UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Fabiano Preisler

Projeto de um dispositivo robótico de cadeia aberta em ambiente de simulação

Florianópolis
2018

Fabiano Preisler

Projeto de um dispositivo robótico de cadeia aberta em ambiente de simulação

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação na disciplina **DAS 5511: Projeto de Fim de Curso** do curso de Graduação em Engenharia de Controle e Automação.
Orientador: Prof. Henrique Simas, Dr. Eng.

Florianópolis
2018

Fabiano Preisler

Projeto de um dispositivo robótico de cadeia aberta em ambiente de simulação

Esta monografia foi julgada no contexto da disciplina DAS5511: Projeto de Fim de Curso e aprovada na sua forma final pelo Curso de Engenharia de Controle e Automação.

Florianópolis, 02 de agosto de 2018

Banca Examinadora:

_____
Felipe Leal Sabino
Orientador na Empresa
Instituto SENAI de Inovação em Sistemas Embarcados

_____
Prof. Henrique Simas, Dr. Eng.
Orientador no Curso
Universidade Federal de Santa Catarina

Prof. Júlio Cezar Frantz
Avaliador
Universidade Federal de Santa Catarina

João Victor Zacchi
Debatedor
Universidade Federal de Santa Catarina

Guilherme da Fonseca Pereira
Debatedor
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Aos meus pais, Alvino e Francisca, aos meus irmãos Sérgio, Eliane, Simone e Fábio, pelo incentivo e suporte durante toda a graduação, mesmo que distantes. Nada disso seria possível sem esse apoio.

Ao meu orientador na UFSC, Professor Henrique Simas, pela amizade, pelo apoio e pelas instruções sobre o trabalho desenvolvido.

Ao SENAI pela oportunidade de colocar em prática o que aprendi na graduação. Meu trabalho é reflexo das ótimas condições que pude lá viver. A confiança depositada no trabalho foi decisiva para o sucesso do mesmo.

Aos colegas de trabalho, por todo o espírito de equipe, amizade e por proporcionarem um ótimo ambiente de trabalho.

A todos meus amigos, colegas e parentes, que de alguma forma contribuíram para que eu chegasse até aqui.

RESUMO

O Instituto SENAI de Inovação em Sistemas Embarcados apresenta-se como uma entidade que oferece soluções em diversos segmentos estratégicos tais como indústrias aeroespacial, automotiva, de energia, óleo e gás, automação e TIC. Este PFC está inserido em um projeto do tipo PoC do inglês, *Proof of Concept* que vem sendo desenvolvido no SENAI. Por conta de uma demanda, o instituto ficou responsável pela construção de um manipulador robótico, o qual deveria possuir oito graus de liberdade, na atual etapa o manipulador apresentará apenas três, e com a capacidade de exercer uma quantidade significativa de torque em seu efetuador final, além disso sua estrutura deveria ser compacta o suficiente para que esse mesmo manipulador se deslocasse através de ambientes restritos. Atualmente já são empregados robôs na linha de montagem da empresa contratante, mas esses robôs são manipuladores industriais antropomórficos comuns, entretanto há problemas envolvidos com o tamanho desses manipuladores, dessa forma há uma necessidade da empresa em se possuir robôs com uma configuração não tão usual. Este trabalho de PFC está focado em dois aspectos: o primeiro na implementação da programação de movimentação do robô com alguns algoritmos de geração de trajetória e o segundo na construção de uma plataforma customizada para esta função. Esta plataforma ainda está em fase de construção, e implementa muitas funções de forma genérica, isso para que possa ser usada não apenas neste projeto.

Palavras-chave: Robótica. Robô. Simulação. Ambiente Virtual.

ABSTRACT

The SENAI Institute for Embedded Systems Innovation is an entity that offers solutions in several strategic segments such as aerospace, automotive, energy, oil and gas, automation and ICT industries. This PFC is part of a PoC project, Proof of Concept, which has been developed in SENAI. Due to a demand, the institute was responsible for the construction of a robotic manipulator, which should have several degrees of freedom and with the capacity to exert a significant amount of torque in its end effector, in addition its structure should be compact enough for that same handler to move through restricted environments. Robots are currently employed on the assembly line of the contracting company, but these robots are common anthropomorphic industrial manipulators, however there are problems involved with the size of these manipulators, so there is a need for the company to own robots with a not so usual configuration. This work of PFC is focused in two aspects: the first in the implementation of the programming of movement of the robot with some algorithms of generation of trajectory and the second in the construction of a platform customized for this function. This platform is still under construction, and implements many functions in a generic way, so that it can be used not only in this project.

Keywords: Robotics. Robot. Simulation. Virtual Environment.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

Sumário

1- INTRODUÇÃO

1.1 - Contextualização

Desde as civilizações antigas ouvem-se relatos de dispositivos automatizados configuráveis pelo usuário e até mesmo autômatos semelhantes a animais e seres humanos, projetados principalmente como entretenimento[1]. Com técnicas mecânicas desenvolvidas através da era industrial, surgiram aplicações mais práticas, como máquinas automáticas, controle remoto e controle remoto sem fio, com um objetivo especificamente voltado para a realização de trabalho[2]. Nos tempos atuais os robôs têm dominado a indústria e vem sendo utilizado como elemento chave da automação[1].

O ramo da tecnologia que lida com o projeto, construção, operação e aplicação de robôs, bem como sistemas de computador para seu controle, *feedback* sensorial e processamento de informações é robótica[3]. Essas tecnologias lidam com máquinas automatizadas que podem tomar o lugar dos seres humanos em ambientes perigosos ou processos repetitivos em ambiente de fabricação [5].

Um robô industrial é oficialmente definido pela ISO-10218 como um "*manipulador multipropósito controlado automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial*".

As concepções cinemáticas mais comuns de robôs utilizadas na indústria incluem desde robôs antropomórficos (o tipo mais comum), robôs tipo SCARA, e robôs cartesianos, também conhecidos como x-y-z[5]. No contexto da robótica geral, uma grande parte dos robôs industriais recebem a classificação de braços robóticos, inerente no uso da palavra "manipulador" mencionada na definição.

Na indústria, os robôs possuem uma autonomia diferenciada sob certos limites, muitas vezes ligada diretamente a sua função e aplicação. Alguns são programados

---

1 O termo robô vem de uma palavra tcheca, *robota*, que significa "trabalho forçado". Outras línguas eslavas também usam este radical em significado similar, por exemplo. Eslavo oriental bielorrusso e russo (rabynya), раб ucraniano (rab) e do sul eslavo bósnio, búlgaro, croata, macedônio, sérvio Роб (rob). Checo é uma língua eslava ocidental, como é eslovaco. Robota, em eslovaco, é usada apenas para significar trabalho.

para realizarem a mesma função de forma repetitiva (exaustivamente), com grande precisão dos movimentos, propriedade esta desejável em geral.

Para as tarefas repetitivas o robô possui funções ou rotinas que foram pré-programadas (programação *Off-line*) onde são especificadas a direção, aceleração, velocidade e etc.. Para as juntas do robô ou para os deslocamentos do efetuador final[7]. Outros, porém são mais flexíveis com relação a sua movimentação e/ou definem uma movimentação em tempo real (programação *on-line*), no caso de operações com objetos que precisem ser identificados a cada iteração.

A utilização do robô mostrou-se como um dispositivo ímpar na automação a partir da década de 1960[8]. Essas tecnologias estão levando a automação industrial para outra transição cujo escopo ainda é desconhecido [9].

Uma das grandes vantagens que contribuiu para que o uso de robótica nas indústrias cresça, certamente, foi devido ao custo desses equipamentos que vem declinando.

A Figura 1 mostra que, no decorrer da década de 1990, os custos de implantação de robôs diminuíram enquanto o custo da mão de obra humana aumentou [5].

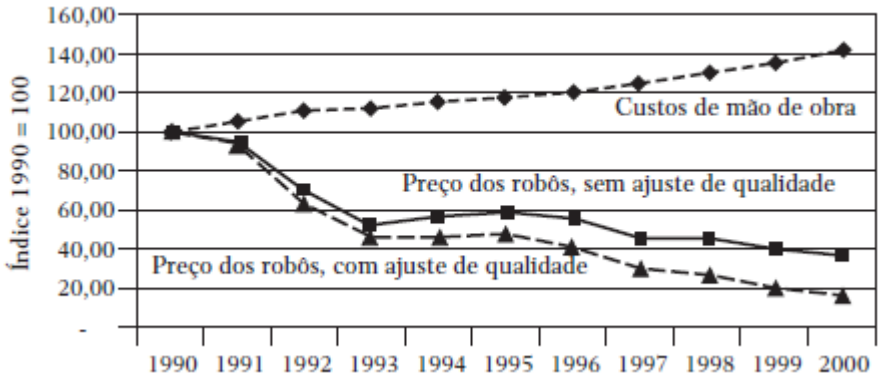


Figura 1 - Custo de mão de obra x custo robôs.

Fonte: [10]

Aliada a essa realidade, atualmente a robótica não está apenas ficando menos custosa, mas também muito mais eficiente. Cada vez mais os robôs se tornam mais rápidos, mais precisos e mais flexíveis. A medida que os robôs se tornam mais econômicos na execução de suas funções, ao passo que a mão de obra humana possui custo maior cada vez mais, observam-se que mais tarefas industriais se tornam

candidatas a automação via robótica. Essa é a principal tendência que vem incentivando o crescimento do mercado de robôs industriais.

1.2 - O Problema

O Instituto Serviço Nacional de Aprendizagem Industrial (SENAI) de Inovação recebeu a proposta de uma empresa multinacional para realizar o projeto e construção de um manipulador robótico redundante. Este robô atenderia não apenas um único processo na empresa, mas sim estaria integrado em muitas etapas de sua linha de montagem. Para tal, o robô deverá possuir as seguintes características:

- Ser robusto e confiável tanto em termos de repetibilidade e precisão,
- Suportar uma carga elevada[2] em seu efetuador final.

Atualmente na empresa cliente a realização de muitas tarefas é executada com auxílio de manipuladores robóticos antropomórficos, ou seja, já são robotizadas, mas esses robôs possuem uma dimensão exagerada para algumas tarefas, com isso, não conseguem executar alguns movimentos e atingir alguns pontos desejados na execução destas tarefas, sendo necessário, em alguns casos, utilizar mais de um manipulador para executar uma tarefa relativamente simples.

Na empresa as peças a serem manufaturadas em muitos casos são consideradas complexas e com espaço interno restringido, dificultando ou impedindo robôs convencionais de passarem por dentro para atingir um ponto do outro lado e efetuar o trabalho, dessa forma, há a necessidade de um novo robô para alcançar tal ponto.

Outro problema é que em alguns processos para realizar a fixação de peças é utilizada uma ferramenta relativamente grande quando comparadas as convencionais, e mesmo que o robô alcance a região interna das peças a ferramenta restringirá ainda mais o movimento. Neste caso, os robôs são posicionados em plataformas mais elevadas executando a tarefa de forma alternativa quando possível.

---

[2] A carga para o tipo de robô que foi desenvolvido para este projeto é estimada em 15 kg. Lembrando que este valor deve ser comparado entre robô do seu porte, pois há robôs como o ABB 6640 que possui um payload de 235 kg.

Portanto, o foco ao qual este projeto está voltado inclui tratar uma série de problemas, não necessariamente para linha de montagem, segurando, cortando e fixando peças, mas também, como por exemplo, para inspeção uma vez que atualmente é realizada de forma não automatizada, causado pela restrição estrutura da manufatura final.

1.3 - Objetivo Geral

Este projeto tem como objetivo geral:

Desenvolver um manipulador robótico de cadeia aberta que possa atuar na realização de múltiplas tarefas, na sua grande maioria em ambientes restritos e/ou obstruídos.

Como objetivos específicos tem-se:

- Desenvolver um sistema de interface para o operador, indicando de forma digital diversos eventos, configurar / carregar movimentações, controle manual.
- Desenvolver um sistema customizado, apresentando numa interface os indicadores desejados em um ambiente gráfico de fácil e rápida compreensão. Além de uma interface para o operador visualizar variáveis importantes para a operação.
- Desenvolver um sistema genérico que possa ser utilizado em outros manipuladores

Adiciona-se aos requisitos o fato de desejar-se que o sistema não seja específico para este caso, aceitando assim que outros equipamentos além de manipuladores robóticos sejam simulados e analisados.

1.4 - Escopo do trabalho

Neste projeto de fim de curso (PFC) será apresentado o desenvolvimento de um driver de comunicação baseada em um kit de desenvolvimento e um ambiente de testes virtual. O driver fará a ponte de toda a estrutura física, do robô construído, com a parte computacional, ou seja, o Ambiente de Testes. O ambiente de testes possibilitará a realização de toda a configuração dos motores, controle manual do robô, geração de trajetórias ou até mesmo carregar trajetórias previamente construídas,

bem como possuirá uma ferramenta de edição para manipuladores, dessa forma, o ambiente servirá para realizar os testes com o atual protótipo e ao mesmo tempo, será possível, realizar simulações para o robô final com oito graus de liberdade.

1.5 - Organização do Documento

O documento está divido em oito capítulos. O primeiro capítulo contextualizou o trabalho relacionado com o PFC e o problema a ser tratado. O Capítulo 2 apresenta uma breve descrição da empresa concedente, Instituto SENAI de Inovação e Sistemas Embarcados. O Capítulo 3 fornece a fundamentação teórica necessária que será abordada no desenvolvimento do projeto, apresentando informações sobre o funcionamento de motores elétricos que consistem na atuação de manipuladores robóticos, bem como uma descrição sobre o robô, tanto em se tratando de estrutura como programação, uma descrição breve sobre a comunicação utilizada para integrar as partes e a arquitetura de software.

O Capítulo 4 descreve o projeto de forma conceitual, sem especificar tecnologias ou ferramentas. Tratando também da metodologia utilizada. O Capítulo 5 mostra como foram selecionadas as tecnologias utilizadas no projeto. O Capítulo 6 trata do projeto implementado, apresentando o que foi feito, com detalhes sobre seu funcionamento e como foi feito.

O Capítulo 7 apresenta os resultados obtidos com a utilização do sistema desenvolvido e finalmente, o Capítulo 8 contém uma síntese sobre o que foi executado e os resultados obtidos frente aos objetivos do trabalho.

2 - A EMPRESA

O Serviço Nacional de Aprendizagem Industrial (SENAI) é uma instituição privada brasileira sem fins lucrativos, criada pelo decreto-lei 4.048 de 22 de janeiro de 1942. No ano de 2016 o Senai foi apontado como umas das principais instituições educacionais do hemisfério sul. A instituição tem como principal objetivo apoiar várias das áreas industriais do país, para isso, vem prestando diversos tipos de serviços técnicos, tecnológicos e educacionais auxiliando na capacitação e formação de recursos humanos. Entre os serviços tecnológicos pode-se citar, por exemplo, assessoria, consultoria, pesquisa aplicada, design, serviço laboratorial e etc. Na área da educação há diversos programas de capacitação profissional que são viabilizados por meio das modalidades de aprendizagem, habilitação, qualificação, aperfeiçoamento, formação técnica, superior e também pós-graduação, esses cursos são ministrados de forma presencial e/ou a distância. [11]

2.1 – SENAI Santa Catarina

Ao se dar início das atividades em Santa Catarina, uma grande pesquisa fora realizada, para que dessa forma fosse possível levantar a situação e as necessidades das indústrias naquela época.

Os resultados obtidos através da pesquisa concluíam que o estado possuía grande iniciativa industrial nos mais diversificados segmentos, porém, a maioria sem uma habilitação específica, dessa forma, eram executados sem formação apropriada chegando a casos onde não se havia formação alguma na área. A partir daí, foi dada a urgência em se criar uma matriz de competência que viesse a atender a demanda de necessidade do atual cenário da indústria catarinense, dando dessa forma oportunidade para a expansão industrial no estado.

No ano de 1944 começaram a funcionar em Florianópolis, Blumenau, Joinville e Siderópolis os núcleos do Senai. Foi em Joinville que os primeiros cursos começaram a ser ministrados, ainda de forma provisória, nas dependências da Escola Prática de Comércio Martins Veras e posteriormente no Círculo Operário, no bairro Bucarein. No ano seguinte, iniciou-se a construção da escola, inaugurada em 1946. Após dez anos da criação das primeiras unidades da entidade no estado, houve a o

desmembramento do Departamento Regional da 7ª Região do Senai, resultando na instalação oficial do SENAI/SC.

Atualmente, o Departamento Regional do Senai de Santa Catarina, na constante expansão de sua estrutura, conta com 63 unidades fixas, 23 unidades móveis, 565 salas de aulas e 923 laboratórios didáticos, sendo 193 laboratórios didáticos móveis, atuando em mais de 256 municípios do Estado.

Também integra três Institutos de Inovação em laser, sistemas embarcados e sistemas de manufatura e sete de Tecnologia, automação e tecnologia da informação e da comunicação; alimentos e bebidas; ambiental; eletroeletrônica; logística; materiais e têxtil, vestuário e design. [12]

2.2 – Institutos Senai de Inovação

Os Institutos de Inovação têm como objetivo principal o de aumentar a produtividade e a competitividade da indústria brasileira, criando e buscando soluções inovadoras para a indústria de grande, médio e pequeno porte. Todo o território nacional é atendido por esses institutos, que oferecem suporte a inovação com base tecnológica por meio dos seguintes fatores: [13]

- Pesquisa aplicada e projetos de inovação tecnológica;
- Apoio laboratorial para prototipagem e plantas-piloto (estágio pré-competitivo);
- Serviços tecnológicos de alta complexidade e alto valor agregado;
- Transferência tecnológica, aumento de performance, redução de riscos tecnológicos.
- Ecossistema de inovação para desenvolvimento de novos produtos, processos e tecnologias;
- Conexão com os principais atores do Sistema Nacional de Inovação;
- Consultoria e treinamento em diversas áreas tecnológicas.

Cada instituto é um ambiente de contínua interação entre a indústria, empreendedores, universidades, institutos de pesquisa e fontes de capital, em suas diversas formas. O resultado é a aceleração do fluxo de conhecimento científico e tecnológico orientado a resultados efetivos para o segmento industrial.

2.2.1 – Instituto SENAI de Inovação em Sistemas Embarcados

O Instituto SENAI de Inovação em Sistemas Embarcados (ISI), localizado atualmente no bairro Saco Grande as margens da rodovia SC 401 em Florianópolis, vem oferecendo soluções em diversos segmentos estratégicos como, por exemplo, indústrias aeroespacial, automotiva, de energia, óleo e gás, automação e tecnologias de informação e comunicação (TIC). A Figura 2 apresenta o logotipo do instituto.

Figura 2 – Logotipo do Instituto SENAI de Inovação Sistemas Embarcados.

Fonte: [13]

O ambiente de trabalho do instituto é bastante agradável e motivador por vários motivos. Um deles deve-se a equipe, atualmente com 33 colaboradores, extremamente competente e dedicada, não apenas com prazos, mas também com a qualidade e satisfação em cada projeto, não importando sua grandiosidade e complexidade. O time de profissionais está sempre disposto a auxiliar os colegas de trabalho, demonstrando um senso de companheirismo ímpar, o que se torna muito benéfico para um estagiário que não possui experiência profissional na área. Outro ponto importante é o incentivo dado por parte do instituto pela busca por aperfeiçoamento profissional e conhecimento, encorajando seus colaboradores a buscarem um maior grau de instrução como, por exemplo, um mestrado, ou até mesmo permitindo que os estagiários foquem nas atividades da graduação quando necessário.

2.2.2 – Robótica dentro do SENAI

Por se tratar de um instituto consideravelmente novo seus respectivos núcleos estão em fase de aperfeiçoamento ou mesmo criação com foco em áreas específicas.

Essas linhas de especializações, chamadas internamente de *core* (núcleo), são voltadas para tecnologia ainda ascendentes, que possam auxiliar no desenvolvimento da indústria nacional.

Dentro dos vários núcleos presentes no instituto, podemos citar um ramo que não é tão novo assim na indústria mundial, mas que na brasileira ainda precisa melhorar muito: a robótica.

Até recentemente, linhas automatizadas por robôs eram implantadas apenas por empresas de grande porte, como multinacionais, mas com o advento da indústria 4.0, cada vez mais empresas de pequeno e médio porte agregam às suas instalações robôs. Apesar disso, a média do uso de robôs na indústria ainda é muito baixa no Brasil, cerca de 1,5 mil robôs são instalados por ano[14]. Considerando este panorama, o SENAI busca colaborar com a indústria no sentido de incentivar cada vez mais o uso de tecnologia robótica para a automatização de linhas de fabricação.

O projeto descrito neste documento, apesar de não ser o primeiro na linha de pesquisa da robótica dentro do SENAI, é muito importante para o desenvolvimento do *core* no instituto, pois há um grande diferencial nele. A customização desde robô, tanto no sentido de estrutura como programação e controle fogem da realidade até agora enfrentada, dessa forma, ao mesmo tempo que o projeto se torna um grande desafio, também é uma fonte de conhecimento, até então, inigualável.

3 – FUNDAMENTAÇÃO TEÓRICA

Os tópicos que serão abordados neste capítulo de fundamentação teórica têm como objetivo apresentar os assuntos que foram estudados para o desenvolvimento do trabalho, ou então, que são importantes para uma melhor compreensão sobre o sistema desenvolvido, a ser apresentado no decorrer do texto nos capítulos posteriores.

Na seção 3.1 é apresentado o *Encoder*, descrevendo basicamente seu funcionamento e tipos. Este assunto é importante, pois, todo o controle de posicionamento do robô desenvolvido na execução do projeto faz uso desta tecnologia. A seção 3.2 contém uma explicação resumida do protocolo de comunicação, utilizado. A seção 3.3 apresenta uma rápida introdução na configuração de motores, explicando suas partes mais importantes e após isso, uma rápida abordagem sobre os servos motores, uma vez que o robô faz uso deste tipo de atuador. A seção 3.4 faz um pequeno apanhado sobre o tema robótica, apresentando algumas definições estruturais e matemáticas sobre manipuladores e uma rápida abordagem sobre geração de trajetórias, e a seção 3.5 apresenta alguns termos e utilizados na implementação do *software* que controlará o robô.

3.1 – *Encoders*

Um *encoder* rotativo, também chamado de codificador de eixo, é um dispositivo eletromecânico projetado para que seja possível converter a posição angular ou movimento de um eixo em um sinal analógico ou digital. [16]

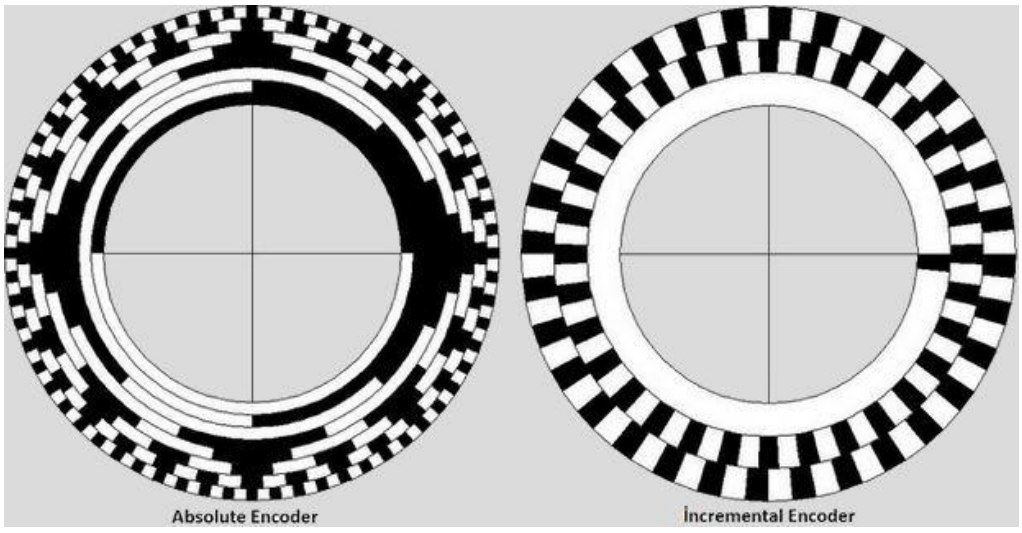
Existem dois tipos principais: absoluto e incremental (relativo). A saída de *encoders* absolutos indica a posição atual do eixo, tornando-os transdutores angulares ao contrário dos *encoders* incrementais que apenas fornecem informações sobre o movimento do eixo, essa normalmente é processada e assim obtemos a velocidade, distância e posição. [16]

Os *encoders* rotativos são usados em muitas aplicações que exigem rotação ilimitada e precisa do eixo - incluindo controles industriais, robótica, lentes fotográficas especiais[17], plataformas de radar rotativas entre outros.

3.1.1 – *Encoder* Absoluto e Incremental

Um *encoder* pode ser classificado em basicamente dois grupos, encoder absoluto e incremental.

Um *encoder* absoluto mantém informações de posição quando a energia é removida do sistema[18]. A posição do *encoder* está disponível imediatamente ao aplicar energia. A relação entre o valor e a posição física do maquinário controlado é definida na montagem; o sistema não precisa retornar a um ponto de calibração para manter a precisão da posição. A Figura 3a apresenta um exemplo de *encoder* absoluto. Como pode ser observado, cada posição do disco na imagem possui uma característica distinta, dessa forma é possível saber exatamente a posição mesmo que o sistema tenha acabado de ser energizado. O *encoder* absoluto (Figura 3a) possui múltiplos anéis de código com várias ponderações binárias que fornecem uma palavra de dados representando a posição absoluta do codificador dentro de uma revolução. Este tipo de codificador é frequentemente referido como um codificador absoluto paralelo[20]. Uma roda de alta resolução mede a rotação fracionária, e as rodas com código de menor resolução registram o número de revoluções inteiras do eixo [19].



