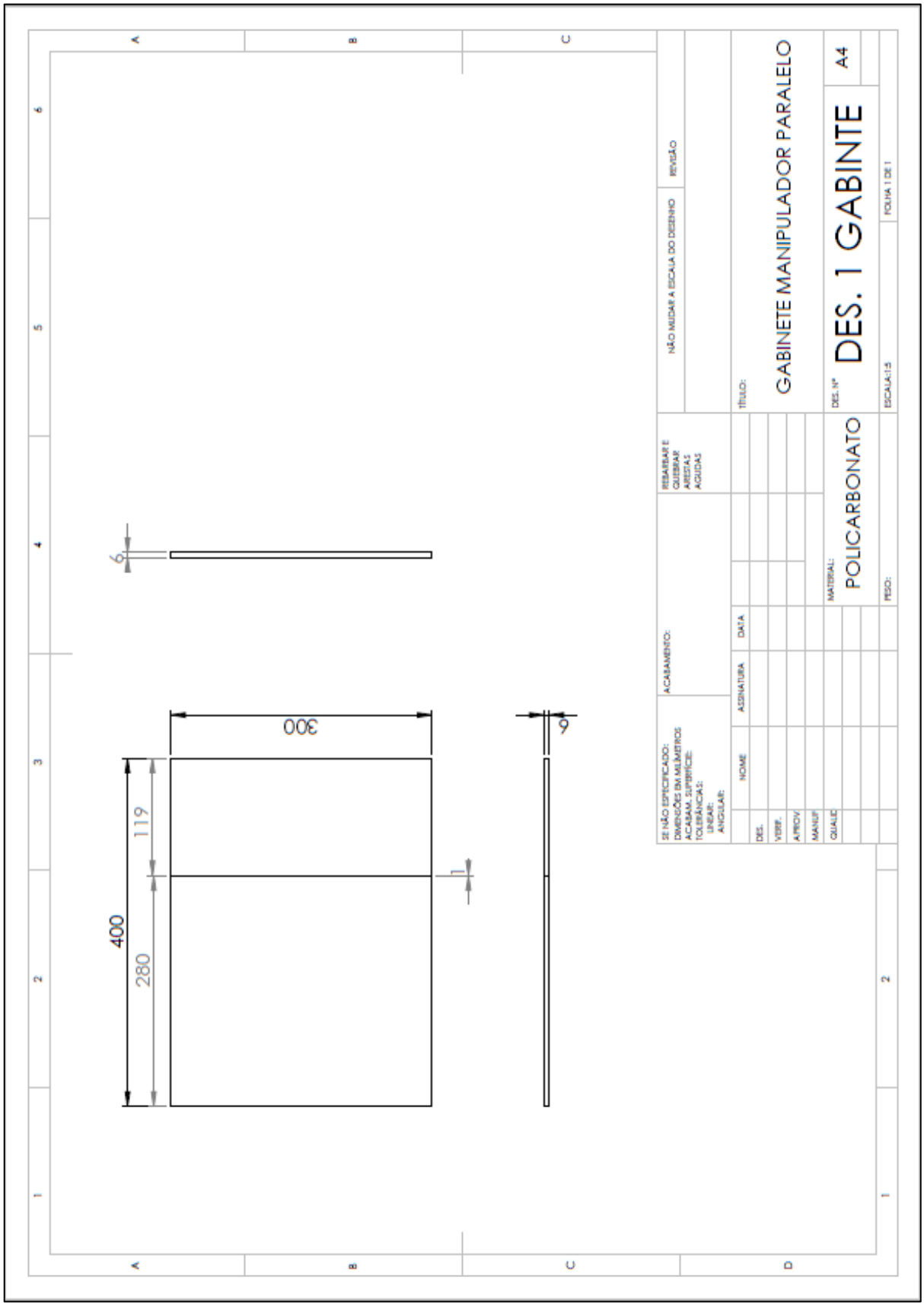
APENDICE G – DESENHO DISTANCIADOR LINKS



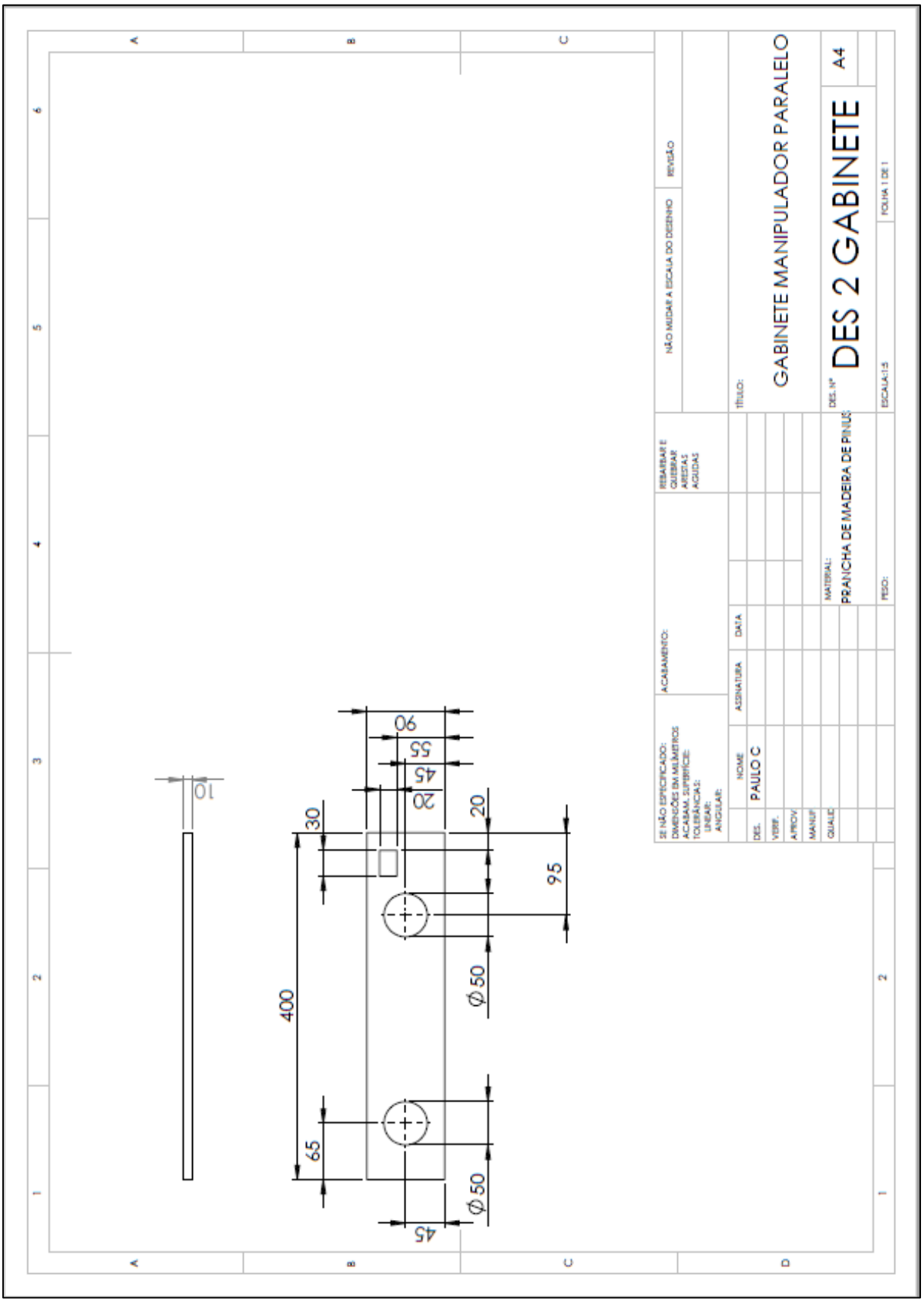
APENDICE H – DESENHO GERAL GABINETE



APENDICE I – DESENHO 1 GABINETE



APENDICE J – DESENHO 2 GABINETE



APENDICE K – FIRMWARE DO MICROCONTROLADOR

```
/*Programa Arduino referente ao trabalho de conclusão de curso dos alunos:
* Paulo Cesar Martins, Paulo H. Garcia de Sá e Rodrigo de Carvalho Ywata, do curso de
* Tecnologia em Mecatrônica Industrial na Universidade Tecnológica Federal do Paraná, UTFPR.