(a) (b)

Figura 3 – (a) *Encoder* Absoluto a (b) *Encoder* Incremental.

Fonte: [15]

O *encoder* "incremental" registra com precisão as alterações na posição, mas não garante uma relação fixa entre o estado do codificador e a posição física. Dispositivos controlados de forma incremental, podem necessitar retornar até um ponto de referência fixo para inicializar a medição de posição. Isso se deve ao fato do seu funcionamento ser diferenciado, ele fornece apenas uma saída de pulsos como pode ser visualizados na Figura 4, esses pulsos chamados aqui de A e B não fornecem informações de suficientes de contagem por si só. O ponto onde a contagem começa depende do contador na eletrônica externa e não da posição do codificador. Para fornecer informações úteis sobre a posição, a posição do *encoder* deve ser referenciada ao dispositivo ao qual ela está conectada, geralmente usando um pulso de índice. A característica distintiva do codificador incremental é que ele relata uma mudança incremental na posição do codificador para a eletrônica de contagem [20].

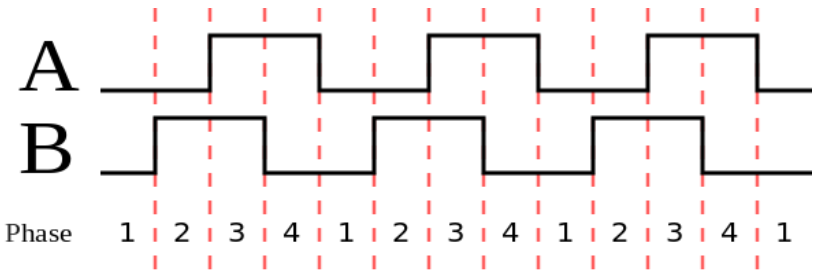


Figura 4 - Processo de Encoder Incremental.

Fonte: [15]

3.2 – Comunicação Serial

A comunicação serial era mais utilizada para grandes distâncias, pois era a mais apropriada para tal, podendo ser destacado como vantagens seu baixo custo, enquanto que a comunicação paralela era mais vantajosa em situações em que a transmissão de dados seria feita em curtas distâncias, devido ao fato de sua alta velocidade[21]. Conforme a tecnologia foi se desenvolvendo a comunicação do tipo paralela sofria com alguns problemas de dessincronização em relação ao tempo dos bits que eram enviados, ou seja, ao invés de chegarem juntos os bits eram recebidos de forma defasada, e com o problema de indução de tensão entre vários fios que se fazia uso esse tipo de protocolo de comunicação, essa indução alterava o nível logico do sinal inserindo no mesmo ruídos. Ao contrário da paralela, a comunicação serial

vem sendo melhorada, tanto em requisitos de confiança como velocidade, dessa forma sua aplicabilidade se tornou muito mais comum, tão comum que, por exemplo, hoje em dia é utilizada para realizar a comunicação entre diferentes componentes de computadores [22].

Na comunicação os dados são transmitidos de forma sequencial, bit a bit, fazendo uso de um único meio físico, podendo ser utilizado para sua comunicação fios mais longos e em menor quantidade. Existem vários critérios que podem ser usados para classificar uma interface serial, como por exemplo, o fluxo de dados. [22]

- **Simplex**: O fluxo de dados possui sentido único.
- **Half-Duplex**: O fluxo de dados ocorre nos dois sentidos, porém de forma alternada, ou seja, um mesmo canal recebe e transmite dados.
- **Full-Duplex**: O fluxo de dados ocorre nos dois sentidos e de forma simultânea. Este modo necessita de dois canais físicos, operando em modo simplex.

A interface pode também ser classificada conforme seu método de sincronização. Como toda comunicação serial requer que os dados sejam amostrados pelo receptor no momento correto, conforme a frequência e fase usada pelo transmissor. As formas mais comuns de classificar esta sincronização são: [22]

- **Síncrona**: Um sinal de *clock* é gerado pelo emissor e enviado pelo transmissor em separado ou junto dos dados. O receptor usa este sinal de *clock* para receber a mensagem corretamente.

- **Assíncrona**: Os *clocks* do transmissor e receptor são configurados com a mesma frequência de transmissão. A comunicação é realizada em caracteres, contendo um start *bit*, dados, um *bit* de paridade e um stop bit. Quando o receptor percebe o start bit, ele espera o tempo necessário para começar a ler os dados que estão sendo transmitidos no meio, desde que o desajuste de frequência seja inferior a um *bit*, não haverá problemas na comunicação.

- **Isócrona**: Utiliza frequência de envio pré-negociada e fixa. Os pacotes de transferência são enviados de forma contínua e não existe qualquer checagem dos dados. Usada para transmissão de grandes quantidades de dados.

A seguir serão apresentados padrões de comunicação mais disseminados no meio industrial como é o caso dos protocolos RS-232 e RS-485, e utilizados para comunicação entre circuitos integrados e dispositivos de pequeno porte.

3.2.1 – USB

O protocolo de comunicação USB (*Universal Serial Bus)* é um tipo de conexão *Plug and Play*, ou como o próprio nome sugere “ligar e usar”, que possibilita uma fácil comunicação entre os periféricos e o computador, podendo ser feita essa ligação sem a necessidade de desligar o PC. Como ele, é construído sobre um barramento que por possuir um conector específico e comum a todos os aparelhos que o usam, cria-se uma certa facilidade de instalação de dispositivos que aderem a essa tecnologia. [23]

Esse padrão foi apresentado por um grupo de vários fabricantes de periféricos, tendo uma boa aceitação no mercado. Tal aceitação se deve ao fato deste padrão apresentar grandes vantagens tais como a já citada *Plug and Play*, que facilita muito a vida dos usuários, mas também pode e deve ser mencionada a característica de *Hot-Switching* que significa que os periféricos podem ser adicionados ou removidos, por exemplo, dos computadores sem a necessidade de desligá-los antes. A grande maioria dos dispositivos podem ser alimentados diretamente pelo canal de comunicação, sem a necessidade de uma fonte própria de alimentação. Esse protocolo foi projetado de maneira que pudesse suportar vários periféricos ligados a um mesmo canal, ou porta USB. Dessa forma, é possível ligar até 127 dispositivos em apenas uma entrada, os quais não haveria qualquer problema, pois, o computador faria todo o gerenciamento das mensagens enviadas e recebidas. [23]

Outras características do protocolo USB: [24]

- Transferências isócronas e assíncronas;
- Baixo overhead do protocolo;
- Alimentação no mesmo cabo 5V @ 100-500 mA, dependendo do hub;

- Dispositivos entram no modo;
- Variedades de tamanhos de pacotes;
- Permite variação nas taxas de dados dos periféricos;
- O protocolo implementa o controle de fluxo.

A arquitetura do fluxo de informações através do barramento pode ser dividida em quatro tipos básicos de dados: [24]

- **Controle** (*control*): Que é utilizado para enviar e receber informações de forma estruturada, também possui correção e detecção de possíveis erros;
- **Volume** (Bulk): Utilizado para trocar informações de grandes volumes que não são estruturados. A detecção e correção de erros também se faz presente nesse tipo de transmissão;
- **Interrupção** (*Interrupt*): Utilizada para dispositivos que necessitem de uma transferência a taxas específica de dados. Possui detecção de erro e correção. Esse tipo de transmissão é geralmente usado em mouses e teclados;
- **Isócrona** (*Isochronous*): Esse tipo de transferência é usado para aplicações com grande fluxo de dados, onde a correção e detecção de erros não se tornam críticos como, por exemplo, é o caso de videoconferência.

3.2.2 – RS-485

Este protocolo é uma especificação de camada física de rede, muito utilizada para implementar redes industriais como PROFIBUS DP, ARCNET, INTERBUS e também é bastante comum em equipamentos de instrumentação, usando RS-485 associada a MODBUS. O padrão especifica que é possível fazer a ligação de até 32 dispositivos, ligação esta é realizada por um par de fios trançados, pelos quais ocorre transmissão diferencial, para evitar interferências de origem indutiva. Dessa forma conclui-se que se trata de uma comunicação assíncrona. [25]

O RS-485 suporta redes locais de baixo custo e links de comunicação multiponto, usando a mesma sinalização diferencial em par trançado do RS-422. Geralmente a transmissão RS-485 é realizada de forma *half-duplex*, com uma velocidade de 100kbps, isso para um comprimento máximo de cabo de 1200m, ou

então, 35Mbps com um cabo de 10m[26]. Como regra geral, a velocidade em bit/s multiplicada pelo comprimento em metros não deve exceder 108. Assim, um cabo de 50 metros não deve sinalizar mais rápido do que 2 MB/s [27].

O uso de resistores se faz necessário no caso de transmissões a longas distâncias, conectando o mesmo na extremidade do par trançado, chamado assim de resistor de terminação, os resistores são utilizados para evitar reflexo de sinal e em todos os casos é preciso usar resistores de *pull-up* e *pull-down* para definir o estado da rede quando nenhum dispositivo está transmitindo[25]. Desta forma, as linhas serão polarizadas para tensões conhecidas e os nós não interpretarão o ruído de linhas não alimentadas como dados reais; sem resistências de polarização, as linhas de dados flutuam de tal forma que a sensibilidade do ruído elétrico é maior quando todas as estações do dispositivo estão silenciosas ou sem alimentação. [28]

Diferentemente que o RS-422, que possui um circuito de acionamento que não pode ser desligado, os drivers RS-485 usam uma lógica de três estados, permitindo que os transmissores individuais sejam desativados. Isso permite que o RS-485 implemente topologias de barramento linear usando apenas dois fios. O equipamento localizado ao longo de um conjunto de fios RS-485 é chamado de nós, estações ou dispositivos[29]. A disposição recomendada dos fios é como uma série conectada de nós ponto-a-ponto (multiponto), isto é, uma linha ou barramento, não uma estrela, anel ou rede conectada multiplamente. Topologias de estrela e anel não são recomendadas devido a reflexões de sinal ou impedância de terminação excessivamente baixa ou alta. Se uma configuração em estrela for inevitável, estão disponíveis repetidores RS-485 especiais que escutam bidirecionalmente os dados em cada trecho e retransmitem os dados para todos os outros trechos.

3.3 – Motores

Um motor é uma máquina projetada para converter uma forma de energia em energia mecânica. Motores de calor queimam um combustível e dessa queima o calor é obtido, posteriormente usado para fazer o trabalho. Já motores elétricos transformam, ou convertem, a energia elétrica em movimento mecânico. Há motores pneumáticos que usam ar comprimido, motores de relojoaria que fazem uso de

energia elástica. Em sistemas biológicos, os motores moleculares, fazem uso de energia química para, por fim, criarem o movimento.

Um motor pode ser colocado em uma categoria de acordo com dois critérios: a forma de energia que ele aceita para criar movimento e o tipo de movimento que ele produz. Dessa forma podemos classificar os motores em macro categorias:

- Máquina a vapor;
- Motor Combustão Interna;
- Motor Combustão Externa;
- Motor de Ar Comprimido;
- Motor Elétrico;
- Motor Híbrido.

Como este trabalho trata do assunto da definição, construção e programação de um manipulador robótico será dada ênfase a motores elétricos, tendo em mente que o projeto aqui descrito terá em sua totalidade motores dessa natureza.

3.3.1 – Motores Elétricos

Um motor elétrico é uma máquina com a capacidade de converter energia elétrica em energia mecânica.

A maior parte dos motores elétricos realiza seu trabalho através da interação das correntes do enrolamento e campo magnético de um motor elétrico para gerar força. Há algumas aplicações onde os motores podem ser usados em reverso para converter energia mecânica em energia elétrica, um exemplo dessa aplicação seria no processo de frenagem regenerativa com motores de tração na indústria de transporte.

Podem ser encontrados nas mais diversas aplicações, podem ser alimentados por fontes de corrente contínua (DC), como baterias, veículos motorizados ou retificadores, ou por fontes de corrente alternada (AC), tais como a partir da rede elétrica, inversores ou geradores. Motores são de uso geral e com várias dimensões e características que, muitas delas altamente padronizadas, fornecem energia

mecânica conveniente para seu devido fim. Fim este que pode ser usado para produzir força linear ou rotativa (torque).

3.3.2 – Arquitetura dos Motores

3.2.2.1 – Rotor

É a parte móvel do motor elétrico, que gira para fornecer a potência mecânica a aplicação. Geralmente sua construção inclui condutores que transportam as correntes que interagem com o estator através de seu campo magnético para assim, gerar as forças que fazem o eixo se movimentar. No entanto, há rotores que possuem ímãs permanentes, sendo assim, o estator segura os condutores.

3.2.2.2 – Rolamentos

São os rolamentos que suportam o rotor, dessa forma permitem que ele gire em seu eixo. Os rolamentos são, por sua vez, suportados pelo invólucro do motor. O eixo do motor se estende pelos mancais para a parte externa do motor, onde a carga é aplicada. Como as forças da carga são exercidas além do mancal mais externo, diz-se que a carga está em balanço. [30]

3.2.2.3 – Estator

O estator é a parte estacionária do circuito eletromagnético do motor, geralmente construído de enrolamentos ou ímãs permanentes. Seu núcleo é composto de muitas chapas finas de metal, chamadas também de laminações. Para evitar as perdas de energia que resultariam de um núcleo sólido, são usadas as laminações.

3.2.2.4 – Intervalo de Ar

A distância entre o rotor e o estator é chamada de intervalo de ar ou abertura de ar. É a principal fonte do baixo fator de potência em que os motores operam. O espaço de ar tem efeitos importantes e é geralmente tão pequeno quanto possível. O

espaço de ar aumenta a corrente de magnetização necessária. Por esse motivo, o entreferro deve ser mínimo, porém, lacunas muito pequenas podem representar problemas mecânicos, além de ruído e perdas.

3.2.2.5 – Enrolamentos

Enrolamentos são fios que são colocados em bobinas, geralmente enrolados em torno de um núcleo magnético laminado de ferro macio, de modo a formar polos magnéticos quando energizados, esses polos de campo magnético possuem duas configurações básicas: saliente e não-saliente.

Na máquina de polosaliente, o campo magnético do polo é produzido por um enrolamento enrolado em torno do polo abaixo da face do polo. No maquinário não-polarizado ou de campo distribuído, ou rotor redondo, o enrolamento é distribuído em fendas de face de polo[31]. Um motor de polo sombreado tem um enrolamento em torno do polo fazendo com que sua fase atrase.

3.2.2.6 – Comutador

Um comutador consiste em segmentos de anéis coletores isolados uns dos outros e do eixo do motor elétrico, é utilizado para alternar a entrada da maioria das máquinas de corrente contínua e de algumas de corrente alternada. A corrente de armadura do motor é fornecida através das escovas estacionárias em contato com o comutador giratório, que causa a inversão de corrente necessária, e aplica energia à máquina de uma maneira ideal quando o rotor gira de um polo a outro[31,32]. O motor travaria, sem a inversão de corrente. Em vista dos avanços significativos nas últimas décadas, devido às tecnologias aprimoradas nos campos de controle eletrônico, controle sem sensor, motor de indução e motor de ímã permanente, os motores comutados eletromecanicamente estão sendo cada vez mais sendo substituídos por motores de indução comutada externamente e motores de imã permanente.

3.3.3 – Principais Categorias

Nos motores magnéticos, campos magnéticos são formados tanto no rotor quanto no estator. Uma força é gerada como produto da interação entre esses dois

campos, portanto, um torque no eixo do motor. Um ou ambos desses campos devem ser alterados de acordo com a rotação do motor. Isso é feito ligando e desligando os polos no momento certo, ou variando a força do polo.

Os principais tipos são motores de corrente contínua e motores de corrente alternada[33], sendo o primeiro cada vez mais substituído pelo segundo. Os motores elétricos CA são assíncronos ou síncronos[34]. Uma vez iniciado, um motor síncrono requer sincronismo com a velocidade síncrona do campo magnético móvel para todas as condições normais de torque. Em máquinas síncronas, o campo magnético deve ser fornecido por meios que não sejam de indução, como de enrolamentos excitados separadamente ou ímãs permanentes. A tabela 1 apresenta as principais características dos motores separadas por categorias.

3.3.4 – Motor Sem Escova CC

Alguns dos problemas presentes no motor de corrente contínua com escovas, são eliminados no design do motor de corrente contínua sem escovas (do inglês *brushless direct current*). O comutador mecânico neste caso é substituído por um comutador eletrônico externo sincronizado com a posição do rotor. Os motores BLDC são normalmente de 85 a 90% de eficiência ou mais. A eficiência de um motor BLDC de até 96,5% foi reportada[42], enquanto os motores DC com escovas normalmente são 75-80% eficientes.

A forma de onda trapezoidal característica do motor BLDC é derivada parcialmente dos enrolamentos do estator sendo uniformemente distribuídos e parcialmente da colocação dos PMs do rotor. Os enrolamentos do estator de motores BLDC trapezoidais podem ser monofásicos, bifásicos ou trifásicos e usam sensores de efeito Hall montados em seus enrolamentos para detecção de posição do rotor e baixo custo de controle do comutador eletrônico.

Esses motores são comumente usados em aplicações onde controle de velocidade se faz necessário, como em unidades de disco de computador ou em videocassetes, fusos em unidades de CD, CD-ROM, entre outros.

Tabela 1 - Principais Categorias de Motor Tipo de Comutação [35] [36] [37] [38] [39]				
Comutador interno			Comutador Externo	
Comutador Mecânico		Comutador eletrônico	Assíncrono	Síncrono
AC[40]	DC	AC	AC	
Motor universal (motor da série comutador CA[40] ou motor CA / CC[39]) * Motor de repulsão	Eletricamente motor DC excitado: * Separadamente Animado; * Séries; * Derivação; * Composto. Motor de PM DC	Com rotor PM: * Motor BLDC Com ferromagnético rotor: * SRM	Motores trifásicos: * SCIM * WRIM Motores de corrente alternada: * Capacitor * Resistência * Dividido Pólo sombreado	Motores trifásicos: * WRSM * PMSM ou Motor BLAC[41] - IPMSM - SPMSM * Híbrido Motores de corrente alternada: * Separação permanente capacitor * Histerese * Stepper * SyRM * Híbrido SyRM-PM
Eletrônica simples	Retificador, transistor linear ou helicóptero DC	Eletrônica mais complexa	Eletrônica mais complexa (VFD), quando fornecida	
AC	DC	AC	AC	
Comutador Mecânico	Comutador eletrônico	Assíncrono	Síncrono	
Comutador interno			Comutador Externo	

Sem um comutador para se desgastar, a vida de um motor sem escovas de corrente contínua pode ser significativamente mais longa em comparação com um motor DC usando escovas e um comutador. A comutação também tende a causar uma grande quantidade de ruído elétrico.

Outro ponto importante destes motores é a ausência de faíscas, ao contrário de motores escovados, tornando-os mais adequados para ambientes com produtos químicos voláteis e combustíveis. Além disso, faíscas geram ozônio, que pode se acumular em edifícios mal ventilados, com risco de prejudicar a saúde dos ocupantes, também são motores acusticamente muito silenciosos, o que é uma vantagem se usado em equipamentos que são afetados por vibrações.

3.3.5 – Servo Motores

Um servo motor é um atuador rotativo ou atuador linear que permite o controle preciso da posição angular ou linear, velocidade e aceleração. Consiste basicamente em um motor acoplado a um sensor de posicionamento obter o *feedback* de posição. Também requer um controlador, que dependendo da aplicação que foi designada para o servo, relativamente sofisticado, geralmente um módulo dedicado projetado especificamente para sua aplicação.

Os servo motores são usados em aplicações como robótica, maquinário de comandos numéricos computadorizados (CNC) ou manufatura automatizada e etc.

3.3.5.1 – Mecanismo

Um servo motor é um servomecanismo de circuito fechado que usa realimentação, ou *feedback*, de posição para controlar seu movimento e posição final. A entrada para o seu controle é um sinal (analógico ou digital) representando a posição comandada para o eixo de saída.

O motor é acoplado com algum tipo de *encoder* para fornecer *feedback* de posição e velocidade, em alguns casos, apenas a posição é medida. A posição medida da saída é comparada com a posição de comando, a entrada externa para o controlador. Caso a posição de saída seja diferente da requerida, é então gerado um sinal de erro, dessa forma o motor continua girando seja qual for a direção, assim na

medida que as posições se aproximam, o sinal de erro é reduzido a zero e o motor atinge a posição desejada.

Os servos motores mais simples usam sensoriamento somente de posição através de um potenciômetro e controle chamado de bang-bang de seu motor; o motor sempre gira a toda velocidade (ou está parado). Este tipo de servo motor não é amplamente usado no controle de movimento industrial, mas forma a base dos servos simples e baratos usados para modelos controlados por rádio. Por outro lado, os servos motores mais sofisticados usam codificadores rotativos ópticos para medir a velocidade do eixo de saída[43] e um inversor de velocidade variável para controlar a velocidade do motor[44]. Ambos os aprimoramentos, geralmente em combinação com um algoritmo de controle proporcional integral derivativo (PID), permitem que o servo motor seja levado à sua posição comandada com mais rapidez e precisão, com menos *overshooting*. [45]

3.4 – Robótica

Robótica é um ramo educacional e tecnológico[47,49] que engloba não apenas robôs, mas também computadores e sistemas de computação, que por sua vez, tratam de sistemas compostos por partes mecânicas automáticas e controladas por circuitos integrados, tornando sistemas mecânicos motorizados, controlados manualmente ou automaticamente por circuitos eléctricos. Os robôs são amplamente utilizados na fabricação, montagem, embalagem e empacotamento, mineração, transporte, exploração de terra e espaço, cirurgia, armamento, pesquisa de laboratório, segurança e produção em massa de bens de consumo e industriais.[48,50]

Neste capítulo serão abordados alguns assuntos relacionados à robótica que, não necessariamente, foram utilizados de forma direta na execução deste trabalho, mas que se fazem necessários para um melhor entendimento do assunto abordado.

3.4.1 - Descrição Elo

Um manipulador robótico pode ser considerado basicamente como um conjunto de corpos conectados em cadeia, por juntas, corpos esses que são chamados de elos [50]. As juntas formam uma conexão entre um par de elos vizinhos.

A Figura 5 mostra os seis tipos possíveis de juntas. A grande maioria dos manipuladores são construídos com juntas rotacionais ou possuem juntas de deslocamento linear, chamadas de juntas prismáticas.

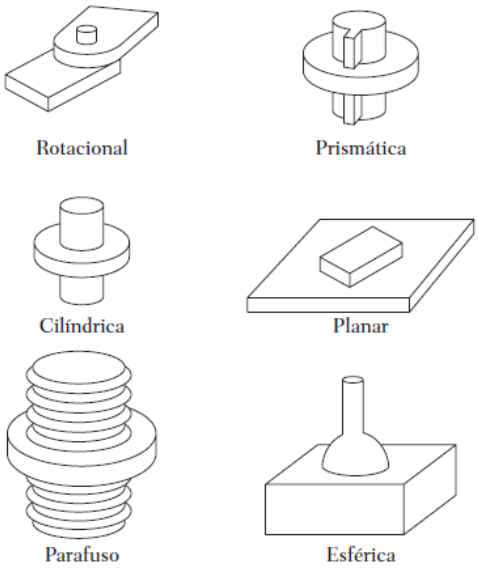


Figura 5 - Tipo de Elos.

Fonte: [50]

Todos os elos são enumerados partindo da base fixa do manipulador que é geralmente chamado de elo 0. Após esse elo, temos o elo 1, que seria o primeiro corpo móvel do manipulador, e assim segue até o fim da cadeia robótica, onde em geral tem-se o efetuador.

A fim de posicionar um efetuador, em geral no espaço tridimensional, é necessário um mínimo de seis juntas. O que intuitivamente faz sentido, pois, para a descrição de um objeto no espaço é necessária possuir três parâmetros para a posição e mais três para a orientação. Comumente os manipuladores possuem entre três ou seis juntas. É importante mencionar que alguns robôs não apresentam uma cadeia de elos dispostos em ordem serial, eles possuem ligações em paralelogramos ou outras estruturas cinemáticas fechadas, caracterizando os robôs chamados paralelos[50]. A Figura 6, apresenta um manipulador paralelo com essa configuração mais complexa.

Figura 6 - Robô paralelo com uma configuração de cadeia fechada.

Fonte:[71]

Resumidamente, para o objetivo de se obter as equações de cinemática do mecanismo, um elo é considerado apenas um corpo rígido que define a relação entre os eixos de duas juntas, vizinhas, de um manipulador [50].

3.4.2 – Parâmetros de Denavit-Hartenberg

Há vários métodos para realizar a descrição cinemática de mecanismos e robôs, mas neste trabalho a metodologia usada será a convenção de Denavit-Hartenberg (DH). Esta convenção é base para o modelamento cinemático por meio parâmetros para cada junta e a definição de uma matriz homogênea [6].

Os parâmetros de Denavit-Hartenberg são quatro parâmetros associados a uma convenção para fixar sistemas de referência a cada elo de uma cadeia cinemática, ou manipulador robótico. Dessa forma, qualquer robô pode ser descrito cinematicamente apenas atribuindo esses valores. Dois deles descrevem o elo em si e os outros dois descrevem a conexão do elo com um elo adjacente.