* Funções principais:
* - stof - converte string em float;
* - data_ready - reenvia a mensagem recebida;
* - data_confirmed - executa movimentos após confirmação do comando;
* - theta - função de cinemática inversa, calcula variáveis de junta;
* - Funções de trajetória: servo_circ_xy, servo_circ_xy_z, servo_circ_rp, func_dof;
*/

/* Protocolo de comunicação
   Iniciador  [
   Terminador ]
   Divisor ;
   Exemplo: [dado_1;dado_2;...;dado_n]

   Comando1 [X;+99.9;+99.9;+999.9;+99.9;+99.9;+99.9;X]
   Comando2 [X;X;X]
   Confirmação [@]
   Em movimento "01"
   EM espera "00"
*/

#include <stdio.h>
#include <math.h>

#define LED 13
#define RX_BUFFER_SIZE 42 //Tamanho do buffer em bytes, 16 -> array = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
#define TX_BUFFER_SIZE 42
#define VAR 10
#define COMMAND_SIZE 10
#define ACTION_SIZE 3
#define SIZEN 5
#define SIZEZ 6
#define VALUE_SIZE 6

// medidas dos componentes da plataforma
// Base
#define aa 90.34
#define bb 44.5
// Hastes
#define dd 215.0 //link plataforma
#define gg 25.0  //haste servomotor
```

```
// Plataforma
#define ee 53.83
#define ff 63.23
#define zz 210.0
float z = 210.0;
float x = 0.0;
float y = 0.0;
float agx = 0.0;
float agy = 0.0;
float agz = 0.0;
float rad = 0.0;
int p = 0;

// Servos
float L[] = {3060,   // largura de pulso para posicao zero do motor 1
             2820,   // largura de pulso para posicao zero do motor 2
             3140,   // largura de pulso para posicao zero do motor 3
             2820,   // largura de pulso para posicao zero do motor 4
             2900,   // largura de pulso para posicao zero do motor 5
             3000 // largura de pulso para posicao zero do motor 6
            };
// converte de radianos para largura de pulso
// motores calibrados a 60 graus ou pi/3 radianos
float k[] = {((4140 - 3060) * 3) / M_PI,
             ((3940 - 2820) * 3) / M_PI,
             ((4320 - 3140) * 3) / M_PI,
             ((3900 - 2820) * 3) / M_PI,
             ((4000 - 2900) * 3) / M_PI,
             ((4120 - 3000) * 3) / M_PI
            };

static float maxtt[] = {60.0, 120.0, 60.0, 120.0, 60.0, 120.0}; // valores máximos dos ângulos
static float mintt[] = {-60.0, -120.0, -60.0, -120.0, -60.0, -120.0}; // valores mínimos dos ângulos
int channel = 1;
int inByte = 0;
float tt[6], angle[6];
int i = 0;
int j = 0;  int t = 0;
String aux;
String aux2;
float var[7];
String var2[2];
int c = 0;
float tempo_up = 0;
float raio = 0;
float tempo_traj = 0.0;
float s = 0;
unsigned long lastUpdate;
```

```
unsigned long lastUpdate2;
char Rx_Buffer[RX_BUFFER_SIZE]; //Array para armazenamento de dados vindos do supervisório
char Tx_Buffer[TX_BUFFER_SIZE]; //Array para armazenar dados que serão enviados
uint8_t rx_buffer_indx;
uint8_t buffer_init;
uint8_t estado_led;
/*_____________________________________________________________________*/

void setup() {
  pinMode(2, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);

  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _BV(WGM11);
  TCCR1B = _BV(WGM13) | _BV(WGM12) | _BV(CS11);
  TCCR3A = _BV(COM3A1) | _BV(COM3B1) | _BV(WGM31);
  TCCR3B = _BV(WGM33) | _BV(WGM32) | _BV(CS31);
  TCCR4A = _BV(COM4A1) | _BV(COM4B1) | _BV(WGM41);
  TCCR4B = _BV(WGM43) | _BV(WGM42) | _BV(CS41);

  ICR1 = 40000;
  ICR3 = 40000;
  ICR4 = 40000;

  OCR3B = L[0]; // pin 2