Nesta convenção, são fixados sistema de coordenadas a cada par de elos, dessa forma, uma transformação é associada a articulação [Z], e a segunda transformação é associada ao elo (X). As transformações de coordenadas ao longo de um robô em série consistindo de n elos resulta nas equações cinemáticas do robô,[51]

$$[T] = [Z_1][X_1][Z_2][X_2] \dots [Z_n][X_{n-1}] \quad (1)$$

onde [T] é a transformação responsável por localizar o elo final.

As articulações conectando os elos são representadas como juntas rotacionais ou prismáticas, cada qual possui uma linha única S no espaço que constitui o eixo da articulação e define o movimento relativo de ambos os elos, tudo isso para determinar as transformações de coordenadas [Z] e [X]. Para cada sequência de linhas S_i e S_{i+1}, existe uma linha normal de $A_{i,i+1}$.[51]

Com a convenção de Denavit-Hartenberg é possível a definição do deslocamento relativo entre elos em torno de um eixo da articulação comum S_i por deslocamento rotacional. A equação (2) apresenta uma matriz homogênea [6] para uma rotação e deslocamento linear.

$$[\mathrm{Z_i}] = \begin{bmatrix} \cos\theta_\mathrm{i} & -\mathrm{sen}\theta_\mathrm{i} & 0 & 0 \\ \mathrm{sen}\theta_\mathrm{i} & \cos\theta_\mathrm{i} & 0 & 0 \\ 0 & 0 & 1 & \mathrm{d_i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

onde θ_i é a rotação e d_i é o deslocamento linear ao longo do eixo Z.

Importante lembrar que dependendo da estrutura do manipulador, um destes parâmetros pode vir a ser constante. Nesta convenção, as dimensões de cada elo na ligação em série são definidas pelo deslocamento rotacional em torno da normal comum $A_{i,i+1}$ a partir da junta S_i até S_{i+1}, que é dado pela matriz homogênea da equação 3

$$[\mathrm{X_i}] = \begin{bmatrix} 1 & 0 & 0 & \mathrm{r_{i,i+1}} \\ 0 & \cos\alpha_{\mathrm{i,i+1}} & -\mathrm{sen}\alpha_{\mathrm{i,i+1}} & 0 \\ 0 & \mathrm{sen}\alpha_{\mathrm{i,i+1}} & \cos\alpha_{\mathrm{i,i+1}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

onde $\alpha_{i,i+1}$ e $i_{i,i+1}$ definem as dimensões físicas do elo em termos do ângulo d da distância ao longo do eixo x.[51]

Resumidamente

1. O eixo z fica na direção do eixo da junta;
2. O eixo x é paralelo à normal comum $x_n = z_n x z_{n-1}$;
3. Se não há uma normal comum única então d é um parâmetro livre. A direção de x_n vai de z_{n-1} para z_n;

4. O eixo y é escolhido depois de x e z de maneira que forme um sistema de coordenadas.

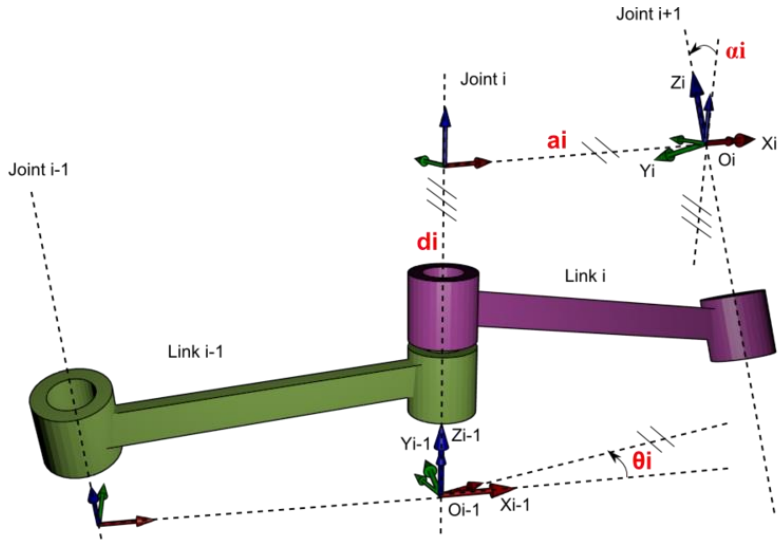


Figura 7 - Parâmetros Denavit-Hartenberg.

Fonte: [51]

Os quatro parâmetros de Denavit-Hartenberg são [52]:

- d: distância ao longo do z anterior até a normal comum;
- θ: ângulo em torno do z anterior, do x anterior até o x;
- r: comprimento da normal comum (ou seja, a, mas usando essa notação cuidado para não confundir com α). No caso de uma junta rotacional, este é o raio ao redor do z anterior;
- α: ângulo em torno da normal comum, do z anterior ao z novo.

A matriz de Denavit-Hartenberg resulta em:

$$\left[{}^{n-1}\mathrm{T}_{\mathrm{n}}\right] = \begin{bmatrix} \cos\theta_n & -\mathrm{sen}\theta_n\cos\alpha_n & \mathrm{sen}\theta_n\mathrm{sen}\alpha_n & r_n\cos\theta_n \\ \cos\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\mathrm{sen}\alpha_n & r_n\mathrm{sen}\theta_n \\ 0 & \mathrm{sen}\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Dessa forma separamos o deslocamento e rotação, como podemos observar na matriz homogênea da equação (5), onde R é a sub-matriz 3x3 que descreve rotação e T é a sub-matriz 3x1 que descreve a translação [53] [54].

$$\left[{}^{n-1}T_n\right] = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \quad (5)$$

3.4.3 – Cinemática Direta

A cinemática direta refere-se ao uso das equações cinemáticas de um robô para calcular a posição do efetuador a partir de valores especificados para os parâmetros da articulação[55]. Geralmente, um sistema de referência é fixado no efetuador final, mais precisamente, na ferramenta e outro sistema de referência é fixado também ao último elo do manipulador (elo n), dessa forma a matriz homogênea que irá descrever a posição e rotação entre estes dois sistemas de coordenadas será constante. Da mesma forma, determina-se um sistema de referência utilizado na base para estabelecer a localização da tarefa a ser executada. Essa referência geralmente tem um deslocamento constante em sua posição em relação sistema 0, que também é fixado na base. [55]

O problema mais comum da cinemática direta é encontrar a posição e a orientação relativa desejadas entre dois sistemas de referência, geralmente entre a base e o efetuador final, dada uma estrutura geométrica do mecanismo robótico e valores de posições das articulações na mesma quantidade que graus de liberdade do mecanismo. Esse problema da cinemática direta é crítico para o desenvolvimento de algoritmos de coordenação de manipulador, pois as posições das juntas são, geralmente, medidas por sensores embarcados nas juntas e se faz necessário calcular as posições dos eixos das juntas em relação à estrutura fixa.

Na prática, o problema da cinemática direta é resolvido calculando a transformação entre uma estrutura de coordenadas fixada no efetor e outra no sistema de referência fixado na base. Isso é bastante simples para uma cadeia aberta, pois, a transformação que descreve a posição do efetor em relação à base é obtida através da concatenação de transformações entre as referências fixadas nos elos adjacentes da cadeia.

As transformações homogêneas fornecem uma notação compacta, mas ainda assim são ineficientes em se tratando de resolver o problema da cinemática direta computacionalmente. Uma simplificação na computação pode ser obtida separando posição de orientação na transformação para eliminar, dessa forma, todas as multiplicações pelos elementos 0 e 1 das matrizes.

3.4.4 – Cinemática Inversa

A cinemática inversa utiliza as equações cinemáticas para determinar os parâmetros da junta que fornecem a posição desejada para cada um dos elos do robô [56]. A especificação do movimento de um robô de modo que seus efetuadores atinjam as coordenadas desejadas é conhecida como geração de trajetória. A cinemática inversa transforma o movimento em trajetórias para cada atuador do robô. As equações cinemáticas, definem o movimento e a configuração da cadeia em termos de seus parâmetros conjuntos. A cinemática direta usa os parâmetros da articulação para calcular a configuração da cadeia, e a cinemática inversa inverte este cálculo para determinar os parâmetros da articulação que atingem uma configuração desejada. [57-59]

No caso comum de um manipulador de cadeia aberta com seis graus de liberdade, o problema da cinemática inversa requer uma solução de conjuntos de equações não-lineares. Neste caso, de um manipulador de seis graus de liberdade, três dessas equações devem se referir ao vetor de posição dentro da transformação homogênea, sendo que as outras três, se relacionam com a matriz de rotação. No último caso, essas três equações não podem vir da mesma linha ou coluna devido à dependência dentro da matriz de rotação. Por ser um sistema de equações não-lineares, é possível que não existam soluções ou existam soluções múltiplas[59]. Para que exista uma solução, a posição e a orientação desejadas do efetuador devem estar no espaço de trabalho do manipulador. Nos casos em que as cadeias cinemáticas sejam complexas ou que ocorra alta relação não linear entre os parâmetros do robô, soluções analíticas são difíceis de serem alcançadas de forma fechada, assim métodos numéricos são necessários [59].

3.4.5 – Espaço de Trabalho

O espaço de trabalho de um robô manipulador é frequentemente definido como o conjunto de pontos que podem ser alcançados pelo seu efetuador[60] ou, em outras palavras, é o espaço no qual o robô trabalha e pode ser um espaço cartesiano ou tridimensional (volume 3D) ou um espaço plano (área sobre uma superfície 2D). A área de trabalho depende diretamente da arquitetura do robô, ou seja, quanto mais juntas e articulações maior destreza terá este robô e maior será seu volume de

trabalho, no sentido de mais completa, não necessariamente significa que a área será maior. Outro fator que está ligado a construção do robô é a quantidade de pontos dentro do seu espaço de trabalho que pode ser atingido por apenas uma única configuração. A figura 8 apresenta alguns exemplos de espaços de trabalho de alguns robôs.

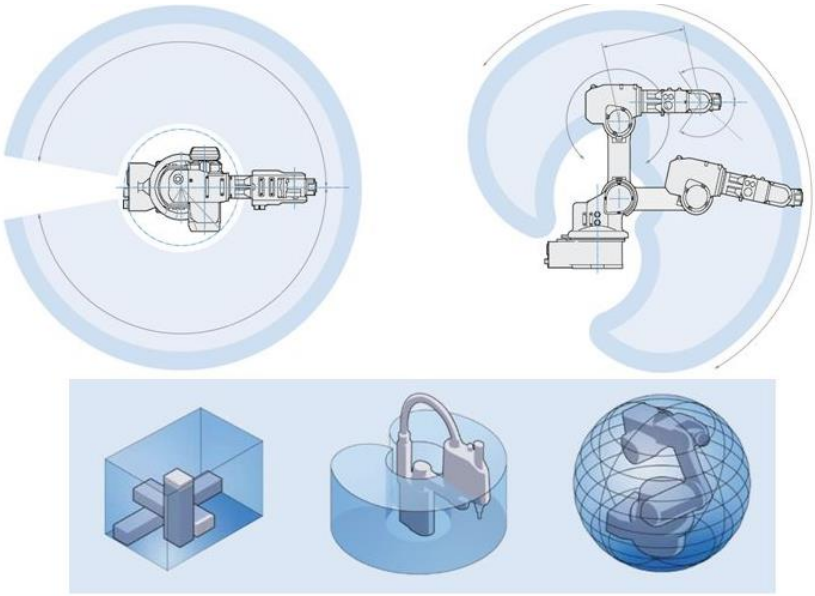


Figura 8 – Representação do espaço de trabalho da alguns robôs.

Fonte: [71]

3.4.6 – Geração de Trajetória

Nesta seção será apresentada alguns métodos e ideias para criar e computar uma trajetória que descreverá um movimento desejado de um manipulador. No contexto deste trabalho, a trajetória pode ser interpretada como sendo um conjunto de posição, velocidade e aceleração desejadas para o efetuador em função do tempo, para cada instante de tempo.

A resolução deste problema dedica-se a como especificar uma trajetória para o manipulador através de uma interface, a fim de tornar essa movimentação amigável e de fácil entendimento para o usuário. Pois, o operador do robô não necessariamente precisa desenvolver funções e algoritmos complicados para tal ação, e sim, com

alguns comandos simples e intuitivos consiga realizar uma implementação de trajetória com o tempo e velocidades desejadas.

Geralmente quando especificamos trajetórias, consideramos os movimentos do efetuador final, ou ferramenta, através do sistema de referência da ferramenta em relação ao sistema de referência da base do manipulador. Dessa forma, desacoplamos toda a descrição do movimento gerado por elos intermediários, isto de certa forma acaba sendo benéfico, pois, há uma generalização do uso da mesma trajetória, que poderá ser aplicada em manipuladores, até mesmo com configurações diferentes, ou então no mesmo manipulador, mas com ferramentas diferentes. A figura 9 apresenta o deslocamento de objeto realizado por um robô entre dois pontos distintos. Entre estes dois pontos uma trajetória é definida.

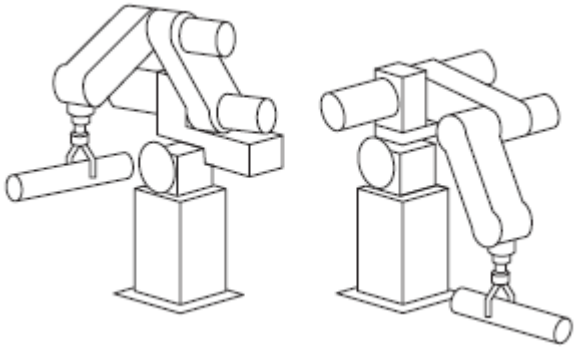


Figura 9 - Manipulador executando trajetória.

Fonte: [50],

A Figura 9 apresenta o problema mais básico, que seria de movimentar o manipulador para uma posição final, ou desejada, partindo de uma posição inicial, ou seja, nossa necessidade é mover o sistema de referência da ferramenta do ponto inicial para o final. Em alguns casos há também a necessidade de uma nova orientação e dessa forma a trajetória envolve um pouco mais de detalhes para que seja possível no fim da movimentação o manipulador expressar uma configuração, igualmente desejada. A Figura 10 apresenta diferentes configurações para um manipulador robótico atingir o mesmo ponto espacial.

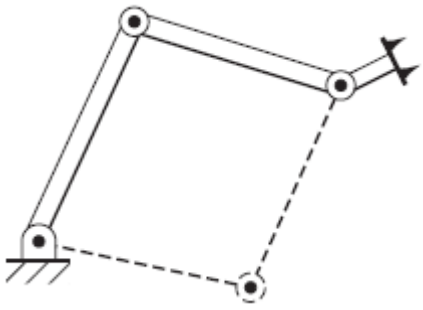


Figura 10 – Diferentes poses para um mesmo ponto.

Fonte: [50] pag 97

Quanto mais complexo é a deslocamento desejado, mais detalhes são exigidos para que dessa forma, o robô se mova da forma mais próxima do esperado. Uma maneira de incluir mais detalhes em uma descrição de trajetória e fornecer uma sequência de pontos de passagem desejados (pontos intermediários entre a posição inicial e a posição final). Assim, para completar o movimento, a ferramenta deve passar por um conjunto de posições e orientações intermediárias, conforme descritas pelos pontos de passagem[50]. Em cada ponto de passagem, é possível agregar várias configurações, como velocidade, aceleração, tempo para atingir este ponto e etc. Cada um desses pontos, é na verdade uma referência em relação a base. Essa forma de criação de trajetória é muito utilizada para robôs que operam em espaços com muitas restrições, como obstáculos, mas mesmo assim são muito bem conhecidos, dessa forma a trajetória do robô pode ser planejada de forma a contornar todos essas restrições, como mostrado na Figura 11.

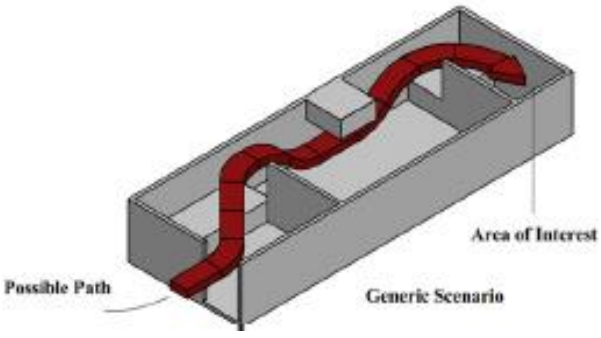


Figura 11 - Robô entrando em ambiente constrangido.

Fonte: [63]

Robôs autônomos, ou que trabalham em ambientes totalmente desconhecidos, precisam de sensores para reconhecer o terreno e só então decidir o caminho, ou seja, a trajetória é criada em tempo real [59]

Em geral, é desejável que o movimento do manipulador seja suave. Muitas vezes funções com a primeira derivada é contínua se comportam de maneira suave, mas em alguns casos a segunda derivada também é aceitável. Movimentos bruscos, com solavancos, tendem a causar um maior desgaste do mecanismo e vibrações no manipulador, bem como na ferramenta. [50]

Os pontos da trajetória, geralmente possuem parâmetros de configuração de uma posição e uma orientação especificada para o sistema de referência do manipulador final, em relação ao sistema da base do robô. Após serem passados esses pontos ao, por exemplo, algoritmo de geração de trajetória, eles são “convertidos” em ângulos para cada junta robótica através da cinemática inversa. Dessa forma, é possível criar uma função que suavize os movimentos das juntas do robô e um tempo é atribuído a cada intervalo entre dois pontos, assim, o algoritmo garante que todas as juntas alcancem os pontos ao mesmo tempo e com isso resultando na posição cartesiana desejada em cada ponto de passagem.

3.4.3.1 - Polinômios Cúbicos

Considere agora movimentar o manipulador de uma posição inicial para uma final. O que precisamos é de uma função suporte para cada deslocamento de junta cujo valor em t_0 seja sua posição inicial é cujo valor em t_f seja a posição final desejada. A Figura 12, mostra que essa movimentação pode ser realizada de várias formas.

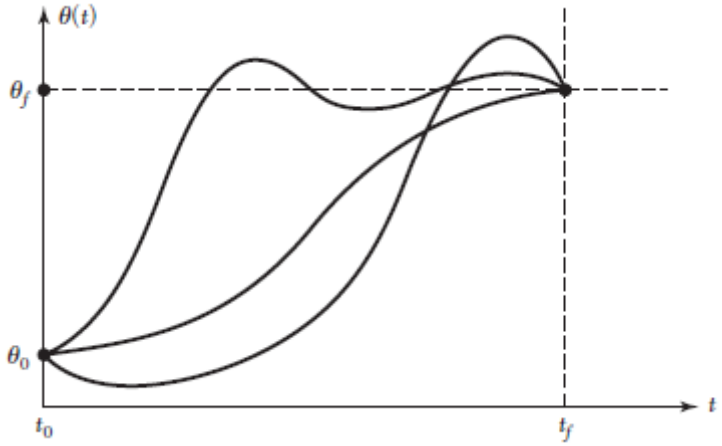


Figura 12 – Várias possibilidades para a mesma movimentação.

Fonte: [50]

Em um único movimento temos, pelo menos, quatro parâmetros ou restrições que são evidentes,

$\theta(0) = \theta_0$, posição inicial
$\theta(t_f) = \theta_f$, posição final
$\dot{\theta}(0) = 0$, velocidade inicial
$\dot{\theta}(t_f) = 0$, velocidade final

Essas restrições, que pertencem a um movimento que possui como velocidade inicial e final igual a zero, podem ser satisfeitas com um polinômio de, pelo menos, terceiro grau, pois possui quatro coeficientes, como mostrado na equação 6.

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (6)$$

de forma que os polinômios referentes a velocidade e aceleração são respectivamente,

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (7)$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \quad (8)$$

Combinando as equações 6, 7 e 8 com as quatro restrições obtemos um sistema com quatro equações e quatro incógnitas,

$$\theta_0 = a_0 \quad (9)$$

$$\theta_f = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (10)$$

$$0 = a_1 \quad (11)$$

$$0 = a_1 + 2a_2 t + 3a_3 t^2 \quad (12)$$

Resolvendo o sistema encontramos os seguintes valores para a_i

$$a_0 = \theta_0 \quad (13)$$

$$a_1 = 0 \quad (14)$$

$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) \quad (15)$$

$$a_3 = \frac{2}{t_f^3}(\theta_f - \theta_0) \quad (16)$$

Com esses valores é possível calcular o polinômio que dá suporte e que conectará a posição inicial com a final desejada para cada junta. Lembrando que esta solução é para uma movimentação que se inicia do repouso e termina no repouso, ou seja, parte com velocidade zero e ao chegar na posição final sua velocidade novamente se igualará a zero. Outras possibilidades de trajetórias podem ser encontradas em [6].

3.4.3.2 - Polinômios Cúbicos com pontos de passagem

Consideremos agora um conjunto de pontos ordenados onde deseja-se passar com o efetuador através de pontos intermediários (pontos de passagem) sem parar, desta forma as restrições precisam ser alteradas. No caso acima, as restrições tanto de velocidade inicial como final eram iguais a zero, portanto, basta alterar a velocidade final com o valor desejado da velocidade que se deseja atravessar o ponto de passagem. Lembrando que desta forma a velocidade inicial do ponto de passagem seguinte será igual ao da velocidade final do ponto atual.

Tome para este caso que o manipulador se encontra em estado de repouso, portanto, sua velocidade inicial é igual a zero, e sua velocidade ao atingir $P1$ seja igual a $30\,graus/s$ e ao atingir $P2$ sua velocidade deve ser de $15\,graus/s$, então as restrições seriam as seguintes,

P1	P2	P3
$\theta(0) = \theta_0$	$\theta(0) = \theta_{P1}$	$\theta(0) = \theta_{p2}$
$\theta(t_f) = \theta_{P1}$	$\theta(t_f) = \theta_{p2}$	$\theta(t_f) = \theta_{p3}$
$\dot{\theta}(0) = 0$	$\dot{\theta}(0) = 30$	$\dot{\theta}(0) = 15$
$\dot{\theta}(t_f) = 30$	$\dot{\theta}(t_f) = 15$	$\dot{\theta}(t_f) = \cdots$

Se tivermos as velocidades das juntas desejadas em cada ponto de passagem, basta aplicarmos as seguintes restrições,

$\theta(0) = \theta_0$, posição inicial
$\theta(t_f) = \theta_f$, posição final
$\dot{\theta}(0) = \dot{\theta}_0$, velocidade inicial
$\dot{\theta}(t_f) = \dot{\theta}_f$, velocidade final

Dessa forma as quatro equações que descrevem o polinômio são

$$\theta_0 = a_0 \quad (17)$$
$$\theta_f = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (18)$$
$$\dot{\theta}_0 = a_1 \quad (19)$$
$$\dot{\theta}_f = a_1 + 2a_2 t + 3a_3 t^2 \quad (20)$$

Encontrando os a_i, obtemos

$$a_0 = \theta_0 \quad (21)$$
$$a_1 = \dot{\theta}_0 \quad (22)$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f \quad (23)$$
$$a_3 = \frac{-2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0) \quad (24)$$

Além dos polinômios, outra opção para criar uma trajetória é a linear. Neste método interpolamos linearmente a posição inicial com a final, mas dessa forma, embora o movimento de cada junta seja linear o manipulador final, não

necessariamente irá se movimentar em linha reta. Isso se deve ao fato de que o método linear insere descontinuidades da velocidade no início e fim do movimento. Para contornar esse problema podem ser adicionados uma região de combinação parabólica em cada ponto de passagem. A Figura 13 demonstra essa combinação.

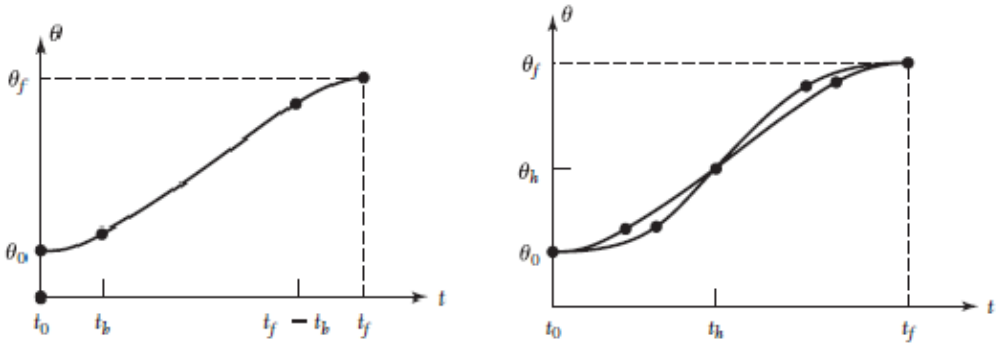


Figura 13 - Interpolação Linear com Combinações Parabólicas.

Fonte: [50] modificada.

No intervalo da combinação, a velocidade é alterada de forma suave, devido ao fato da aceleração constante. Portanto, na função linear foram agregadas essas duas funções parabólicas, formando assim um percurso contínuo em relação a velocidade e posição.

Como podemos ver na Figura 13, há muitas soluções para o problema, mas a resposta é sempre simétrica, em relação ao ponto médio do gráfico. A velocidade ao final da região combinatória deve ser igual da seção linear e dessa forma,

$$\dot{\theta}t_b = \frac{\theta_h - \theta_b}{t_h - t_b} \tag{25}$$

onde θ_bé o valor da posição no fim da região da combinação entre as funções linear e parabólica e $\ddot{\theta}$ é a aceleração aplicada nesta mesma região. Como θ_b é dado por

$$\theta_b = \theta_o + \frac{\acute{\theta}t_b^{\,2}}{2} \tag{26}$$

Combinando as equações 25 e 26, juntamente com o tempo de duração do movimento $t = 2t_h$ temos,

$$\acute{\theta}t_b^{\,2} - \acute{\theta}t_b + (\theta_f - \theta_o) = 0 \tag{27}$$

Geralmente escolhendo uma aceleração alta o suficiente, que respeite a seguinte restrição

$$\acute{\theta} \geq \frac{4(\theta_f - \theta_o)}{t^2} \tag{28}$$

podemos resolver a equação acima para encontrar o valor de t_b

$$t_b = \frac{t}{2} - \sqrt{\frac{\acute{\theta}^2 t^2 - 4\acute{\theta}(\theta_f - \theta_o)}{2\acute{\theta}}} \tag{29}$$

Com o valor da aceleração sendo igual a infinito a região combinatória torna-se nula e voltamos ao caso de uma interpolação linear simples. A medida que diminuímos o valor, a região combinatória se torna maior até que a trajetória não possua mais região linear.

Da mesma forma que com polinômios, as funções lineares com combinações parabólicas também podem ser utilizadas para realizar a interpolação de vários pontos de passagem. A Figura 14, mostra um conjunto de pontos que foram interpolados com funções lineares, mas que em cada ponto de passagem combinações parabólicas auxiliaram para suavizar as curvas.

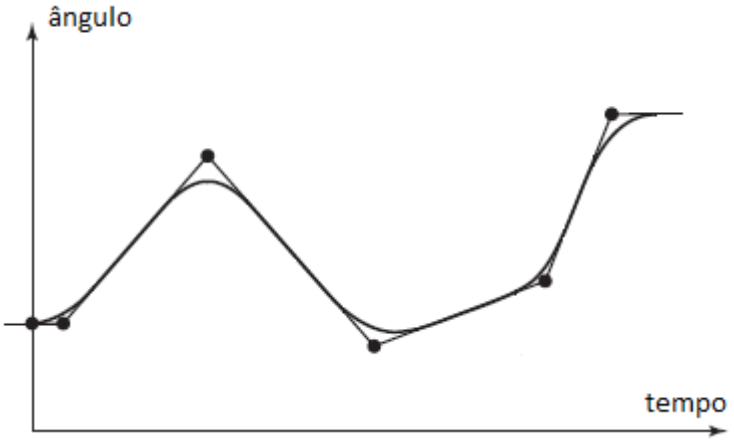


Figura 14 - Pontos de passagem interpolados

Considere agora as seguintes suposições

- os três pontos intermediários como sendo $\theta_1, \theta_2, \theta_3$;
- a duração da combinação do ponto 2 é t_2;
- a duração da linearidade entre 1 e 2 é t_{12};

- a duração entre a conectividade dos pontos 1 e 2 é $t_{total12}$;
- a velocidade na linearidade entre 1 e 2 é $\ddot{\theta}_{12}$;

Essas considerações podem ser visualizadas na Figura 15,

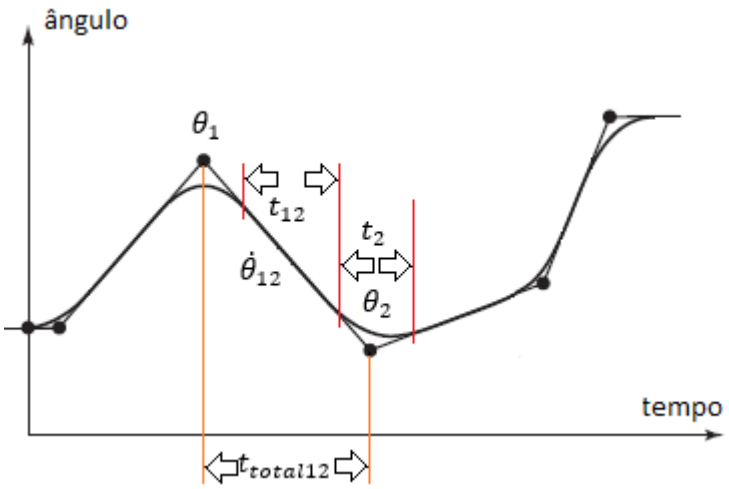


Figura 15 - Pontos de passagem Interpolados

de forma análoga a um único ponto, aqui há múltiplas soluções que dependem do valor da aceleração usada nas combinações. Caso seja conhecido os valores dos pontos da trajetória θ_2, com sua aceleração $\ddot{\theta_2}$ e duração do movimento t_{12}, podemos resolver as equações para o tempo $t_2$que é o tempo em que o manipulador passará pelo ponto θ_2. Para os pontos internos da trajetória as equações são as seguintes

$$\dot{\theta}_{12} = \frac{\theta_2 - \theta_1}{t_{total12}} \tag{30}$$

$$\ddot{\theta_2} = SGN(\dot{\theta}_{23} - \dot{\theta}_{12})|\ddot{\theta_2} \tag{31}$$

$$t_2 = \frac{\dot{\theta}_{23} - \dot{\theta}_{12}}{\ddot{\theta_2}} \tag{32}$$

$$t_{12} = t_{total12} - \frac{t_1}{2} - \frac{t_2}{2} \tag{33}$$

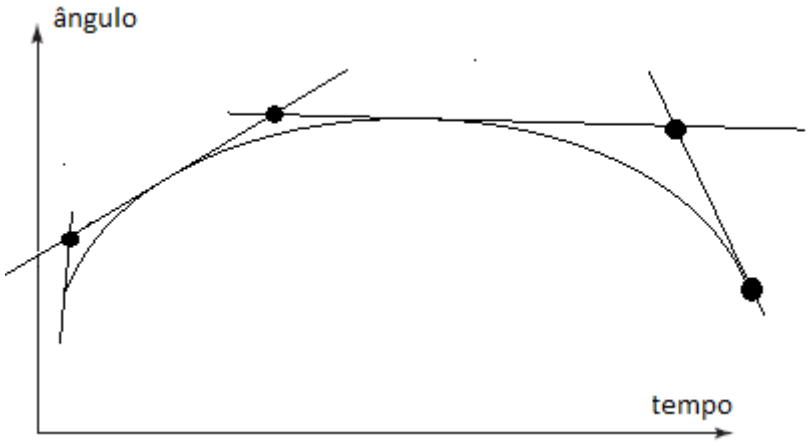


Figura 16 – Geração de trajetória com interpolação parabólica

Como pode ser visualizado na Figura 16, nesse método de criação das trajetórias chamadas de *splines de 2ª ordem*, o manipulador não passa exatamente no ponto desejado, e sim, apenas próximo dele. Quando há atuadores com uma aceleração alto o suficiente, as trajetórias chegam mais próximas dos pontos. Porém, se há a necessidade da passagem exata do manipulador por estes pontos definidos, sem parar é necessário a criação de *pseudopontos* de passagem. Como pode ser visualizada na Figura 17, os pontos em forma de estrela são os desejados, e os pontos acima são os pontos criados, geralmente pelo sistema, para que o manipulador robótico ande exatamente nos pontos desejados.

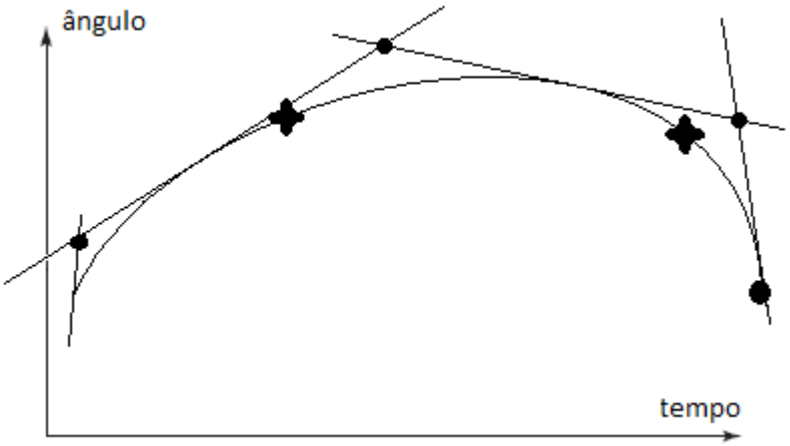


Figura 17 - Trajetória com pseudopontos

3.5 – Programação

Nesta seção será abordado alguns temas relacionados a programação do manipulador.

3.5.1 – SDK

O kit de desenvolvimento de software, do inglês *Software development kit*, ou simplesmente SDK, é basicamente ferramentas que facilitam o desenvolvimento de software que permite a criação de aplicativos para plataformas de hardware, sistema de computadores, entre outros[64,65]. As empresas geralmente fornecem os SDKs para que programadores externos tenham uma melhor integração melhorada com o software ou produto proposto[65]. Um exemplo de um SDK é o Platform SDK da Microsoft que inclui documentação, código e utilitários, facilitando assim o trabalho de qualquer programador.

3.5.2 - Biblioteca de Vínculo Dinâmico

Uma biblioteca de vínculo dinâmico, do inglês *dynamic-link library*, ou DLL é uma biblioteca que possui códigos e dados que podem, e provavelmente deverão, ser usados por mais de um programa ao mesmo tempo. Um bom exemplo disso são as DLL encontradas em sistemas operacionais.

Fazer uso de uma DLL, possibilita que um programa possa ser modularizado em vários componentes. Como existe uma modularidade o código não precisa ser todo carregado, dessa forma o tempo de carregamento do programa é mais rápido e um módulo é carregado somente quando essa funcionalidade é solicitada [66].

Resumidamente, o uso de DLLs na elaboração do software implica em três pontos positivos:

- Utiliza menos recurso;
- Promove a arquitetura modular;
- Implantação e instalação são facilitadas.

3.5.3 – Driver de Comunicação

A função de um driver é aceitar requerimentos abstratos do software independente do dispositivo em que esteja sendo executado e zelar para que a solicitação seja executada, permitindo que o software interaja com o dispositivo, por exemplo, executando alguma função responsável pela interface.

É importante mencionar que um driver não é um processo ou tarefa gerenciada pelo sistema, e sim um conjunto de funções que contém informação sobre o dispositivo periférico ao qual envia comandos.

3.6 - Considerações finais

Este capítulo apresentou a teoria utilizada para desenvolvimento deste PFC. Foram discutidos diversos temas, todos eles, direta ou indiretamente, se fizeram presente no atual projeto. Todo o conteúdo acima se fez presente durante a graduação do autor, dessa forma, bastou apenas que essas informações fossem compiladas em forma de texto, para que assim, o leitor, mesmo não tendo experiencia ou conhecimentos na área, obtenha um melhor entendimento do sistema desenvolvido.

4 – O PROJETO

Neste capítulo será descrito ainda em forma conceitual o que é o sistema a ser implementado, descrevendo a proposta para solução do problema. Aqui serão mostrados diagramas com pontos de vista complementares do projeto, além dos requisitos de sistema e a metodologia de projeto para seu desenvolvimento.

4.1 – Solução Proposta (projeto proposto)

A tarefa planejada para resolver o problema apresentado na introdução deste trabalho passa por desenvolver um manipulador robótico de cadeia aberta com elos redundantes. Nesta etapa do projeto, a execução será concentrada nos últimos elos da cadeia. Há a necessidade de uma plataforma para geração de trajetórias, bem como, realização de testes de algoritmos de movimentação, portanto alguma interface gráfica deve ser providenciada.

4.2 – Requisitos do Robô

Após reuniões com o cliente, alguns dos requisitos do sistema foram decididos, porém, se observou que o cliente não tinha uma noção precisa e completa de sua demanda. Identificou-se no projeto uma prova de conceito (*Proof of Concept*) que visava, basicamente, a viabilidade da substituição de manipuladores antropomórficos, por um robô ainda a ser projetado. Como a tarefa em que seria aplicado o robô não estava toda definida, em reuniões internas da equipe, ficou claro que deveria ser projetado um manipulador de forma que pudesse atender a maior quantidade de tarefas possíveis.

Complementarmente, neste PFC e após discussões internas no SENAI, propuseram-se novas análises. Abaixo serão descritos alguns requisitos do projeto, alguns deles sua execução não entra no trabalho do autor deste PFC, mas será apresentado apenas para registro.

4.2.1 – Requisitos

Para o robô a ser construído foram levantados como requisitos:

- Posicionamento suficiente para a realização de processos com a necessidade de alta precisão, solda a laser, por exemplo;
- Movimentação com velocidade superior ou, pelo menos, semelhante a outros robôs industriais comuns de acordo com especificações de catálogos;
- Maximizar a capacidade de aplicar forças no efetuador final;
- Implementar um *software* onde possa ser realizada a programação do robô;
- Monitorar variáveis importantes, tais como posição de juntas, correntes nos motores, entre outras;
- Plataforma precisa manter um registro histórico das operações realizadas.

4.3 – Arquitetura do Sistema

A arquitetura final para este projeto é apresentada na Figura 18, as setas indicam o fluxo da comunicação entre cada bloco.

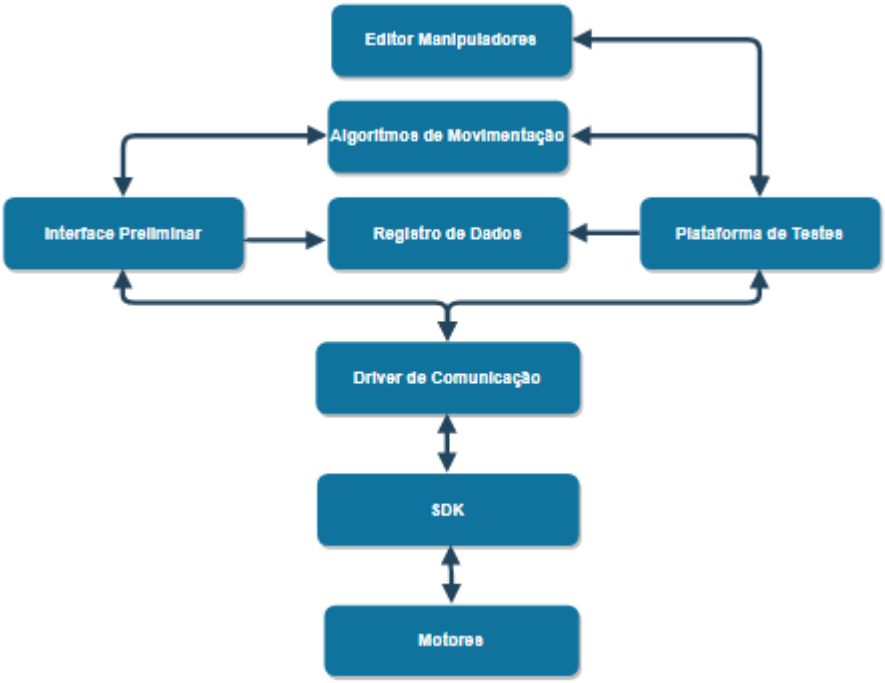


Figura 18 - Comunicação entre os sistemas desenvolvidos.

Fonte: Arquivo Pessoal.

A arquitetura final foi desenvolvida praticamente toda, no processo concepção do projeto, por meio de iterações com o cliente e entre as reuniões dos integrantes do projeto. Após esse período optou-se por um desenvolvimento modular devido aos seguintes argumentos:

- A possibilidade do desenvolvimento modular, facilita o planejamento do trabalho durante o desenvolvimento.
- A independência de cada módulo permite, por exemplo, que a interface seja trocada ou que mais de uma interface diferente acesse a camada inferior. Outra possibilidade é de que a comunicação pode ser alterada, como para modelos diferentes de motor.

Esses benefícios são realmente importantes, pois existe a grande possibilidade deste sistema ser usado em novas aplicações e essa característica é muito interessante para se adequar a diferentes realidades em diferentes clientes.

4.4 – Diagrama de Implementação

O diagrama de implantação modela a estrutura física do sistema, mostrando a relação entre os componentes do software e o hardware. O diagrama de implementação da Figura 19 modela o sistema desenvolvido para o manipulador desta primeira etapa do projeto, ou seja, com apenas três motores.

Os dispositivos da Figura 19 são os motores responsáveis pela atuação na estrutura, o primeiro motor é diretamente ligado ao conversor USB ←→ RS485, o segundo e o terceiro são ligados em série a partir do primeiro motor. Essa configuração de ligação é possível devido ao protocolo de comunicação RS485, e cada dispositivo ligado nessa rede possui a denominação de nó, a seção 3.2.2 apresentou este conceito.

A caixa intermediária, como o próprio nome sugere, é um conversor de dados, ele faz o papel de “ponte” entre o computador e os motores do manipulador. Ele recebe as informações através da porta usb do computador, atualmente, e envia para os motores, e “coleta” as informações provenientes do manipulador para transmitir para o computador. Note que apesar de ter sido usada a palavra coleta, este

dispositivo não possui qualquer tipo de computação ou armazenamento das informações, simplesmente é um meio no qual a informação transita.

Por fim, o dispositivo da direita inferior é o hardware usado para o processamento, tanto comandos como interpretação dos resultados, do manipulador. Atualmente este *hardware* é desempenhado por um computador, mas facilmente pode ser adaptado para um servidor na nuvem com um poder de processamento muito maior, o qual não teria problemas em comandar vários robôs simultaneamente.

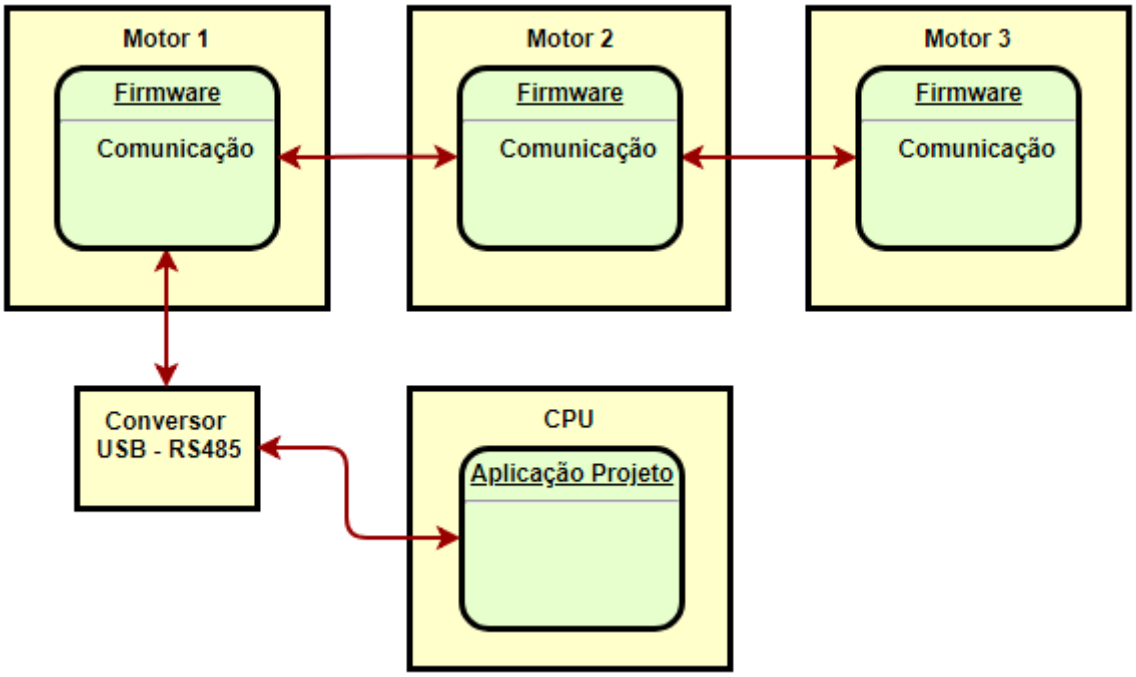


Figura 19 - Arquitetura da ligação física.

Fonte: Arquivo Pessoal

4.5 – Metodologia do Desenvolvimento

O projeto foi dividido em três macro partes para serem, primeiramente, desenvolvidas independentemente e essas partes por sua vez, possuem mais uma divisão em tarefas menores que podem ou não ser compartilhadas por macro diferentes, como é o caso do registro de dados que é compartilhada em duas macros e o editor de manipuladores que faz apenas parte da plataforma virtual de testes. Essas partes podem ser visualizadas na Figura 20.

O bloco laranja e o bloco rosa, mais a esquerda e direita respectivamente, tiveram sua implementação iniciada de forma paralela, após a parte da interface preliminar e driver de comunicação tenha sido finalizada o bloco em verde, ao centro, teve seu início. Dentro de cada bloco as tarefas foram desenvolvidas utilizando a metodologia em cascata, ou seja, partindo dos requisitos foi projetada uma arquitetura e implementada uma parte de cada vez, até se chegar ao produto final, porém, com a realização dos testes é normal que se faça necessário alguns ajustes. A Figura 21 ilustra os passos para implementação do sistema.

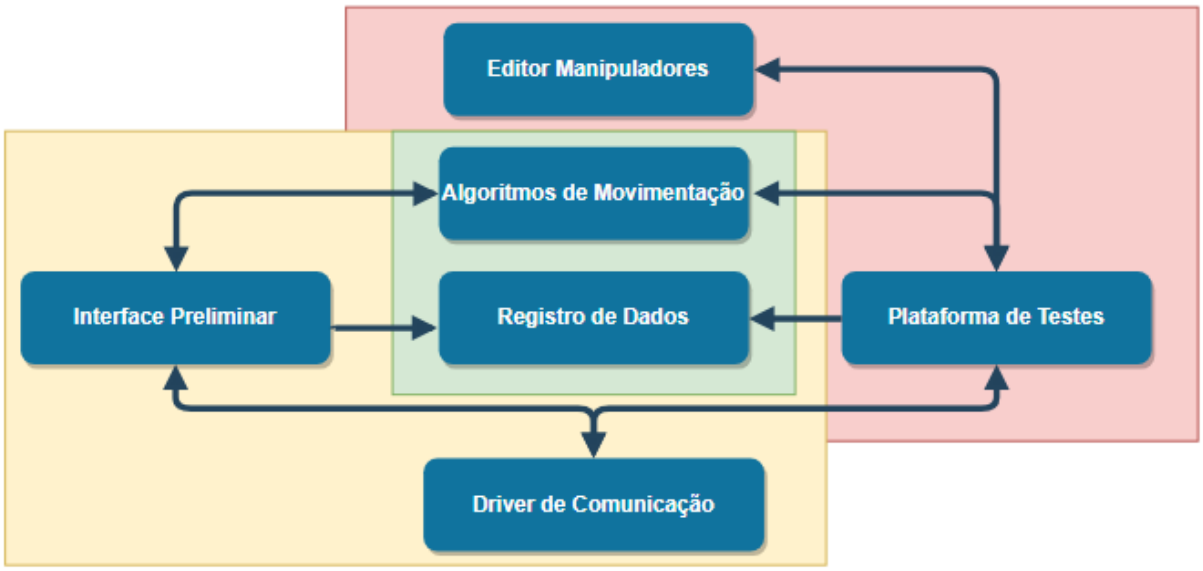


Figura 20 – Macro tarefas do projeto.

Fonte: Arquivo Pessoal

O modelo em cascata considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, e representa cada uma delas como fases distintas, como: especificação de requisitos, projeto de software, implementação, teste e assim por diante. [61]

No entanto, houveram mais de uma versão para cada tarefa implementada até que fosse considerada pronta e adequada para atingir os requisitos. Assim, considera-se que cada uma das três partes teve um desenvolvimento incremental.

A abordagem do desenvolvimento incremental é intercalada entre as atividades de especificação, desenvolvimento e validação. Dessa forma, o sistema é desenvolvido como uma série de versões, de maneira que cada versão adiciona funcionalidade à anterior. Esses modelos não são mutuamente exclusivos e

geralmente usados em conjunto, especialmente quanto o desenvolvimento resultará em sistemas de grande porte. Para sistemas de grande porte, faz sentido combinar algumas das melhores características do modelo em cascata e dos modelos de desenvolvimento incremental. [61]

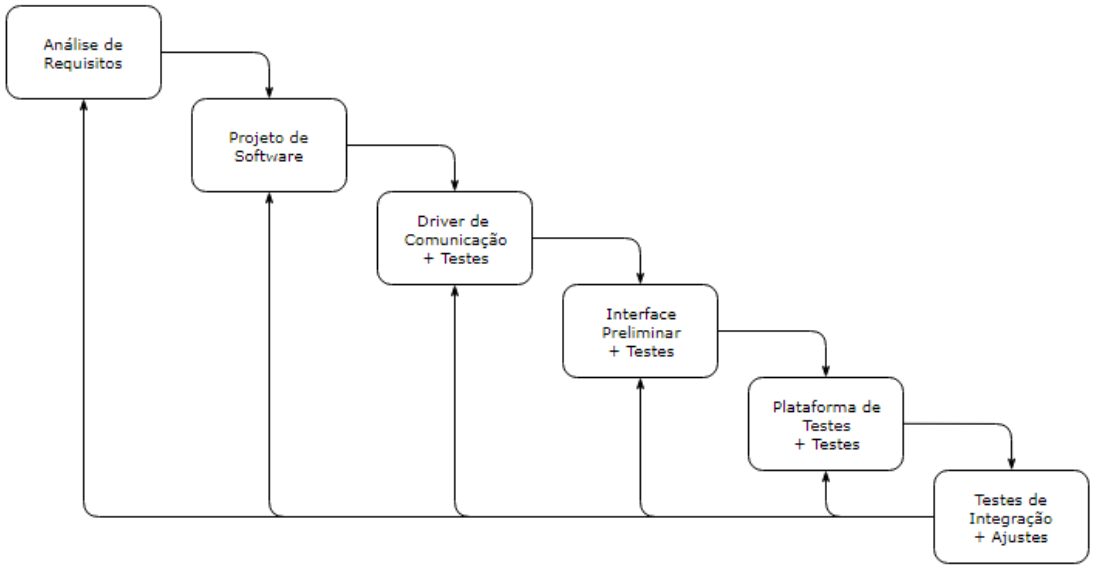


Figura 21 - Representação dos Passos do Projeto na Metodologia Cascata.