  OCR3A = L[1]; // pin 5
  OCR4A = L[2]; // pin 6
  OCR4B = L[3]; // pin 7
  OCR1A = L[4]; // pin 11
  OCR1B = L[5]; // pin 12

  Serial.begin(115200);
  while (!Serial) {
    ;
  }
}
/*_____________________________________________________________________*/

void loop() {

  if (Serial.available() != 0)
  {
    char d = Serial.read();
    switch (d)
    {
```

```
      case '[': //Iniciar protocolo
        buffer_init = 1;
        break;
      case ']': //Finalizar protocolo
        buffer_init = 0;
        data_ready();
      break;
     case '@':
       if(buffer_init==0){
                  Serial.write("01");
        data_confirmed();
                  Serial.write("00");
       }
     break;
      default:
        if (buffer_init == 1) {
          Rx_Buffer[rx_buffer_indx] = d;
          rx_buffer_indx++;
        }
        break;
    }
  }
}
/*_____________________________________________________________________________*/

void clear_Rx_Buffer() {
  for (int i = 0; i < RX_BUFFER_SIZE; i++) {
    Rx_Buffer[i] = 0;
  }
}

void clear_var() {
  for (int i = 0; i < 10; i++) {
    Rx_Buffer[i] = 0;
  }
}
/*_____________________________________________________________________________*/

float stof(String array_dado, int c) {
  float valor = 0;
  float decimals = 0;
  uint8_t n;
  if (array_dado[0] == '-') {
    n = 0;
    c = c - 1;
  } else {
    n = 1;
  }

  switch (c){
```

```
    case 5:
      valor += (array_dado.charAt(1 - n) - 48) * 100;
      valor += (array_dado.charAt(2 - n) - 48) * 10;
      valor += (array_dado.charAt(3 - n) - 48);
      decimals += (array_dado.charAt(5 - n) - 48);
      decimals = decimals / 10;
      valor += decimals;
    break;

    case 4:
      valor += (array_dado.charAt(1 - n) - 48) * 10;
      valor += (array_dado.charAt(2 - n) - 48);
      decimals += (array_dado.charAt(4 - n) - 48);
      decimals = decimals / 10;
      valor += decimals;
    break;

    case 3:
      valor += (array_dado.charAt(1 - n) - 48);
      decimals += (array_dado.charAt(3 - n) - 48);
      decimals = decimals / 10;
      valor += decimals;
    break;
  }

  if (n == 0) {
    valor = valor * (-1);
  }

  return valor;
}
/*_____________________________________________________________________________*/

void data_ready() {

  for (int i = 0; i < rx_buffer_indx; i++) {
          Serial.write(Rx_Buffer[i]);
  }
  rx_buffer_indx = 0;
}
/*_____________________________________________________________________________*/

void data_confirmed() {

  i = 0;
  j = 0;
  c = 0;

  if (Rx_Buffer[0] == 'C' || Rx_Buffer[0] == 'T'){
     while(Rx_Buffer[i] != 'Q') {
```

```
        if(Rx_Buffer[i] != ';') {
          aux += Rx_Buffer[i];
          c++;
        }
        else if (0 < j < 7) {
          var[j] = stof(aux,c);
          j++;
          aux = "";
          c = 0;
        }
        i++;
      }
  }

  if (Rx_Buffer[0] == 'C'){
      theta(var[1], (var[2]), (var[3]), (var[4])*M_PI / 180, (var[5])*M_PI / 180,
(var[6])*M_PI / 180, tt);
      for (i = 0; i < 6; i++) {
        angle[i] = k[i] * tt[i] + L[i];
        }

      OCR3B = angle[0]; // envia angulo ao motor 1;
      OCR3A = angle[1]; // envia angulo ao motor 2;
      OCR4A = angle[2]; // envia angulo ao motor 3;
      OCR4B = angle[3]; // envia angulo ao motor 4;
      OCR1A = angle[4]; // envia angulo ao motor 5;