Fonte: Arquivo Pessoal

A Figura 22 ilustra essa situação de união entre as duas metodologias tradicionais e mostra uma abordagem mais realista em relação a como o sistema foi desenvolvido.

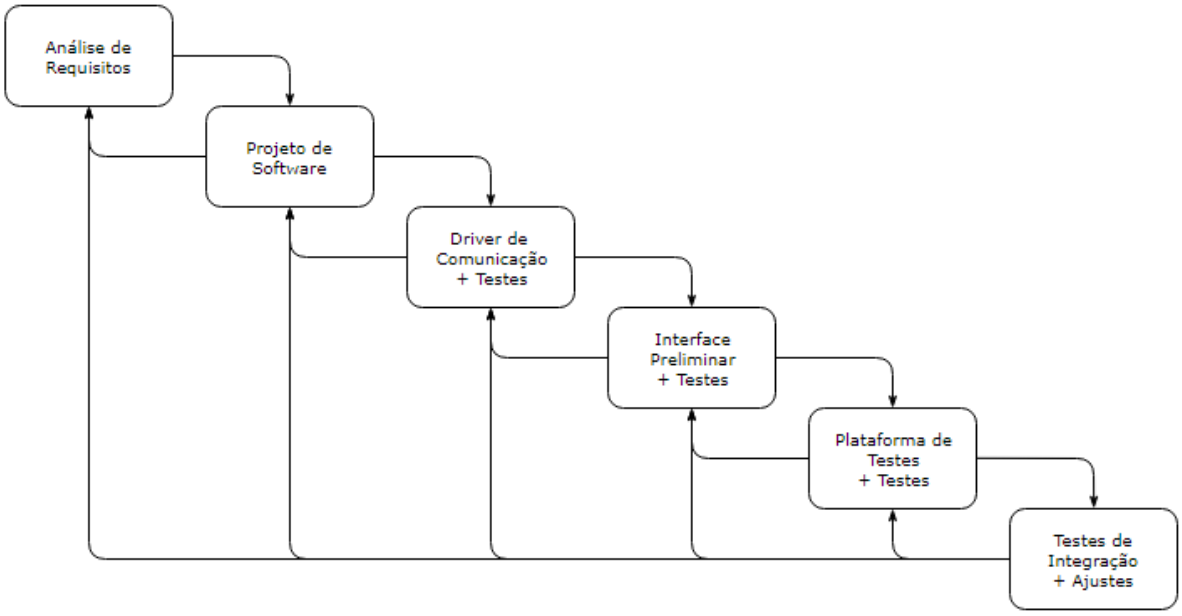


Figura 22 - Metodologia Combinada Entre Cascata e Incremental.

Fonte: Arquivo Pessoal

4.6 – Considerações Finais

Neste capítulo foram apresentados os aspectos teóricos para o desenvolvimento do projeto. Todo este processo pré-projeto foi de grande valia, pois, dessa forma, todo o desenvolvimento ficou mais claro para todos os envolvidos em sua realização. No próximo capítulo serão apresentadas as ferramentas utilizadas no desenvolvimento efetivo do projeto.

5 - SELEÇÃO DE FERRAMENTAS E TECNOLOGIAS

Neste capítulo serão apresentadas as ferramentas e opção para a realização destas escolhas. Algumas das escolhas foram tomadas de forma puramente técnicas, outras por motivos de preferência do próprio cliente, isso, aos olhos do autor foi muito interessante, porque nem sempre as melhores soluções estarão disponíveis, seja por falta de prazo para realizar a compra ou simplesmente, por motivos de orçamento. É importante mencionar que o autor deste, participou apenas das escolhas com a qual estava diretamente envolvido, como controle dos motores, programação dos softwares e camada física responsável pela comunicação de todos os módulos.

5.1 – Robô

No início da execução do projeto houve um período de pesquisa para levantar informações suficientes para dar continuidade ao trabalho. A equipe toda participou e após a finalização desta tarefa, todas as informações foram compiladas em uma grande apresentação a qual seria utilizada na reunião para defender a estrutura, topologia de informação, programação projetada e etc.

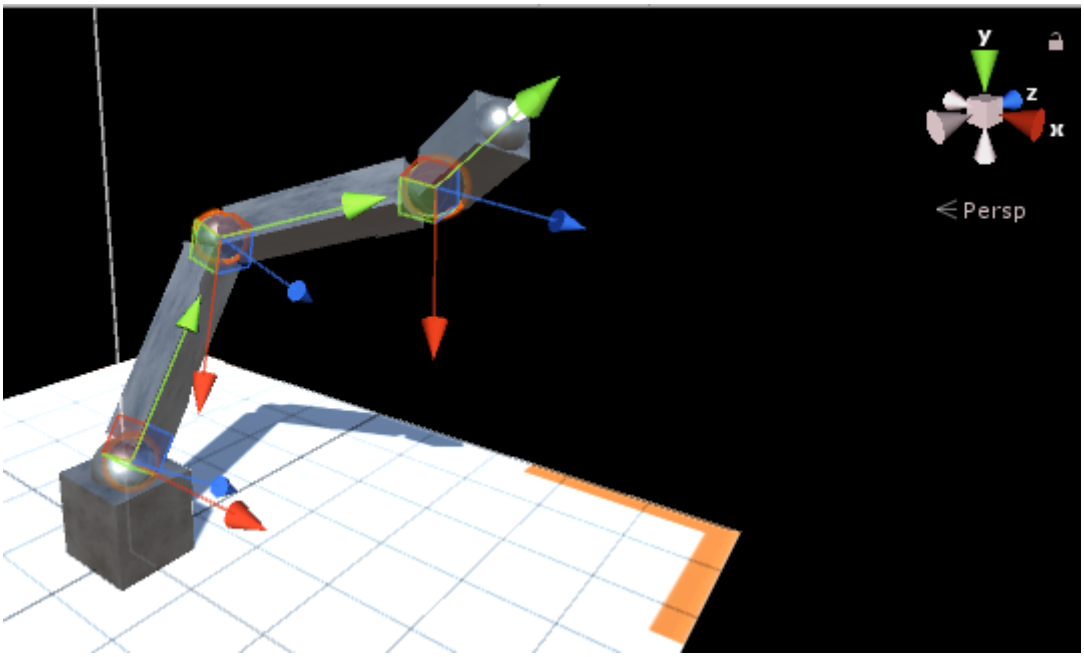


Figura 23 - Modelo Virtual do Robô Utilizado na Primeira Entrega do Projeto.

Fonte: Arquivo Pessoal

Ao fim da reunião ficou acordado que a estrutura do robô deveria ter oito graus de liberdade (8GDL), porém pelo entendimento da equipe, a primeira etapa poderia ser construída com apenas três. Dessa forma a entrega desta etapa seria toda desenvolvida em uma estrutura robótica como da Figura 23. O primeiro e terceiro motor realizam a movimentação sobre o eixo Z e o segundo motor realiza a movimentação sobre o eixo X [3].

Uma vez definida a concepção cinemática para o protótipo, o foco passou para a escolha dos motores. A discussão entre os membros da equipe e o cliente finalizou com a escolha de motores da fabricante DYNAMIXEL, modelo h54-200-s500-r. A Figura 24 mostra o servo motor escolhido.

Figura 24 - Servo Motor Utilizado no Manipulador Robótico.

Fonte: [67]

O servo motor apresenta muitas vantagens para a realização deste projeto, pois, é integrado por um sistema de redução de engrenagem, o que lhe garante grande precisão e torque. Por se tratar de um servo motor, já possui controladores de posição (Proporcional) e velocidade (Proporcional Integral) em sua estrutura, além de um controlador de torque. Com uma resolução de aproximadamente 0,000717 graus, e capacidade de realizar 44,7 Nm este motor atende perfeitamente o projeto, pois é capaz de realizar grande quantidade de força com uma enorme precisão e 855 gramas[68]. Sua comunicação é baseada no protocolo RS485, o que permite que seja

[3] Observe que cada eixo tem por referência o elo local.

criada uma “rede” com os motores com apenas dois fios, facilitando a fabricação dos elos.

5.2 – Plataforma de Desenvolvimento

Para o desenvolvimento dos softwares havia uma grande quantidade de opções, tanto comerciais como gratuitas cada uma com suas vantagens e limitações. Atendendo a um requisito do cliente, os softwares deveriam ser criados de forma customizada e a partir do zero. Não era desejável aplicações rodando em plataformas como Robo DK, Webots, entre outros.

O driver de comunicação seria desenvolvido em cima da SDK fornecida pelo próprio fabricante, que por sua vez foi implementado utilizando a IDE Visual Studio, que é um ambiente de desenvolvimento integrado da Microsoft. Dessa forma, a ferramenta para este desenvolvimento seria o próprio Visual Studio. Tanto a DLL como os projetos exemplos haviam sido implementados com esta IDE, o que acabou se tornado algo muito positivo para o desenvolvimento futuro. A linguagem escolhida para o desenvolvimento do driver foi “C”, pois sua integração com posterior plataforma ou aplicações seria menos trabalhosa e, novamente, pelo fato do desenvolvedor possuir um conhecimento bastante sólido nesta linguagem.

A plataforma de testes por possuir requisitos mais avançados, não seria desenvolvida no Visual Studio. Com exigências de simulações em um ambiente 3D, edição de manipuladores com uma interface gráfica, testes de algoritmos e se possível multiplataforma, o desenvolvimento requeria de uma ferramenta mais apropriada para tal, porém, como dito anteriormente softwares como Robo DK e afins não deveriam ser utilizados. Softwares como LabVIEW e Matlab entraram na lista de possíveis candidatos, mas as aplicações em 3D se tornariam muito complexas, se possíveis, para serem desenvolvidas. Sendo assim a engine Unity foi escolhida para a realização desta tarefa, com um foco voltado para a criação de jogos e renderização suas funções gráficas seriam perfeitas para desenvolver toda a interface que o usuário necessitaria.

Figura 25 - Logotipo das Plataformas Utilizadas no Desenvolvimento.

Fonte: [69,70]

5.3 - Considerações finais

Neste capítulo foram apresentadas as escolhas realizadas de ferramentas para a implementação do sistema. Nem sempre está é uma tarefa simples, muitas vezes não podemos atingir estado ideal, que seria as melhores ferramentas ou as quais estamos mais adaptados. Dessa forma, precisamos manter um equilíbrio para atender as especificações o orçamento e o prazo para a realização do projeto.

6 – IMPLEMENTAÇÃO DO SISTEMA

Este capítulo dedica-se a mostrar os aspectos de projeto para atender os objetivos do trabalho, ou seja, como foi realizado o desenvolvimento do sistema robótico. Apesar de estar dividido em vários tópicos para facilitar a compreensão, é importante salientar que todas as tarefas foram implementadas de forma paralela e iterativa. Ademais, mostra-se como foi o desenvolvimento da solução.

6.1 – Estudo dos Motores e SDK

Após o recebimento dos motores que seriam utilizados para o desenvolvimento do protótipo do manipulador, iniciou-se a fase de aprendizado do funcionamento dos motores. Esta etapa foi muito importante para descobrir quais eram os parâmetros dos motores que teríamos acesso e como acessá-los.

Como já comentado no capítulo 4 a ligação dos motores foi realizada em uma forma de rede onde cada motor estaria ligado a um par de fios. O software de comando, executado no PC, envia comandos e recebe as respostas fazendo uso de uma porta USB, que estaria ligado ao dispositivo conversor RS485, e logo em seguida estariam todos os servos motores ligados em uma topologia de barramento linear onde é usado apenas dois fios, conforme apresentado na seção 3.2.2 – RS-485. A Figura 26 apresenta essa ligação.

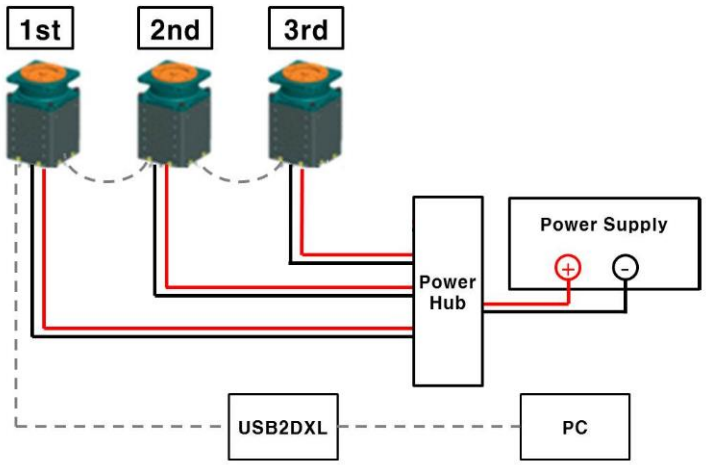


Figura 26 - Ligação Física dos Motores.

Fonte: [67] modificada

Com o auxílio do software de demonstração e da documentação do motor, ambos fornecidos pelo próprio fabricante, em poucas horas foi possível conhecer o funcionamento dos motores. A Figura 27 apresenta uma tela do software fornecido pelo fabricante, nela podemos observar que ao lado esquerdo estão os identificadores dos três motores conectados, na região central estão alguns valores presentes na programação do servo, e ao lado direito está o menu onde é possível alterar esses valores.

Apenas com a leitura da documentação haviam ficado algumas dúvidas sobre como o motor funcionava em alguns pontos específicos, para ser mais preciso, no modo torque, mas com ajuda deste *software* todas as dúvidas foram resolvidas.

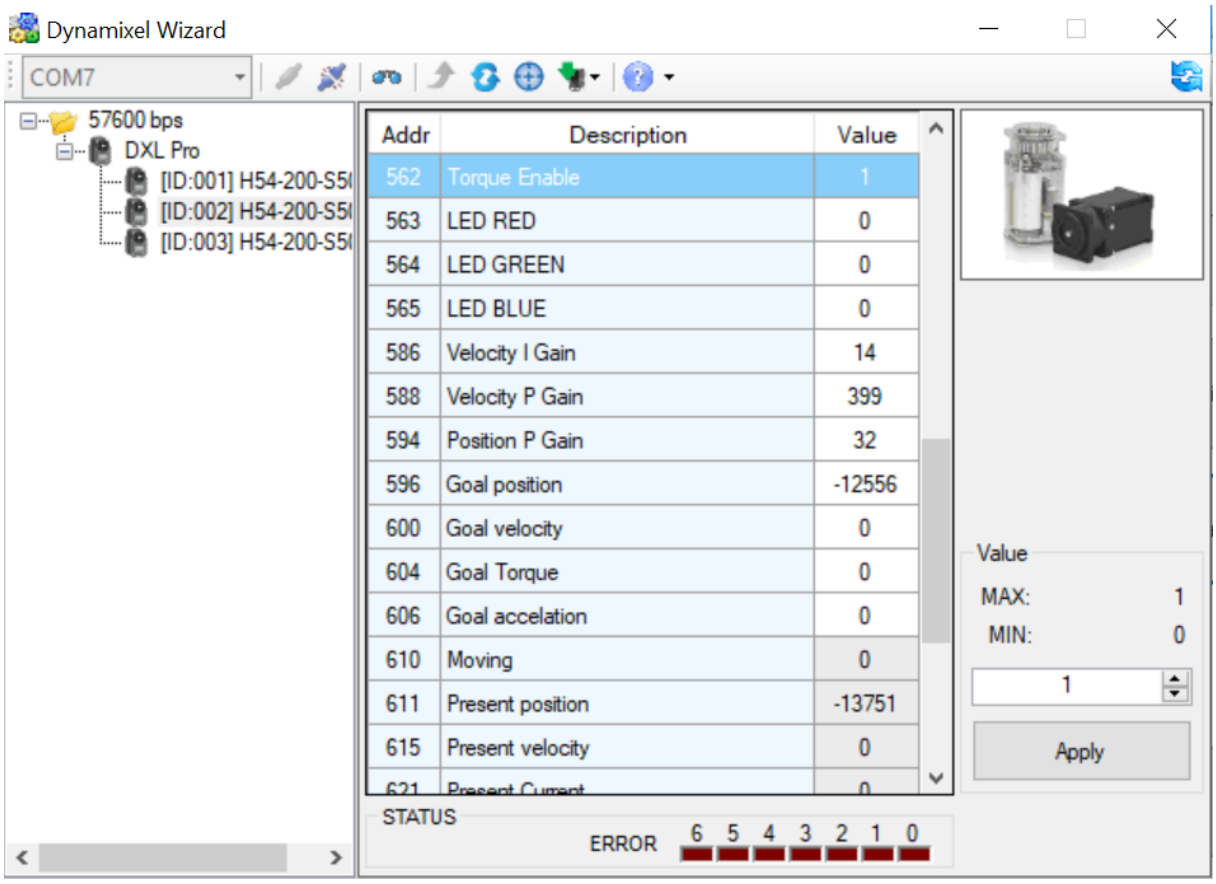


Figura 27 - Tela do Software Fornecido pelo Fabricantes dos Motores.

Fonte: Arquivo Pessoal

No kit de desenvolvimento ou SDK, estão alocadas as funções mais básicas, essas funções executam a passagem de apenas uma única instrução aos motores,

como por exemplo, “*mover para posição x*”, “*trocar velocidade desejada para y*”, ou então para realizar consultas como “*qual sua posição atual?*”.

Com a execução de alguns projetos exemplos ficou claro para o autor deste trabalho, o funcionamento da biblioteca, em termos de programação. Após isso, foi finalmente possível testar o driver de comunicação. É importante mencionar que o driver já estava em desenvolvimento, mas que até então, não havia em mãos os motores para verificar a funcionalidade do mesmo, dessa forma seu desenvolvimento se caracterizou mais com viés de um trabalho “teórico”.

Com as primeiras versões do driver de comunicação, ainda com funções bastante primárias, foram obtidos dados para a criação dos primeiros gráficos de resposta dos motores, curvas de velocidade, aceleração, posição e etc. Isso era necessário para que fosse avaliada a eficiência dos motores para realizarem a tarefa para que foram designados.

6.2 – Motores

Após alguns testes constatou-se que a sintonia do controlador de posição dos motores estava causando uma resposta muito conservativa, o motor parava onde deveria, mas para se atingir o ponto desejado com a precisão requerida os motores estavam demandando um tempo maior do que o esperado. Independente da velocidade ou aceleração que foram configurados previamente, a movimentação se comportava da mesma maneira.

A Figura 28 apresenta o gráfico de posição dos três servos para algumas posições com velocidade, aceleração e parâmetros do controlador com a configuração proveniente de fábrica. Como se pode observar os motores atingem a vizinhança do ponto desejado muito rápido, mas a partir deste ponto os motores se movem muito lentamente.

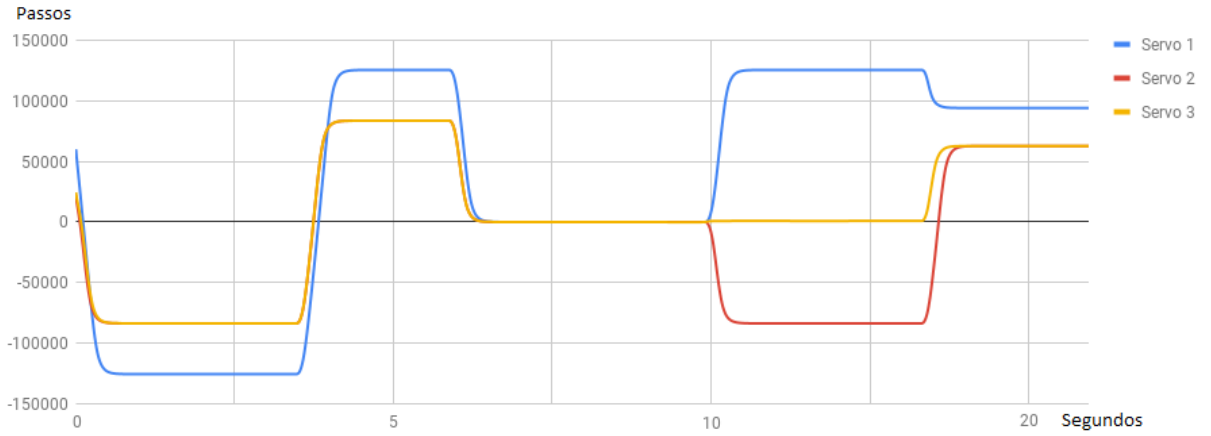


Figura 28 - Gráfico de Posição dos Motores.

Fonte: Arquivo Pessoal

Foram levantadas algumas soluções para este problema. A primeira delas é logo descartada seria reduzir as especificações de posicionamento dos motores. Porém, apesar da solução se mostrar eficiente, no requisito tempo, a grande quantidade de motores que o robô deverá comportar no fim do projeto pode, e com certeza irá, implicar em grandes erros de posicionamento na ferramenta no manipulador.

Dessa forma, a abordagem decidida foi de realizar a configuração dos parâmetros dos controladores responsáveis pelo posicionamento dos servo-motores. Sem muito tempo disponível para realização de testes e planejamento necessário para levantar os modelos dos motores e assim configurar os controladores adequadamente, optou-se pela realização de uma calibração automática, enquanto o teste de resistência dos motores era realizado.

Após a conclusão de uma primeira versão estável do driver, estava previsto um teste de resistência, para analisar o comportamento dos motores na estrutura. Porém, agora além do teste seria realizada também a calibração dos controladores de posicionamento dos motores[4].

Realizou-se os testes através da inclusão de vários ângulos nos motores, aproximadamente dez combinações de ângulos para motor, através de uma funcionalidade do driver. Esses ângulos seriam inseridos de forma cíclica e contínua,

[4] Nestes testes e análises o motor estava sem carga no eixo

e ao fim de cada ciclo, alguns dados dos servos passariam a ser coletados para posterior análise, como temperatura por exemplo.

Paralelamente a essa função, haveria outra que estaria incumbida de realizar a configuração dos parâmetros do controlador e analisar as respostas dos motores, que neste caso seria a posição dos motores. A Figura 29 contém um fluxograma do teste realizado.

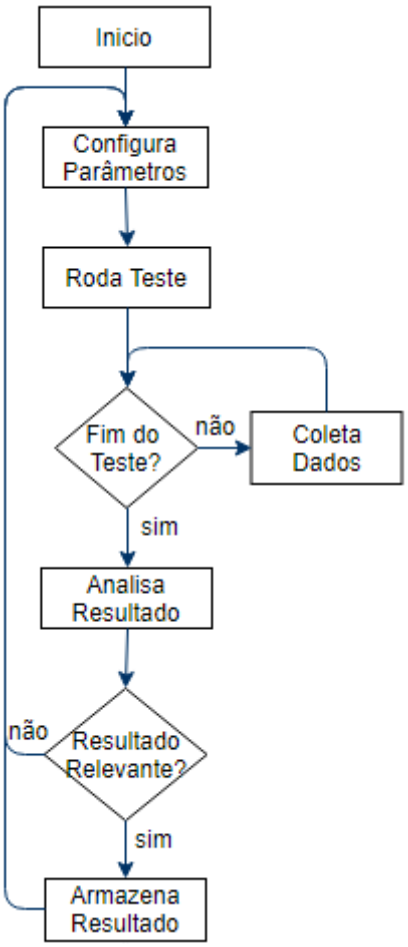


Figura 29 - Fluxograma do Algoritmo de Configuração dos Controladores.

Fonte: Arquivo Pessoal

Após algumas horas de simulação a fase de teste foi encerrada. Com resultados considerados, pelo menos para esta primeira fase, suficientes e com isso o desenvolvimento teve continuidade. Em relação aos resultados de resistência, não houveram problemas com os motores ou com a estrutura, a temperatura de funcionamento não ultrapassou 60 graus, nem mesmo quando a velocidade,

aceleração e torque estavam configurados próximos do máximo. Quanto ao resultado de posicionamento as Figura 30 e Figura 31 apresentam os valores obtidos.

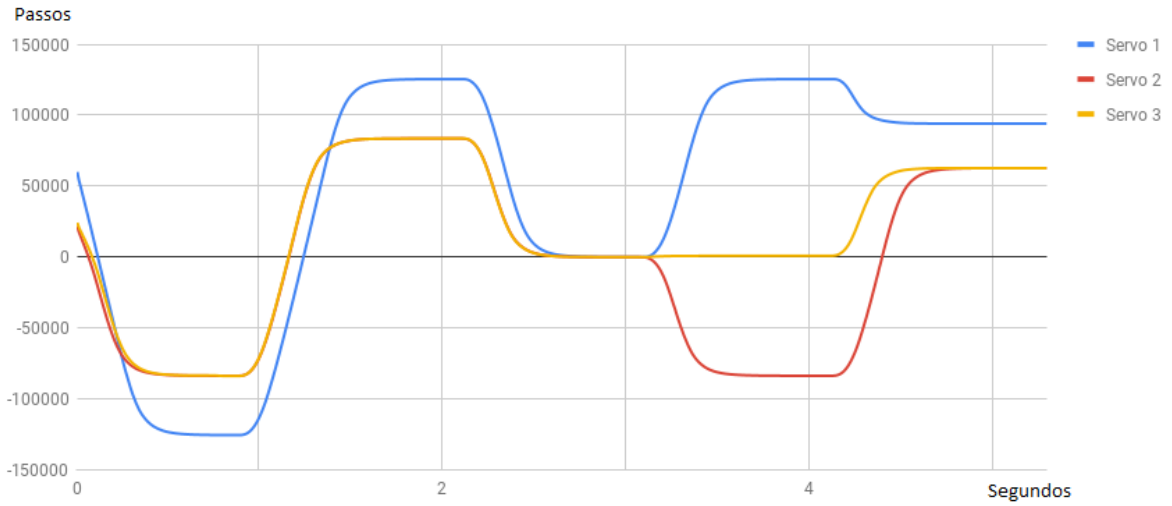


Figura 30 - Gráfico de Posição dos Motores Após Configuração dos Controladores.

Fonte: Arquivo Pessoal

A Figura 30 mostra o resultado de posicionamento onde foram utilizados os mesmos pontos do gráfico da Figura 28 referente à configuração de fábrica. Pode-se observar uma diferença considerável no tempo de execução dos posicionamentos. Agora a movimentação completa está levando aproximadamente 5 segundos, no primeiro gráfico foram necessários aproximadamente 30 segundos. Abaixo a Figura 31 apresenta o gráfico de velocidade dos motores.

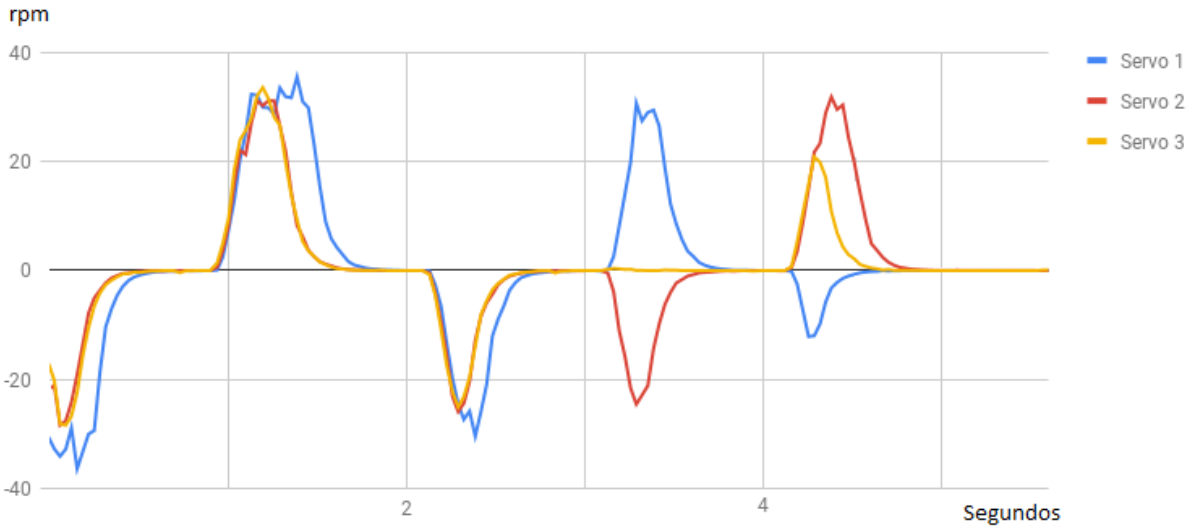


Figura 31 - Gráfico de Velocidade dos Motores Após a Configuração dos Controladores.

Fonte: Arquivo Pessoal

6.3 – Driver de Comunicação e Interface

Após a etapa de estudo e aprimoramento dos motores que serão utilizados como atuadores do manipulador, a atenção foi passada para a conclusão da etapa de implementação do driver de comunicação com o PC. Sua finalidade, primeiramente, era de testar algumas funções mais elaboradas para utilização dos motores e verificar o comportamento da estrutura do robô em movimentação. Para isso o objetivo do driver é, integrar várias dessas funções em uma única, tornando-a assim mais complexa e interessante. Um exemplo disso seria uma função para a movimentação para uma coordenada específica, mas essa movimentação deve ser realizada e um tempo determinado ou então em uma velocidade desejada. As funções, assim como o driver, foram implementadas, com várias versões preliminares.

Esta etapa do projeto foi, inicialmente, planejada para ser de forma simples e enxuta, apenas para atender as necessidades iniciais, mas logo no início das atividades ficou claro que sua complexidade e aplicação teriam uma participação maior no projeto, devido a problemas no desenvolvimento da Plataforma de testes, que será abordada mais adiante. Além das opções de testes de funcionamento, foram implementadas, também, algumas funções com o propósito de realizar testes de algoritmos de movimentação e outras com a finalidade de apresentação do robô para fins comerciais.

Apesar da sua importância ter sido consideravelmente ampliada dentro do escopo, sua interface continuava sendo implementada de forma minimalista, simplificando ao máximo, para que com isso o foco ficasse no desenvolvimento das funções e testes do robô. A Figura 32 apresenta a tela inicial do software, nela podem ser vistas as principais funcionalidades que foram implementadas.

A primeira opção, MOVIMENTO MANUAL, é uma implementação que aceita comandos através do teclado do dispositivo no qual está sendo executada a aplicação. Fazendo uso dessa opção, o usuário, através de simples comandos, movimenta cada junta do robô de forma independente.

Escolhendo este modo de funcionamento, uma nova tela é apresentada ao usuário onde é possível visualizar o tamanho do passo que será aplicado a junta a cada comando do usuário, por exemplo, se o passo estiver configurado para um

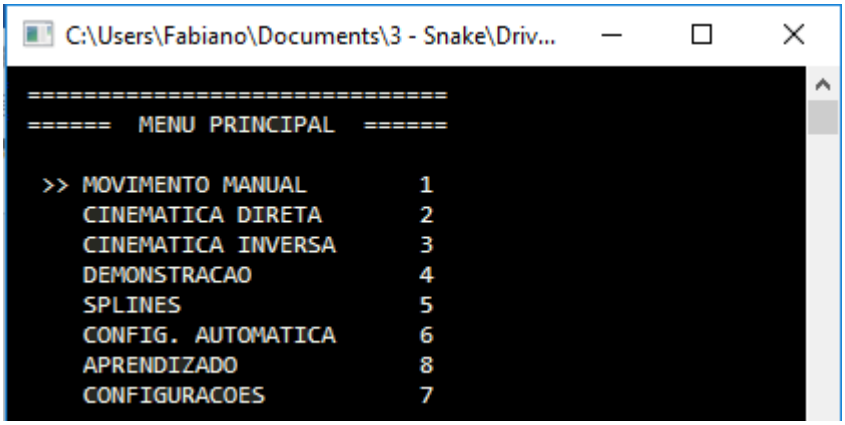


Figura 32 - Menu Principal Driver.

Fonte: Arquivo Pessoal

grau, cada vez que algum comando de movimento for dado ao manipulador, ele irá se mover um grau na junta selecionada. Logo abaixo é apresentado os ângulos de cada junta e a coordenada do efetuador final, como pode ser visto na Figura 33.

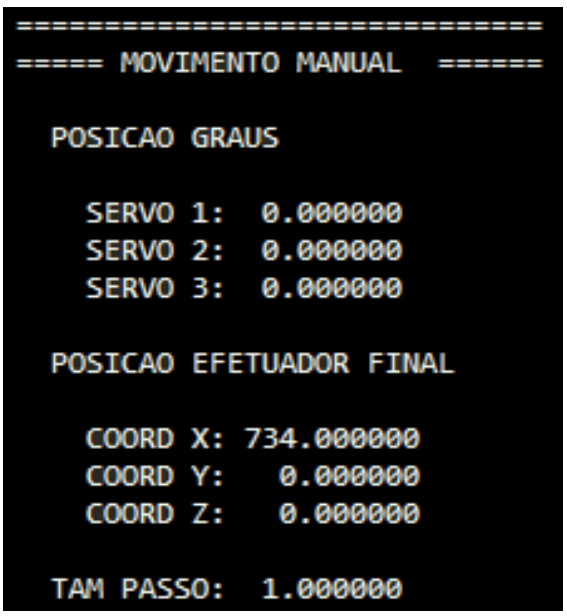


Figura 33 - Tela Driver Movimento Manual.

Fonte: Arquivo Pessoal

A segunda opção do menu principal, CINEMATICA DIRETA, é o modo onde é possível passar diretamente para os motores o ângulo desejado (em graus), como o nome já sugere essa opção implementa a cinemática direta do manipulador. Os ângulos são inseridos no *prompt* de comando através do teclado e após isto, a movimentação dos servos é iniciada. Esta é uma função simples em se tratando de movimentação do robô, onde apenas a posição final importa, dessa forma, a função não garante uma trajetória definida durante a movimentação, sem um controle do posicionamento do manipulador entre o ponto inicial e final, a função só garante o posicionamento final. A Figura 34 apresenta o processo de inserção dos ângulos desejados em cada junta.

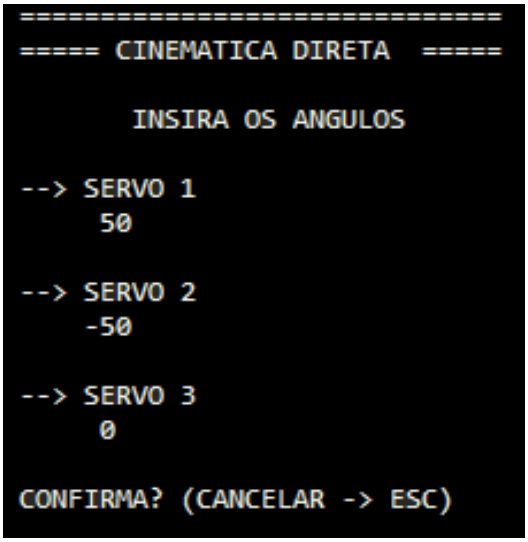


Figura 34 - Tela Driver Cinemática Direta.

Fonte: Arquivo Pessoal

A terceira opção como o próprio nome sugere implementa a CINEMATICA INVERSA. Esta função está diretamente relacionada com o uso de bibliotecas externas acessada através de uma API desenvolvida especificamente para a realização dessa integração entre a biblioteca e o driver. Neste modo a posição final do atuador é passada através de coordenadas. A Figura 35 ilustra a entrada de dados e o retorno deles após execução na biblioteca externa. Primeiramente, a coordenada desejada é inserida de forma semelhante ao modo CINEMATICA DIRETA. Após a inserção das coordenadas é criada uma trajetória linear entre os pontos de início e fim, os cálculos fazem uso de um modelo virtual do robô, que é instanciado ao iniciar

a execução do driver, esse modelo também é criado pela biblioteca externa chamada pela API. Caso a trajetória esteja fora da área de trabalho do robô, a API terá como resposta a trajetória mais próxima.

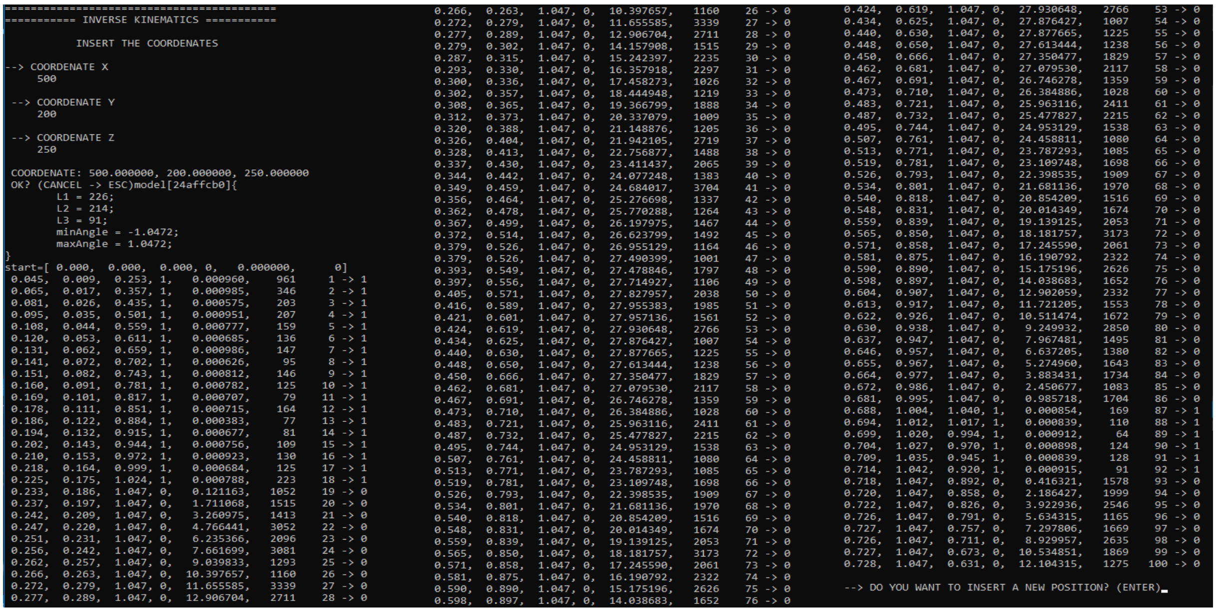


Figura 35 - Tela de Resultados Cinemática Inversa.

Fonte: Arquivo Pessoal

Como dito anteriormente a Figura 35 apresenta as entradas e saídas do modo CINEMATICA INVERSA, mas agora iremos apresentar cada detalhe da imagem. Primeiramente, é inserida a coordenada desejada (mm) para mover o manipulador, neste caso (x, y, z) = (500, 200, 250). Logo em seguida, é apresentado o modelo virtual utilizado para os cálculos da cinemática inversa, e os ângulos máximos e mínimos que cada junta poderá alcançar, os valores são apresentados em radianos.

Após isso, os primeiros valores amostrados são o ponto de partida do robô, neste caso ele está partindo da origem, ou seja, (x,y,z) = (0, 0, 0). Na sequência são apresentadas cem linhas de cálculo,

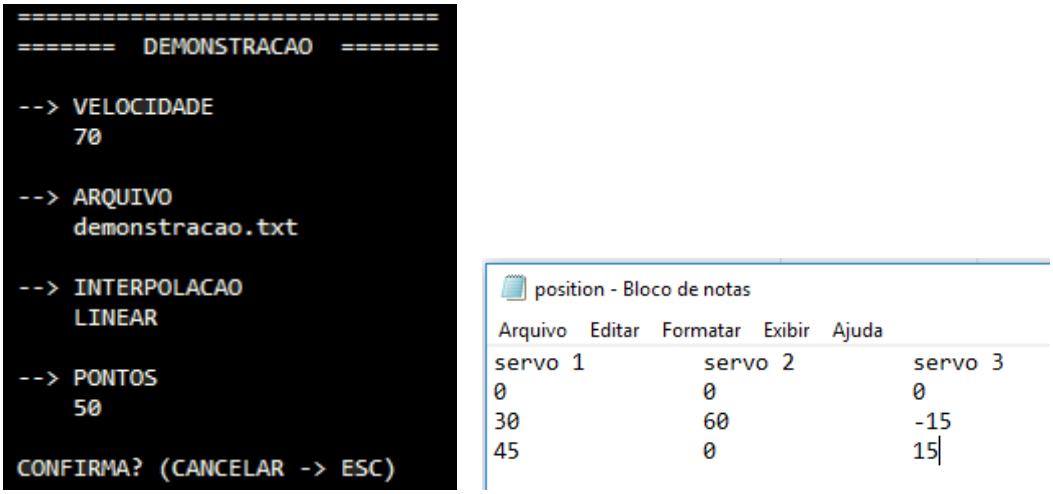
- os três primeiros valores de cada linha, serão os ângulos aplicados nas juntas,
- o quarto e o último valor, representam se o robô pode atingir este ponto da trajetória, caso o valor seja igual a zero significa que o manipulador não poderá atingir este ponto, caso contrário o ponto estará dentro do espaço de trabalho do robô,

- o quinto valor representa a distância da trajetória que o robô percorrerá e a trajetória desejada,
- o sexto valor é a quantidade de iterações que foram realizadas para encontrar o valor que será aplicado nos motores. Como podemos observar, e faz todo sentido, ao se deparar com pontos inatingíveis pelo robô, o algoritmo realiza muitas iterações para encontrar o ponto mais próximo que seja atingível pelo manipulador, dessa forma os maiores valores da iteração, são obviamente em posições que o retorno é igual a zero.

O seguinte modo de funcionamento, DEMONSTRACAO, é simplesmente uma função implementada para a demonstração do robô. As coordenadas são lidas diretamente de um arquivo de configuração, dessa forma, o robô executa uma gama de movimentações sem a necessidade de um usuário estar operando.

A Figura 36 (a) apresenta a tela de configuração desse modo. Primeiramente o usuário escolhe a velocidade, em percentagem, dos motores, no segundo comando deve ser inserido o arquivo no qual está descrita a movimentação do robô, a terceira opção se trata do tipo de movimentação que irá ser executada entre os pontos contidos no arquivo, neste caso há duas opções NORMAL, onde a movimentação é semelhante ao modo CINEMATICA DIRETA, e a segunda é a LINEAR a qual faz uso da função CINEMATICA INVERSA, ou seja, chamando a biblioteca externa, a quarta opção apenas surge se a interpolação for linear, neste caso é a quantidade de pontos que o algoritmo deve criar entre dois conjuntos de ângulos descritos no arquivo.

A Figura 36 (b) apresenta como o arquivo “txt” utilizado nesta função deve ser configurado. A Figura 37 mostra a tela de execução deste modo, nela podemos observar para quais ângulos os motores estão se movendo, no caso de interpolação linear, também é apresentado em qual ponto a movimentação atual está e logo abaixo a posição seguinte.



(a) (b)

Figura 36 - Tela Driver Demonstração e Arquivo de Configuração.

Fonte: Arquivo Pessoal

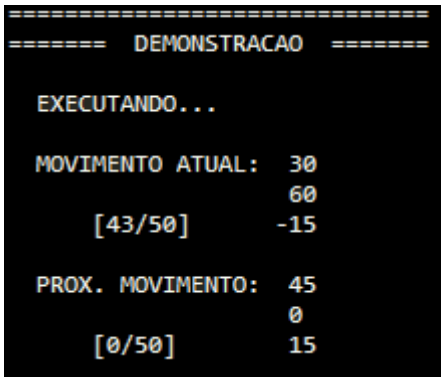


Figura 37 - Tela Driver Execução Demonstração.

Fonte: Arquivo Pessoal

A quinta opção chamada de SPLINES, é uma forma melhorada de movimentação da opção anterior, nela é possível configurar o tipo de movimentação que se deseja obter do robô. Além dos ângulos é possível também escolher a velocidade de início e fim de cada movimento, a Figura 38 mostra a configuração que o arquivo de entrada para esta função deve ter e a tela da opção.

A opção seguinte trata-se do modo CONFIG AUTOMATICA, este é o modo com o qual é realizada a configuração automática dos parâmetros dos controladores de cada servo motor. Após iniciar o modo o driver irá executar comandos de posicionamento para os motores e tentará encontrar uma sintonia para os motores

que atenda a especificação. Importante mencionar que tanto a especificação quanto os ângulos dessa movimentação estão pré-definidos no driver, ou seja, não há, pelo menos por enquanto, como alterar esses valores. A Figura 39 apresenta a tela para esta função.

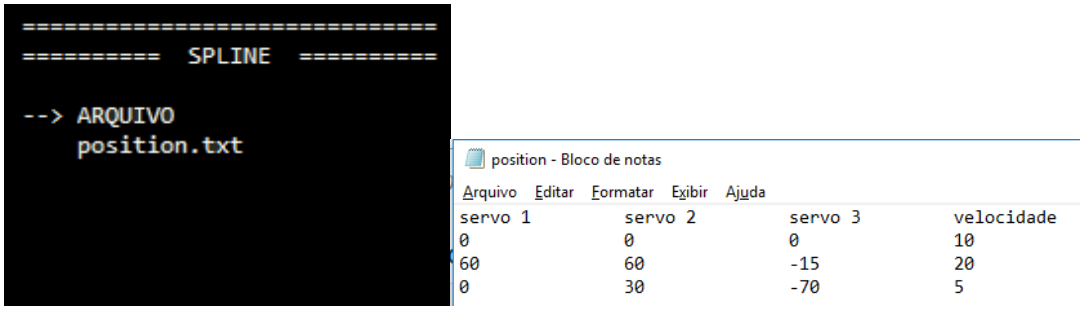


Figura 38 - Tela Driver Spline e Arquivo de Configuração.

Fonte: Arquivo Pessoal

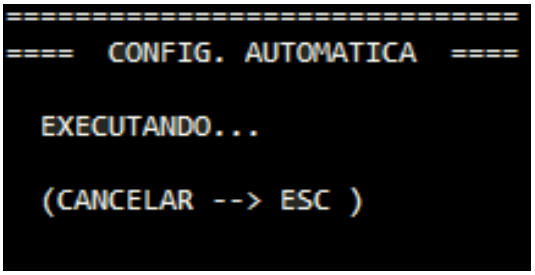


Figura 39 - Tela Driver Configuração Automática.

Fonte: Arquivo Pessoal

A próxima opção é chamada de AMOSTRAGEM, neste modo de funcionamento, o manipulador tem o torque desligado, dessa forma é possível manipular o robô manualmente, desse modo o driver começa a ler e gravar a posição dos *encoders* de cada junta, fazendo assim um registro para que possa ser executado posteriormente. Resumindo, o robô irá executar o mesmo movimento que o usuário realizou. As configurações deste modo são as seguintes, em arquivo o usuário deve inserir o nome que a função registrará os dados, contínuo ou amostra significa que, se for contínuo o driver irá ler os *encoders* continuamente, sempre que o robô se movimentar, e com a opção amostra, o usuário irá precisar pressionar a tecla “*enter*”

do teclado para que o driver registre a posição atual do robô. Caso a opção tempo esteja selecionada o driver também irá registrar o tempo de execução, caso contrário quando executar o arquivo criado apenas a movimentação é garantida o tempo poderá ser diferente do amostrado. A Figura 40 abaixo apresenta a tela do modo de amostragem e um trecho do arquivo criado com o modo.

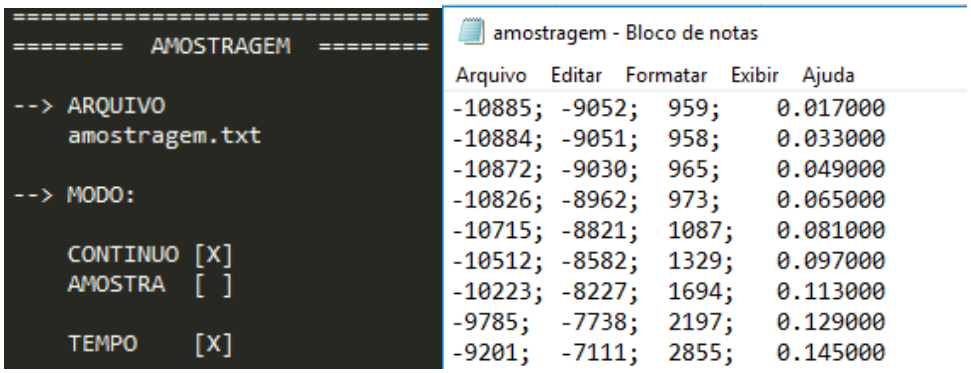


Figura 40 - Tela Driver Amostragem e Arquivo de Configuração.

Fonte: Arquivo Pessoal

Por fim, a opção CONFIGURACAO leva a uma segunda tela onde várias configurações do motor podem ser realizadas. Nessa segunda tela é possível alterar vários parâmetros como velocidade, aceleração desejada, angulação limite para os motores, opções de torque e etc. A Figura 41 apresenta a tela de configurações os números ao lado de cada opção representa o que o usuário deve digitar para selecionar a opção.

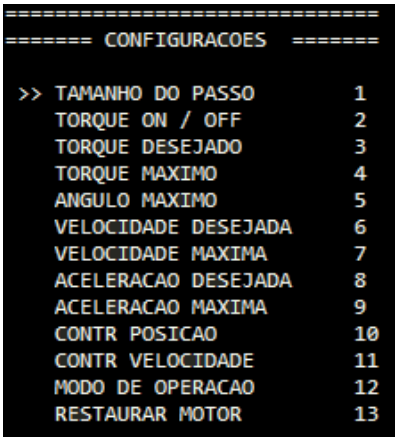


Figura 41 - Tela Driver Configurações.

Fonte: Arquivo Pessoal

6.3 – Desenvolvimento API

Como já apresentado, a API neste projeto fará a ponte entre o driver de comunicação e uma biblioteca desenvolvida em paralelo por outro colaborador da empresa.

A API faz uso de funções para a construção de um modelo virtual do robô que é utilizado para cálculos matemáticos de cinemática inversa, ambos já apresentados na terceira opção do menu principal, na seção anterior. A Figura 42, exemplifica a troca de informações para a realização da cinemática inversa, mas essa comunicação não é utilizada exclusivamente para a cinemática inversa, ela também ocorre logo no início da execução do software que faz a leitura de arquivos de configuração, previamente construídos, com informações de comprimento de elo, quantidade de motores, eixos que cada motor interfere e etc. O modelo virtual é criado com a função descrita na Figura 43 que são retirados de um arquivo de configuração apresentado na Figura 44.