      OCR1B = angle[5]; // envia angulo ao motor 6;
    }

  if (Rx_Buffer[0] == 'T'){

    if (Rx_Buffer[2] == 'D'){
      servo_circ_xy(10000.0,2.0,0.5,20.0);
      zero();
      delay(500);
      servo_circ_xy_z(10000.0,2.0,0.5,20.0);
      zero();
      delay(500);
      servo_circ_rp(10000.0,0.0,0.3,10.0);
      zero();
      delay(500);
    }
    else{
      func_dof(Rx_Buffer[2]);
    }

  }
  clear_Rx_Buffer();
  clear_var();
}
```

```
/*_____________________________________________________________________*/

//funcao cinematica
int theta(float x, float y, float z, float agx, float agy, float agz, float *tt)
{
  float l1, l2, l3, b = bb, f = ff;
  float Qx, Qy, Qz;
  int i;
  static float phi[] = {0.0, 0.0, 120 * M_PI / 180, 120 * M_PI / 180, 240 * M_PI / 180, 240
* M_PI / 180};
  float aux;

  // Cinematica inversa
  for (i = 0; i < 6; i++) {
    // b e f sao negativos para membros impares (primeiro, terceiro e quinto)
    b = -b;
    f = -f;
    // pontos na plataforma (denpendem da posicao;orientacao desejadas)
    Qx = x - (cos(agx) * sin(agz) - cos(agz) * sin(agx) * sin(agy)) * (f * cos(phi[i]) + ee *
sin(phi[i])) + cos(agy) * cos(agz) * (ee * cos(phi[i]) - f * sin(phi[i])) ;
    Qy = y + (cos(agx) * cos(agz) + sin(agx) * sin(agy) * sin(agz)) * (f * cos(phi[i]) + ee *
sin(phi[i])) + cos(agy) * sin(agz) * (ee * cos(phi[i]) - f * sin(phi[i])) ;
    Qz = z - sin(agy) * (ee * cos(phi[i]) - f * sin(phi[i])) + cos(agy) * sin(agx) * (f * cos(phi[i])
+ ee * sin(phi[i]));

    l1 = 2.0 * (aa * cos(phi[i]) - b * sin(phi[i])) * Qx + 2 * (b * cos(phi[i]) + aa * sin(phi[i]))
* Qy + dd * dd - aa * aa - b * b - gg * gg - Qx * Qx - Qy * Qy - Qz * Qz;
    l2 = 2.0 * gg * (b + sin(phi[i]) * Qx - cos(phi[i]) * Qy);
    l3 = -2.0 * gg * Qz;

    aux = l2 * l2 + l3 * l3 - l1 * l1;
    if (aux < 0.0) return 0;
    aux = sqrt(aux);
    // calcula theta
    tt[i] = 2.0 * atan((l3 + ((i & 1) ? -aux : aux)) / (l2 + l1)); // +/- bracos para fora, -/+ para
dentro
    // braco de membros impares sao montados na posicao normal (primeiro, i = 0)
    // braco de membros pares sao montados ao contrario (segundo, i = 1)
    if (i & 1) { // bracos pares sao rabatidos de 180 graus
      if (tt[i] >= 0.0)
        tt[i] -= M_PI;
      else
        tt[i] += M_PI;
    }
  }
  return 1;
}
/*_____________________________________________________________________*/
void servo_circ_xy(const float tempo_traj, float tempo_up, float inc, float raio)
{
```

```
  float incz = 26/((2*raio)/inc);
  int teste = 0;
  z = 210.0;
  s = 1;
  lastUpdate = millis();
  lastUpdate2 = millis();

     while (millis() - lastUpdate2 < tempo_traj){
       lastUpdate2 - millis();
       if((millis() - lastUpdate) > tempo_up)
         {
            lastUpdate = millis();
            y = sqrt(abs(raio*raio - x*x)) ;
            x = x += inc;

            theta(x,y*s,z,agx*M_PI/180,agy*M_PI/180,agz*M_PI/180,tt);

            for(i=0;i<6;i++){
             if( (tt[i]*180/M_PI < mintt[i]) || (tt[i]*180/M_PI  >= maxtt[i]) ){
               teste = 1;
               Serial.println("erro");
             }
             angle[i] = k[i]*tt[i]+L[i];
            }

            if ((x >= raio) || (x <= -raio))
              {
                s = -s;
                inc = -inc;
              }
       }

      if(teste == 1){
        break;
      }
      else{
          OCR3B = angle[0]; // envia angulo ao motor 1;
          OCR3A = angle[1]; // envia angulo ao motor 2;
          OCR4A = angle[2]; // envia angulo ao motor 3;
          OCR4B = angle[3]; // envia angulo ao motor 4;
          OCR1A = angle[4]; // envia angulo ao motor 5;
          OCR1B = angle[5]; // envia angulo ao motor 6;
      }
    }
     zero();
}
/*_____________________________________________________________________*/
void servo_circ_xy_z(const float tempo_traj, float tempo_up, float inc, float raio)
{
  float incz = 26/((2*raio)/inc);
```

```
  int teste = 0;
  z = 210.0;
  s = 1;
  lastUpdate = millis();
  lastUpdate2 = millis();

    while (millis() - lastUpdate2 < tempo_traj){
      lastUpdate2 - millis();
      if((millis() - lastUpdate) > tempo_up)
        {
          lastUpdate = millis();
          y = sqrt(abs(raio*raio - x*x)) ;
          x = x += inc;
          z = z += incz;

          theta(x,y*s,z,agx*M_PI/180,agy*M_PI/180,agz*M_PI/180,tt);

          for(i=0;i<6;i++){
           if( (tt[i]*180/M_PI < mintt[i]) || (tt[i]*180/M_PI  >= maxtt[i]) ){
             teste = 1;
             Serial.println("erro");
           }
           angle[i] = k[i]*tt[i]+L[i];
          }

          if ((x >= raio) || (x <= -raio))
            {
              s = -s;
              inc = -inc;
              incz = -incz;
            }
       }
      if(teste == 1){
        break;
      }
      else{
          OCR3B = angle[0]; // envia angulo ao motor 1;
          OCR3A = angle[1]; // envia angulo ao motor 2;
          OCR4A = angle[2]; // envia angulo ao motor 3;
          OCR4B = angle[3]; // envia angulo ao motor 4;
          OCR1A = angle[4]; // envia angulo ao motor 5;
          OCR1B = angle[5]; // envia angulo ao motor 6;
      }
    }
    zero();
}
/*_____________________________________________________________________________*/
void servo_circ_rp(const float tempo_traj, float tempo_up, float inc, float raio)
{
  int teste = 0;
```

```
  float inc2 = inc;
  z = 210.0;
  s = 1;
  lastUpdate = millis();
  lastUpdate2 = millis();

     while (millis() - lastUpdate2 < tempo_traj){
       lastUpdate2 - millis();
       if((millis() - lastUpdate) > tempo_up)
         {
           lastUpdate = millis();
           agy = sqrt(abs(raio*raio - agx*agx));
           agx += inc;

           theta(x,y,z,agx*M_PI/180,s*agy*M_PI/180,agz*M_PI/180,tt);

           for(i=0;i<6;i++){
            if( (tt[i]*180/M_PI < mintt[i]) || (tt[i]*180/M_PI  >= maxtt[i]) ){
              teste = 1;
              Serial.println("erro");
            }
            angle[i] = k[i]*tt[i]+L[i];
           }

           if ((agx >= raio) || (agx <= -raio))
             {
               inc = -inc;
               s = -s;
             }
       }

      if(teste == 1){
        break;
      }
      else{
          OCR3B = angle[0]; // envia angulo ao motor 1;
          OCR3A = angle[1]; // envia angulo ao motor 2;
          OCR4A = angle[2]; // envia angulo ao motor 3;
          OCR4B = angle[3]; // envia angulo ao motor 4;
          OCR1A = angle[4]; // envia angulo ao motor 5;
          OCR1B = angle[5]; // envia angulo ao motor 6;
      }
    }
    zero();
}

/*_____________________________________________________________________*/
void func_dof(int grau)
{
  int teste = 0;
```

```
float inc = 1.0;