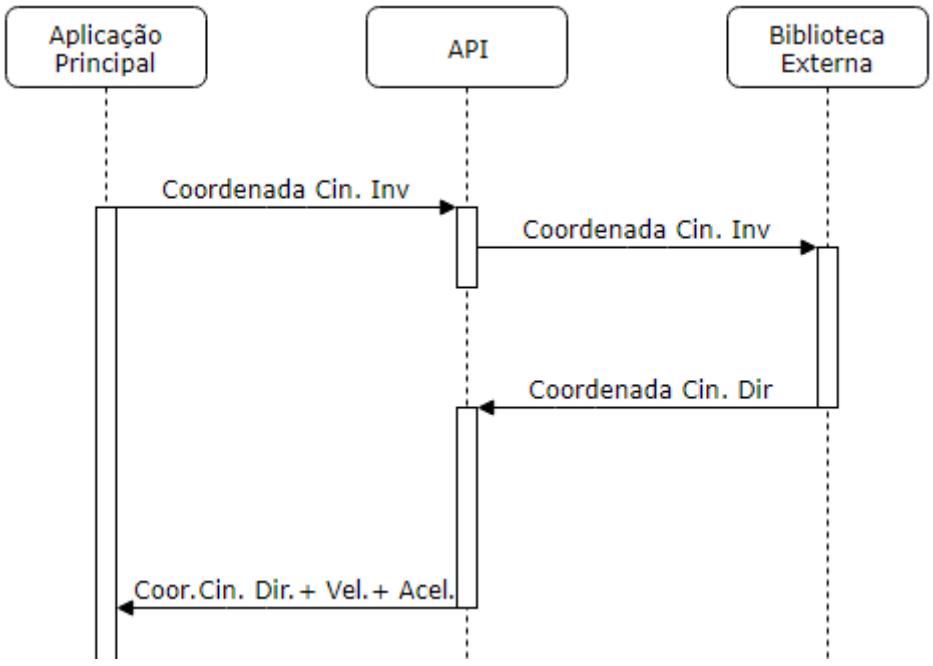


Figura 42 - Troca de Informações Entre Aplicação, API e Biblioteca Externa.

Fonte: Arquivo Pessoal

```
SnakeModel createSnakeModel(double L1, double L2, double L3, double minAngle, double maxAngle){
    srand(time(0));
    SnakeModel model = {L1, L2, L3, minAngle, maxAngle, 1e-3, 0.5, 1000};
    return model;
}
```

Figura 43 - Código Responsável por Criar Modelo Virtual do Robô.

Fonte: Arquivo Pessoal

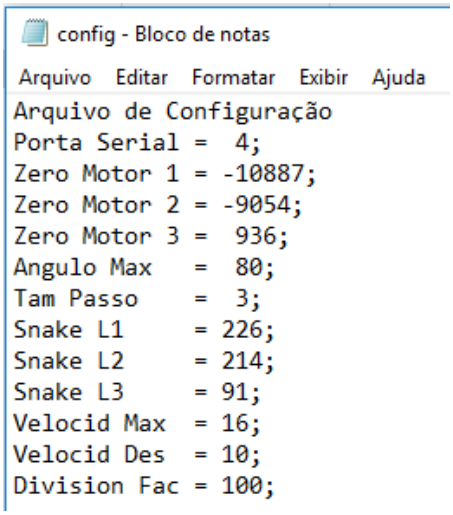


Figura 44 - Arquivo Configuração do Driver de Comunicação.

Fonte: Arquivo Pessoal

Após essa inicialização, o programa faz uso da API para, como já comentado, cálculos da cinemática inversa que funciona da seguinte forma, dada uma determinada coordenada desejada para o robô, na aplicação principal, a API faz uso da biblioteca externa, passando os valores de coordenadas e, posteriormente, recebendo os dados de movimentação para o robô, mas agora os dados retornados são ângulos os quais serão repassados para os motores fazendo a movimentação do mesmo através da cinemática direta.

É importante ressaltar que antes dos dados serem repassados para a aplicação principal, eles recebem um tratamento, dentro da API, onde serão adicionados velocidades e acelerações para que a movimentação se torne mais suave, contínua e para que todos os eixos trabalhem de forma coordenada.

6.4 – Plataforma de Testes

A plataforma de testes foi desenvolvida ao longo de todo o projeto. Foi a primeira tarefa a se iniciar e também a última a ser encerrada. Seu objetivo é de proporcionar uma interface gráfica amigável para o usuário do sistema, apresentar informações importantes do status do robô, testar movimentações e principalmente deveria possuir a capacidade de realizar testes com robôs diferentes do modelo físico trabalhado neste projeto. Este último requisito se deve tanto pelo fato de que o robô atualmente testado não ser a versão final, bem como para a plataforma se adaptar para outros projetos.

Esta seção foi dividida em duas partes, primeiramente será apresentado o software, telas e funcionalidades disponíveis e após isso, será apresentada a arquitetura de algumas funções, bem como o fluxo de dados software x manipulador.

6.4.1 – Telas do Ambiente Virtual

A tela apresentada na Figura 45, é responsável por demostrar os dados dos motores que podem ser incluídos ou excluídos da tabela, de acordo com a necessidade do usuário. Quando pressionado o botão de adicionar, o botão pode ser visualizado na Figura 46, é apresentado em tela uma janela com várias opções, dessa forma basta que o usuário selecione as informações desejadas. Também há uma representação do manipulador, que executará os mesmos movimentos do robô real. Esta aba do software implementa uma função análoga ao movimento manual do driver de comunicação, porém, aqui mais de um eixo do robô pode ser movimentado ao mesmo tempo, isso se deve a programação concorrente existente na programação do ambiente.

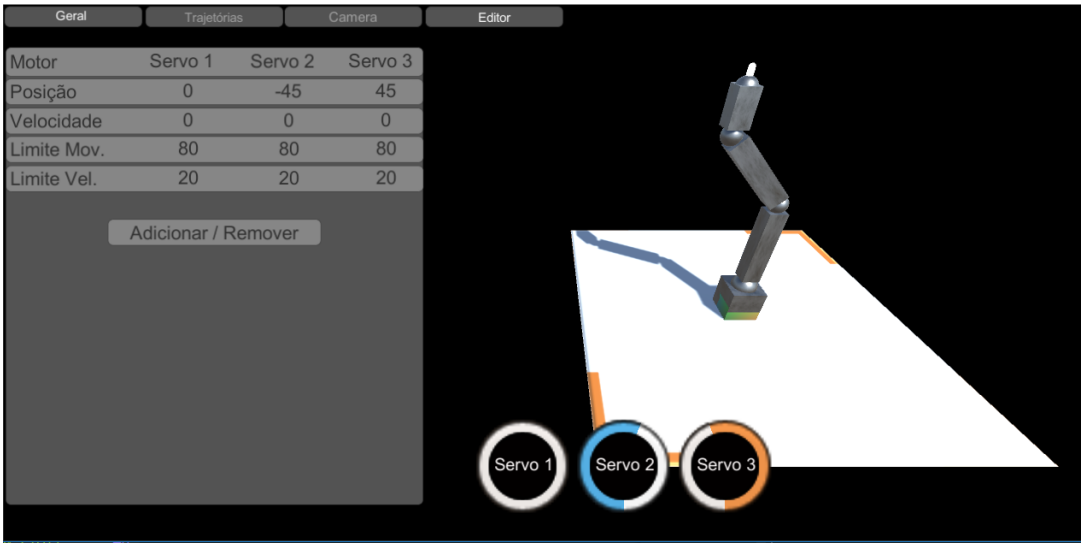


Figura 45 - Tela aba Geral.

Fonte: Arquivo Pessoal

Há duas maneiras de passar comandos para os motores nesta aba, através do teclado, assim como no driver. A segunda opção é diretamente em tela, os três círculos na região inferior representam o ângulo atual. Para tal, basta clicar no botão correspondente ao motor que se deseja movimentar e arrastar o mouse para o lado esquerdo para diminuir o ângulo e para o lado direito para aumentar. O círculo possui três cores de representação, todo branco significa ângulo no motor igual a zero, a cor laranja representa ângulos positivos e azul negativos, portanto as cores laranja e azul não podem coexistir.

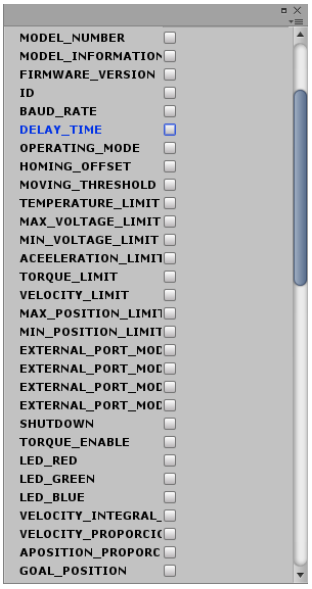


Figura 46 - Opções de Visualização dos Dados dos Motores.

Fonte: Arquivo Pessoal

A tabela como já dito, pode ser configurada, de acordo com o desejo do usuário, para acompanhar alguns dados do motor, mas esta tabela também permite que o usuário envie comandos para o robô. Ao se alterar um valor da tabela, ela dispara uma flag e requisita que um comando seja enviado para o motor em questão.

A aba seguinte apresenta a tela onde são criadas as trajetórias para o robô, Figura 47. Esta tela possui dois menus além da visão do robô, no menu superior há duas abas a primeira relacionada a criação de trajetória e a segunda serve para inserção de obstáculos no ambiente do manipulador.

Os tipos de trajetória disponíveis atualmente são, linha, curva de Bezier e curva Splines. O segundo menu apresenta as curvas inseridas, e posteriormente elas poderão ser executadas. A Figura 48 mostra o segundo menu com duas curvas configuradas, ao lado é possível visualizar essas curvas, primeiramente a linha reta, está ligada ao manipulador e depois uma curva *spline*. A Figura 49 além de mostrar as curvas comentadas, há no fundo uma nova curva que ainda não foi editada, ou seja, ela acabou de ser inserida no ambiente, no lado direito da imagem são apresentadas as janelas de configurações de cada tipo de trajetória. Linha aceita duas coordenadas como entrada, que pode ser editada nesta janela ou simplesmente arrastar seus pontos para o espaço desejado. A segunda é a splines, essa é a única que não aceita receber parâmetros através do teclado, caso a *checkbox* não esteja

selecionada a curva terá um começo e um fim em pontos distintos, caso contrário ela será uma cura fechada, um *loop*.

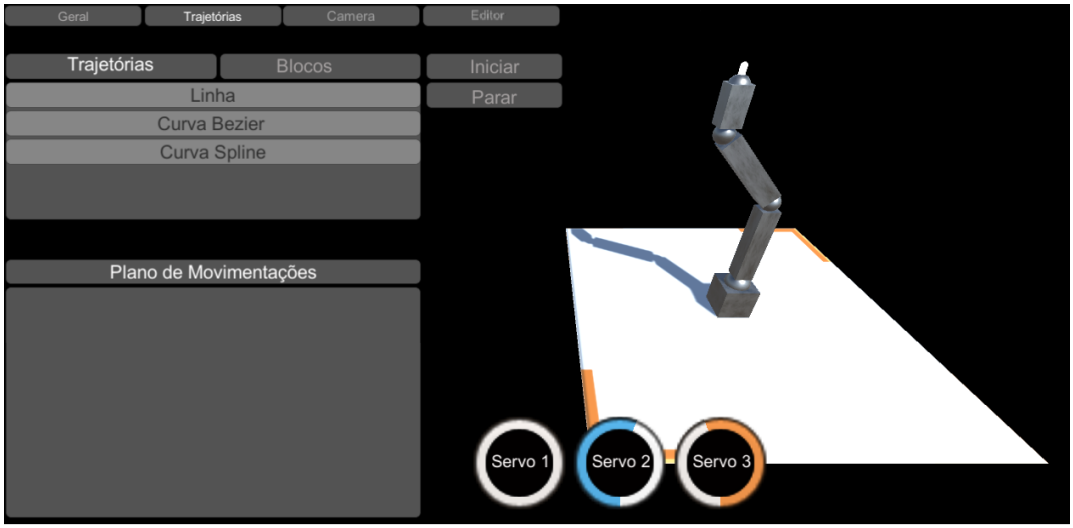


Figura 47 - Tela aba Trajetórias.

Fonte: Arquivo Pessoal

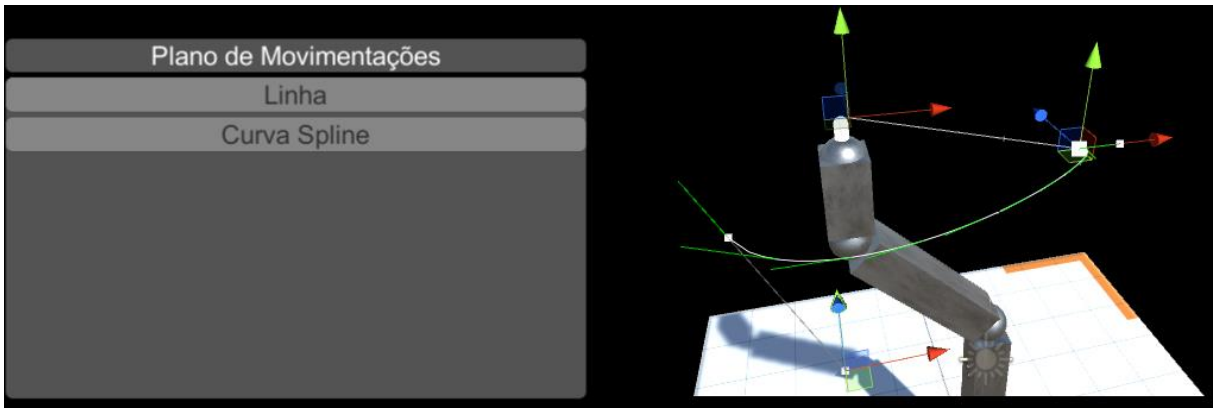


Figura 48 - Robô com Trajetórias.

Fonte: Arquivo Pessoal

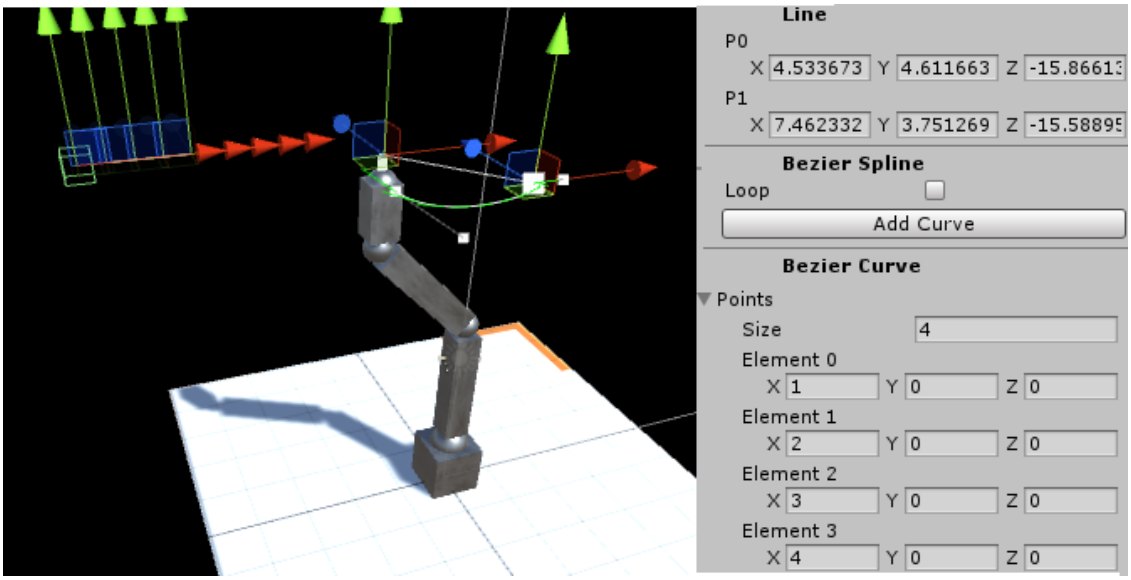


Figura 49 - Configuração de cada tipo de Curva.

Fonte: Arquivo Pessoal

Voltando agora para o menu superior da janela de trajetórias, mais especificamente para a aba chamada blocos. Nesta aba, é possível fazer a inserção de objetos na área de trabalho do robô. Clicando nas opções o objeto 3D é criado na tela do usuário e após isso ele pode dimensionar o objeto e definir uma textura para o mesmo, a Figura 56 apresenta as texturas disponíveis por enquanto. A finalidade dos objetos, por enquanto, é apenas para simular melhor o ambiente de trabalho do manipulador, mas já existe um trabalho sendo realizado para que se encontre um caminho alternativo para uma trajetória desejada sem que ocorra colisões, tudo isso de forma automática. A Figura 50 mostra a área de trabalho do robô com um bloco.

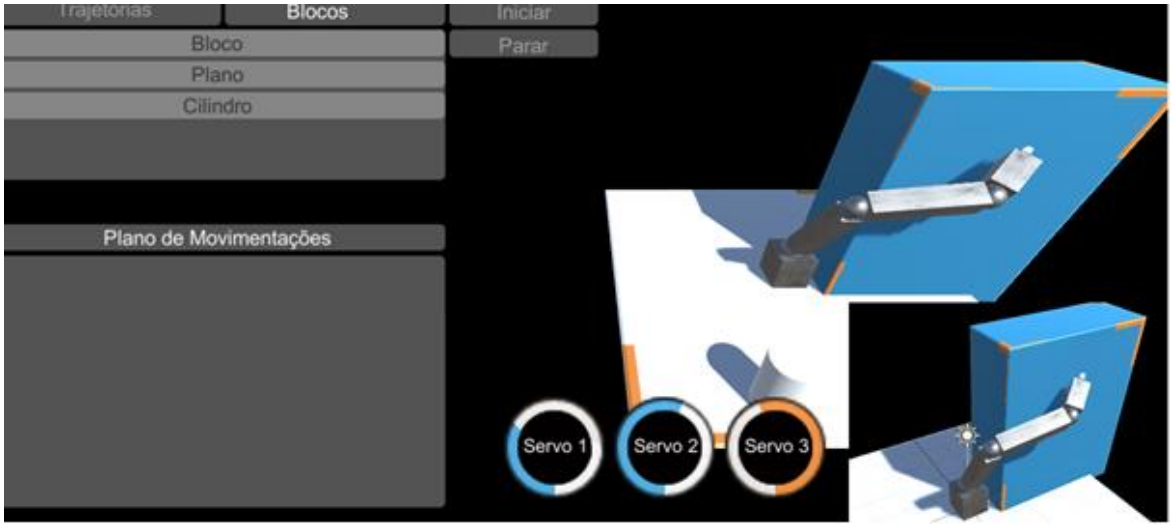


Figura 50 - Bloco Inserido na Área de Trabalho do Robô.

Fonte: Arquivo Pessoal

Figura 51 – Tela aba Câmera.

Fonte: Arquivo Pessoal

A quarta aba chamada câmera foi implementada para demonstração do robô, por exemplo, após iniciar uma movimentação pode-se alterar para esta tela e visualizar o robô com mais facilidade já que não há menus na tela, como pode ser visto na Figura 51, outra funcionalidade, agora não apenas para esta aba e sim para o software todo, é a possibilidade de abrir câmeras auxiliares.

Novamente na Figura 51, é possível visualizar a existência de três câmeras adicionais na lateral esquerda da tela. Esta opção foi implementada, principalmente para facilitar a visualização e configuração das trajetórias que dependendo do ângulo podem passar uma impressão errada, como pode ser visto na Figura 52, onde a imagem da esquerda aparenta mostrar uma trajetória plana, porém com o auxílio da imagem da direita, que poderia ser uma outra câmera, é possível ver que se trata de basicamente uma inclinação.

Na última aba, chamada de editor, é realizada a “construção” de manipuladores robóticos. Similar a aba de trajetórias, está também possui dois menus dispostos no lado esquerdo. Neles é possível selecionar o objeto desejado, conforme descrição no próprio botão, e posteriormente edita-lo para que atenda ao requisito desejado. Ao selecionar um objeto no manipulador são apresentados os demais blocos que são “dependentes” do selecionado.

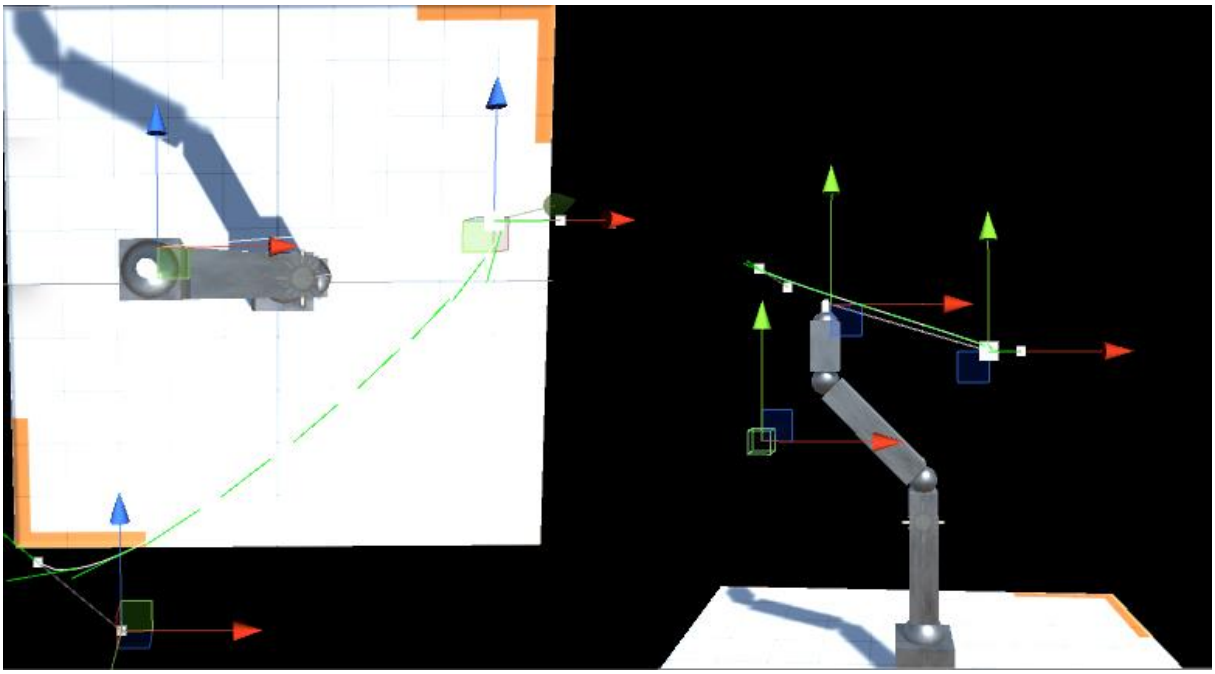


Figura 52 – Ângulos Diferentes de Visão de Uma Mesma Trajetória.

Fonte: Arquivo Pessoal

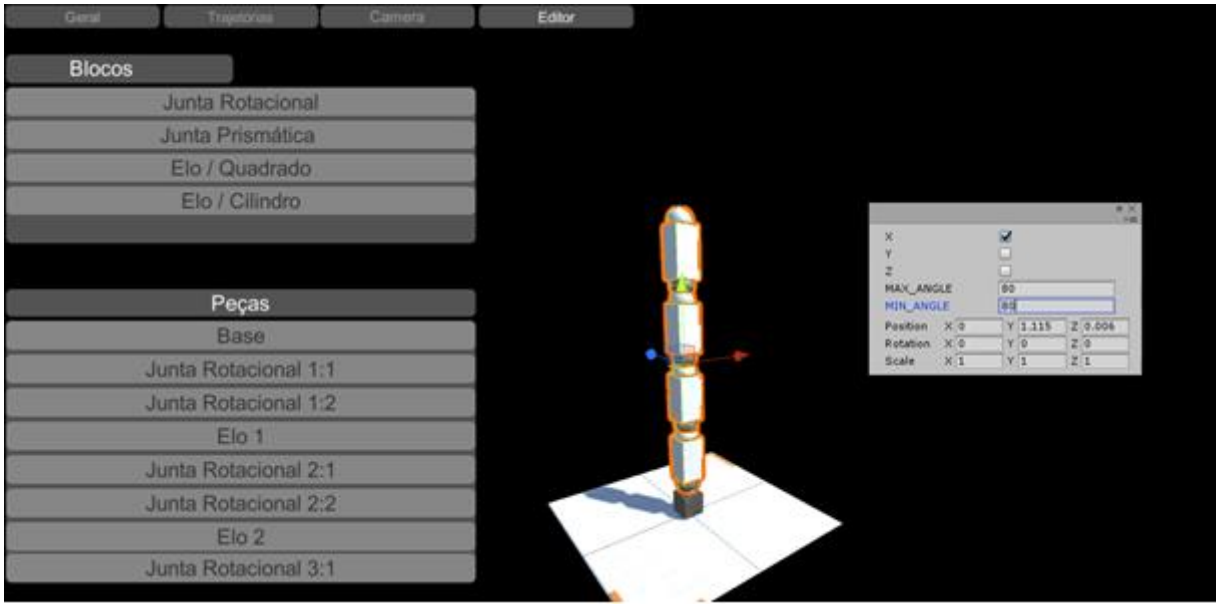


Figura 53 – Tela aba Editor com Manipulador Sendo Construído.

Fonte: Arquivo Pessoal

Por exemplo, ao selecionar uma junta rotacional, o resto da cadeia é “selecionada” automaticamente, pois, ao se movimentar aquela junta, o restante da cadeia é afetado, este efeito de seleção pode ser visualizado na Figura 53, além de selecionar o restante da cadeia, uma janela de configuração aparece para que o usuário, marque que tipo de movimentação a junta executará, quais os ângulos máximos e mínimos, tamanho, localização e orientação no espaço. No caso apresentado, a junta executará movimentos em torno do eixo x, com limite superior e inferior de 80 graus para o lado positivo e 80 graus para o lado negativo. A Figura 54, apresenta um robô com uma configuração de pose diferenciada, isso auxilia na construção do manipulador para que seja possível visualizar a sua alcançabilidade.