float limite = 0;
float pos = 0.0;
s = 1;
tempo_traj = 4000.0;
tempo_up = 10.0;
lastUpdate = millis();
lastUpdate2 = millis();

while (millis() - lastUpdate2 < tempo_traj){
    lastUpdate2 - millis();
    if((millis() - lastUpdate) > tempo_up)
     {
       lastUpdate = millis();
       pos += inc;
       switch (grau){
         case 'X':
           x =+pos;
           limite = 30.0;
         break;

         case 'Y':
           y =+pos;
           limite = 30.0;
         break;

         case 'Z':
           z = zz + pos;
           limite = 15.0;
         break;

         case 'R':
           agx =+pos;
           limite = 12.0;
         break;

         case 'P':
           agy =+pos;
           limite = 12.0;
         break;

         case 'W':
           agz =+pos;
           limite = 12.0;
         break;

       }
       theta(x,y,z,agx*M_PI/180,agy*M_PI/180,agz*M_PI/180,tt);

       for(i=0;i<6;i++){
```

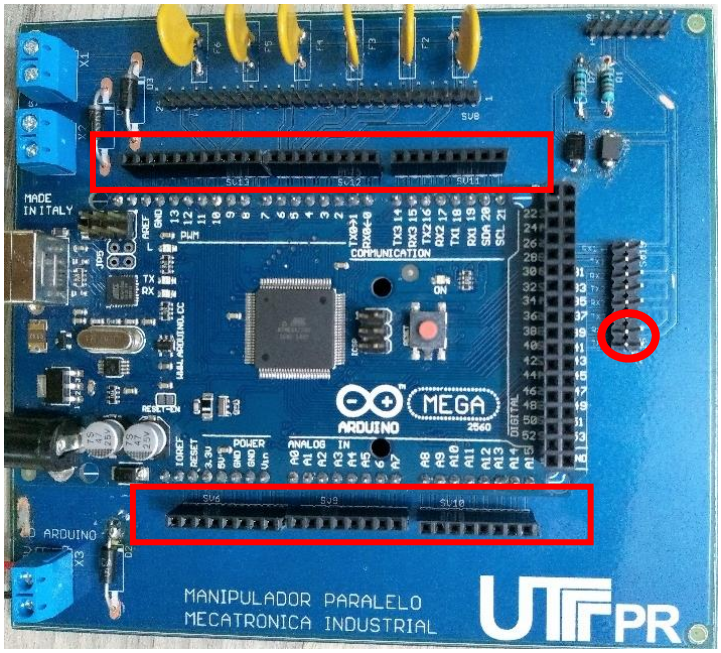
```
            if( (tt[i]*180/M_PI < mintt[i]) || (tt[i]*180/M_PI  >= maxtt[i]) ){
              teste = 1;
              Serial.println("erro");
            }
            angle[i] = k[i]*tt[i]+L[i];
           }

           if ((pos >= limite) || (pos <= -limite))
            {
              inc = -inc;
              s = -s;
            }
       }

      if(teste == 1){
        break;
      }
      else{
          OCR3B = angle[0]; // envia angulo ao motor 1;
          OCR3A = angle[1]; // envia angulo ao motor 2;
          OCR4A = angle[2]; // envia angulo ao motor 3;
          OCR4B = angle[3]; // envia angulo ao motor 4;
          OCR1A = angle[4]; // envia angulo ao motor 5;
          OCR1B = angle[5]; // envia angulo ao motor 6;
      }
    }
    zero();
}
/*_____________________________________________________________________________*/

void zero(){
  OCR3B = L[0]; // pin 2
  OCR3A = L[1]; // pin 5
  OCR4A = L[2]; // pin 6
  OCR4B = L[3]; // pin 7
  OCR1A = L[4]; // pin 11
  OCR1B = L[5]; // pin 12
  tempo_up = 0;
  raio = 0;
  tempo_traj = 0.0;
  s = 0;
  lastUpdate = 0.0;
  lastUpdate2 = 0.0;
  x = 0.0;
  y = 0.0;
  z = 210.0;
  agx = 0.0;
  agy = 0.0;
  agz = 0.0;
  }
```

APENDICE L – INSTRUÇÃO PARA UTILIZAÇÃO DA PLATAFORMA



1. Instalar o aplicativo de supervisão em celular com sistema operacional Android.
2. Parear o celular com o módulo *bluetooth* H-05:
    a. Identificação do Módulo: **BT_PLATAFORMA**;
    b. Senha (PIN): **UTFPR**;
3. Ligar a plataforma no botão de cor verde localizado na tampa do gabinete.
4. O *Led* do módulo fica piscando constantemente quando está aguardando conexão.
5. Clicar no botão "*Connect*" e selecionar o módulo já pareado:
    a. A mensagem "***Bluetooth connected***" é exibida no aplicativo assim que a conexão for estabelecida e o módulo *bluetooth* pisca um *led* a cada 2 segundos;
6. Para carregar novos *firmwares* no Arduino, é necessário retirar os dois *jumpers* dos pinos indicados na figura acima por um círculo vermelho, caso contrário a porta serial de gravação entrará em conflito com a serial do módulo *bluetooth*;
7. As conexões indicadas por retângulo vermelho são as mesmas portas lógicas que estariam disponíveis na placa Arduino Mega2560;

Nota: Arquivos *firmware*, esquemáticos, *layout* e projeto do aplicativo estarão disponíveis com o professor Dr. Luís Paulo Laus.