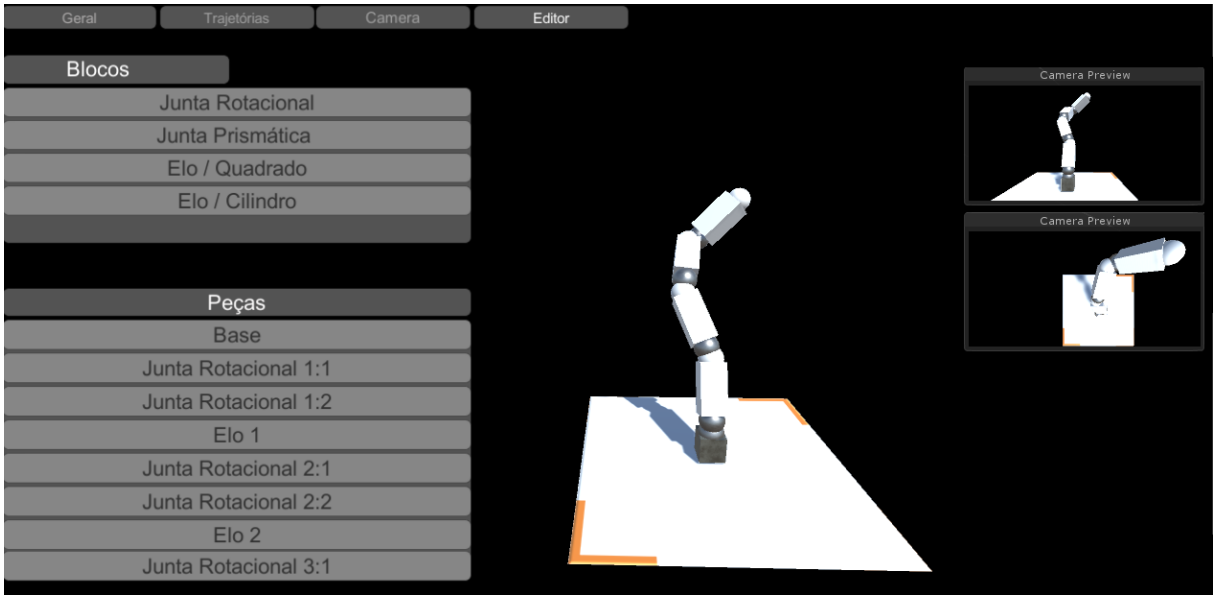


Figura 54 – Manipulador Sendo Editado.

Fonte: Arquivo Pessoal

Após o fim da edição, assim como nos objetos na aba de trajetória, é possível agregar as partes do robô texturas, como visto na Figura 55.

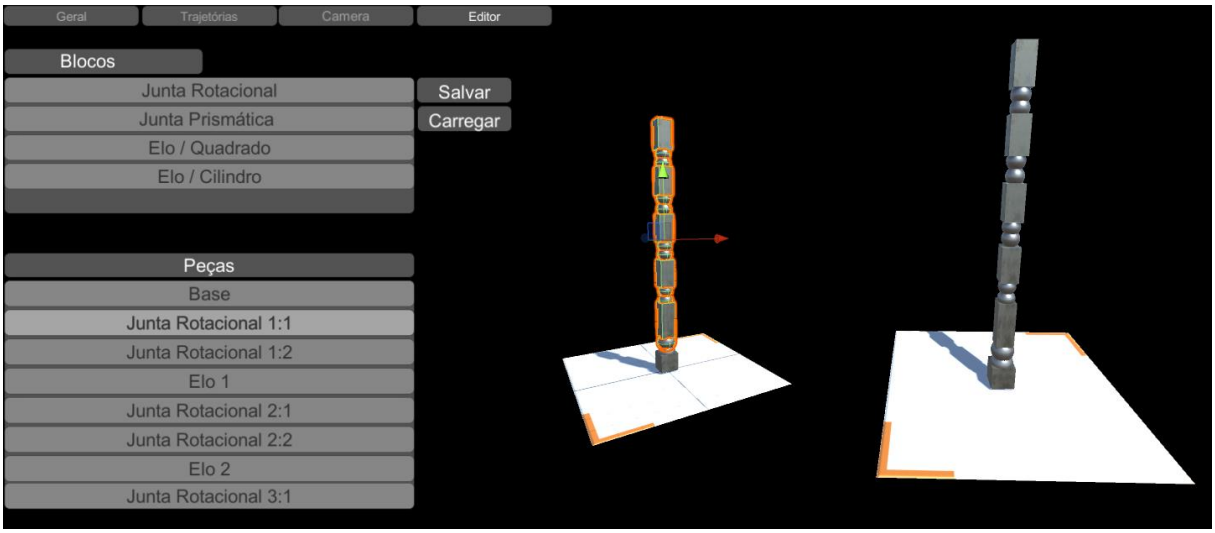


Figura 55 – Robô editado e texturizado.

Fonte: Arquivo Pessoal

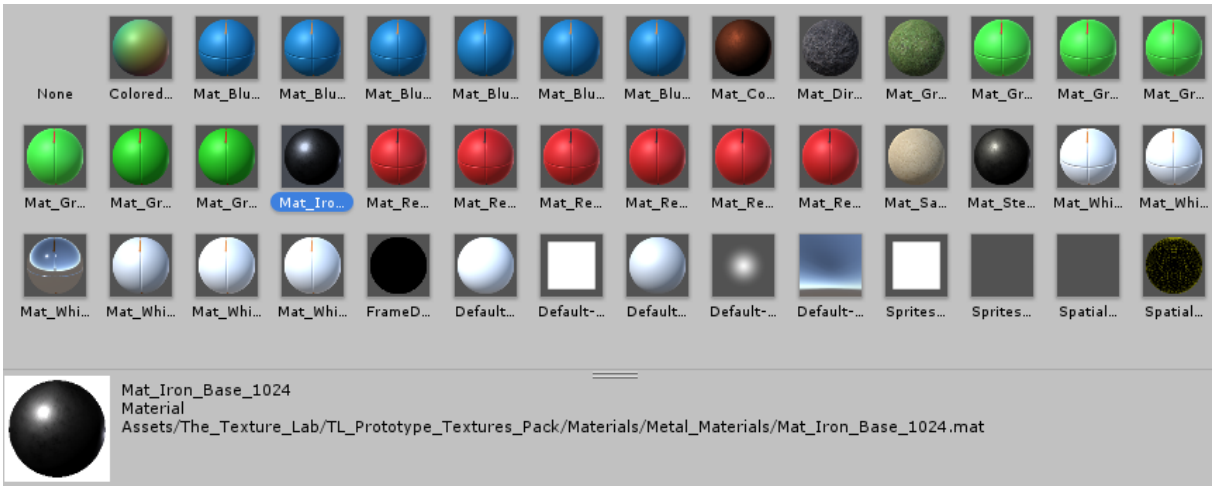


Figura 56 – Textura de Material Disponível para os Objetos 3D.

Fonte: Arquivo Pessoal

6.4.2 – Criação de Trajetórias

A geração de trajetórias é uma das tarefas mais importantes no Ambiente de Testes, pois são com elas que controlaremos todo o manipulador. Passando do mais básico, uma simples reta, para até atingir funções complexas de movimentação. A abordagem utilizada para a construção dessas trajetórias foi a curva de Bézier (Figura 57) que é uma curva polinomial expressa como interpolação linear entre alguns pontos de controle.

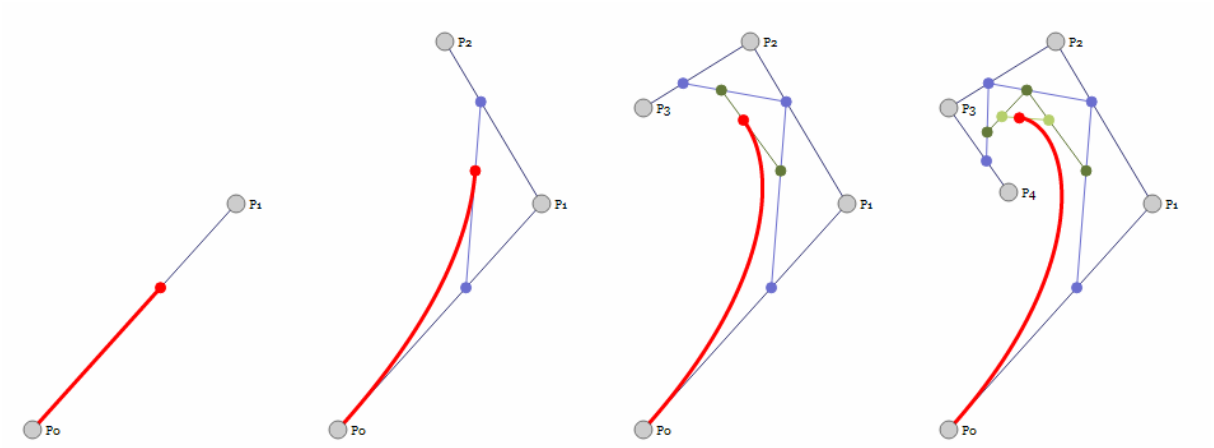


Figura 57 - Curvas de Bézier de ordens (a) 1; (b) 2; (c) 3; (d) 4.

Fonte: [72]

A classe *Editor* possui uma variável, que é definida para o objeto a ser desenhado quando *OnSceneGUI* é chamado. Dessa forma essa variável foi convertida em uma linha e, em seguida, desenhada unindo os dois pontos com a classe *Handles o* algoritmos pode ser visualizado abaixo Algoritmo 1.

```
private void OnSceneGUI () {
                Line line = target as Line;
                Handles.color = Color.white;
                Handles.DrawLine(line.p0, line.p1);
        }
```

Algoritmo 1 - Desenho de linhas

Agora os pontos devem ser convertidos em pontos espaciais, caso contrário não há como agregar a eles funções de movimento e giro, Algoritmo 2.

```
private void OnSceneGUI () {
                Line line = target as Line;
                Transform handleTransform = line.transform;
                Quaternion handleRotation = handleTransform.rotation;
                Vector3 p0 = handleTransform.TransformPoint(line.p0);
                Vector3 p1 = handleTransform.TransformPoint(line.p1);

                Handles.color = Color.white;
                Handles.DrawLine(p0, p1);
                Handles.DoPositionHandle(p0, handleRotation);
                Handles.DoPositionHandle(p1, handleRotation);
        }
```

Algoritmo 2 - Conversão para pontos espaciais

Com isso já era possível descrever rotas para o robô através de linhas retas. Agora já era hora de iniciar a programação das curvas, especificamente, uma curva Bézier. Uma curva de Bézier é definida por uma sequência de pontos. Começa no primeiro ponto e termina no último ponto, mas não precisa passar pelos pontos intermediários. São esses pontos que afastam a curva de ser uma linha reta.

Esta nova etapa primeiramente modificou o *script* de linhas para que aceitasse, inputs de quantidade de pontos e foi adicionada uma interface para que a manipulação dos pontos pudesse ser realizada de forma visual, arrastando os pontos no *workspace* ou então preenchendo os valores diretamente através do teclado. A Figura 58, apresenta a imagem da tela de inserção de pontos e uma interpolação dos vários pontos criados, ainda em linha reta.

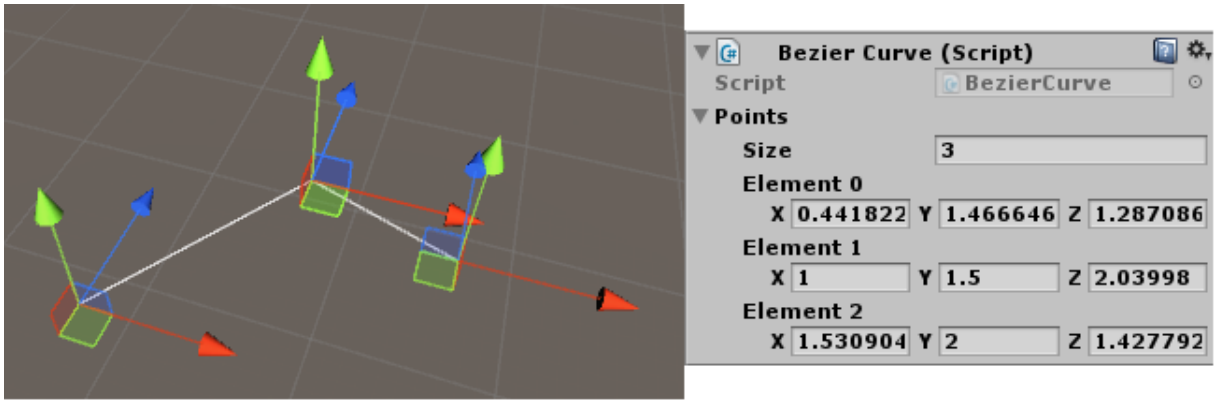


Figura 58 - Curva de Bezier Ainda em Construção, e sua Configuração.

Fonte: Arquivo Pessoal

As curvas de Bézier são paramétricas. Se você der um valor - normalmente chamado de *t* - entre zero e um, você obtém um ponto na curva, a localização deste ponto estará no início da linha se t for igual a zero e no fim da linha caso seja igual a um. Para tal, foi adicionada uma nova classe chamada de Bezier, apresentada no Algoritmo 3, com um método que realiza o cálculo para qualquer sequência de pontos. Esse método retorna o resultado para o *workspace*. A Figura 59 apresenta o resultado.

```
public static class Bezier {

	public static Vector3 GetPoint (
			Vector3 p0, Vector3 p1, Vector3 p2, float t) {
		return Vector3.Lerp(p0, p2, t);
	}
}
```

Algoritmo 3 - Classe Bezier

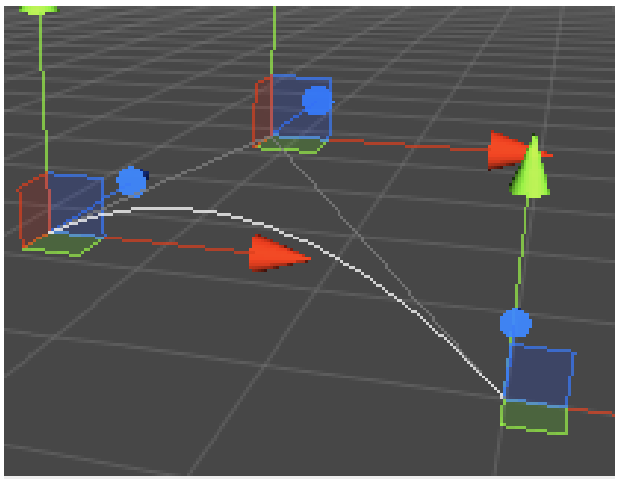


Figura 59 - Curva de Bézier.

Fonte: Arquivo Pessoal

Esse tipo de curva é conhecido como uma curva quadrática de Bézier, devido à matemática polinomial envolvida. A curva linear pode ser escrita como

$$B(t) = (1 - t)P0 + tP1, \quad t \in [0,1] \tag{34}$$

Um passo a mais e é possível obter a Bézier de ordem 2:

$$B(t) = (1 - t)\big((1 - t)P0 + tP1\big) + t\big((1 - t)P1 + tP2\big) \tag{35}$$

Basicamente continua sendo uma curva linear, porém, $P0$ e $P1$ são substituídos por outras novas curvas lineares. Uma outra maneira de escrever a função acima é

$$B(t) = (1 - t)^2P0 + 2(1 - t)tP1 + t^2P2, \quad t \in [0,1] \tag{36}$$

O método *GetPoint,* apresentado em Algoritmo 4, foi construído utilizando esta função. Ela pode ser observada abaixo acompanhada da função da primeira derivada e de velocidade.

```
        public static Vector3 GetPoint (Vector3 p0, Vector3 p1, Vector3
p2, float t) {
                t = Mathf.Clamp01(t);

                return (1f -t)*( 1f -t)*p0 + 2f*(1f -t)*t*p1 + (t*t)*p2;
                      }
```

Algoritmo 4 – Método GetPoint

Como podemos observar na Figura 59 fica difícil de visualizar a curva estando um em plano 3D, para auxiliar o usuário, foram desenvolvidas funções extras, uma para inserir na curva de trajetória linhas de direção, Figura 60, e um *script* dedicado a movimentação da câmera, chamado *CameraHandler*, apresentado no apêndice A. Ao fim da edição da curva é possível salvar a curva para posterior uso, simular apenas no ambiente ou executar juntamente no protótipo. Conforme explicado anteriormente.

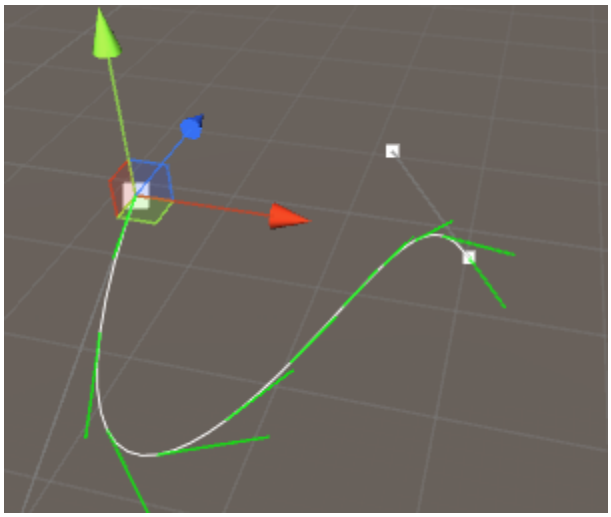


Figura 60 - Linhas de Direção em uma Trajetória.

Fonte: Arquivo Pessoal

6.4.3 – Fluxo de dados

Como pode se perceber as curvas geradas no ambiente de testes, além de todos os outros comandos, exigem uma grande quantidade de troca de informações. Esta subseção dedica-se a explicar como foi implementada essa comunicação. A Figura 61 apresenta os principais blocos de comunicação.

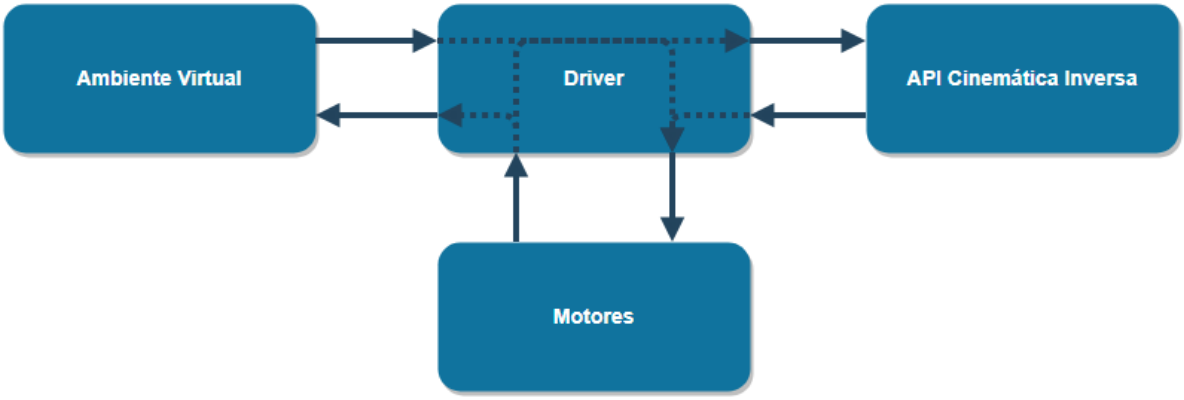


Figura 61 - Blocos de Comunicação.

Fonte: Arquivo Pessoal

Toda a comunicação é controlada pelo *driver*, dessa forma é neste bloco que acontece todo o encadeamento do fluxo dos comandos e respostas. O protocolo possui quatro níveis de prioridade, são eles em ordem de maior para menor prioridade, Controle, Comandos e Respostas.

Mensagens com prioridade “controle” são criadas apenas em movimentações mais complexas, como trajetórias. Elas exigem uma maior parte do processamento, devido ao fato dessas instruções necessitarem de todo um controle temporal e de posicionamento atual dos motores para que, assim, o *driver* conheça quando e qual comando enviar para os motores. A Figura 62, exemplifica essa comunicação. Quando o *driver* recebe uma mensagem com esta prioridade, ela era executada sem interrupções para que fosse garantida todo o controle do movimento, mas isso inseria alguns problemas, principalmente porque a interface gráfica do robô não era atualizada, pois o *driver* não executava os comandos de atualização da plataforma, pois esses comandos possuem prioridade “resposta”. Dessa forma, para manter o robô 3D atualizado graficamente com o robô real, foi implementada uma interrupção,

que executaria o envio do mesmo dado recebido para executar o *feedback* do controle da trajetória, com isso executamos uma solicitação aos motores e duas tarefas são alimentadas.

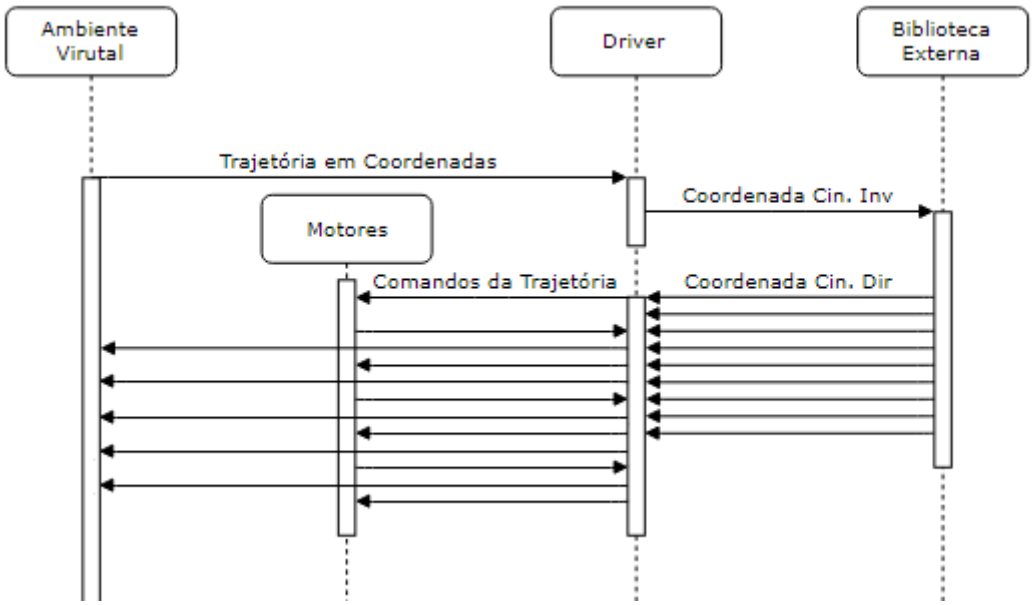


Figura 62 - Comunicação Proveniente de uma Solicitação de Trajetória.

Fonte: Arquivo Pessoal

As prioridades de “comandos” e “respostas” são muito semelhantes, mas, ainda assim, a implementação da preferência na execução para prioridade “comando”. Mensagens com prioridade “respostas” são basicamente atualizações gráficas, geralmente perguntas para os motores do tipo, querendo conhecer sua velocidade, ângulo ou então se está se movimentando. Por isso, respostas possuem a menor prioridade. Mensagens do tipo, comando, são geralmente configurações aos motores, como alteração de angulação máxima, quantidade de torque, velocidade e etc.

As várias mensagens são enviadas para o *driver* o qual se encarrega de encadeá-las em uma fila de prioridades. Cada elemento da fila é uma requisição do *driver* ou das aplicações ligadas a ele. A Figura 63 apresenta uma fila de prioridades duplamente encadeada, assim como foi implementada no *driver*.

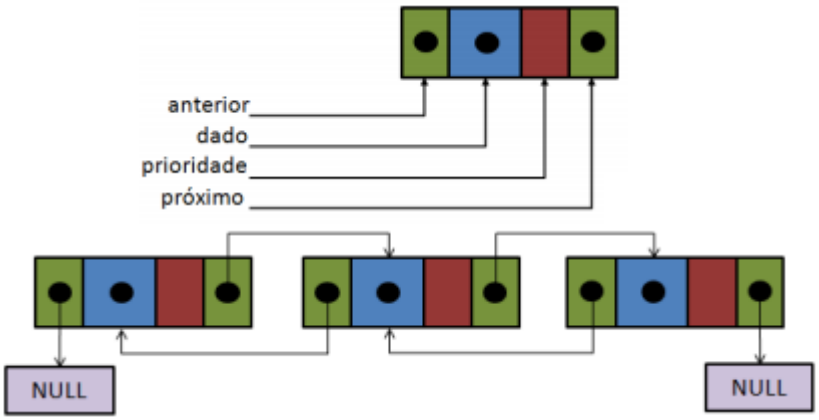


Figura 63 - Fila de Prioridades Duplamente Encadeada.

Fonte: [73]

7 - RESULTADOS

Os testes tanto do *driver* de comunicação como o ambiente virtual de testes foram realizados ao longo de toda a implementação, mas, mesmo assim, houve uma tarefa específica para isto, depois de todo o desenvolvimento.

O resultado para o driver e Ambiente de Testes podem ser visualizados ao longo de todo capítulo 6, todas as telas e resultados apresentados foram obtidos durante essa tarefa. Para testar o sistema como um todo, manipulador físico também, o autor deste trabalho contou com a ajuda de colegas do Instituto SENAI de inovação em Sistemas de Manufatura e Processamento Laser. É importante mencionar que eles fizeram parte do projeto, mas trabalharam no desenvolvimento estrutural da cadeia aberta do robô. Estes testes se basearam na precisão e repetibilidade do posicionamento do efetuador final. A Figura 64 mostra a área de trabalho do robô.

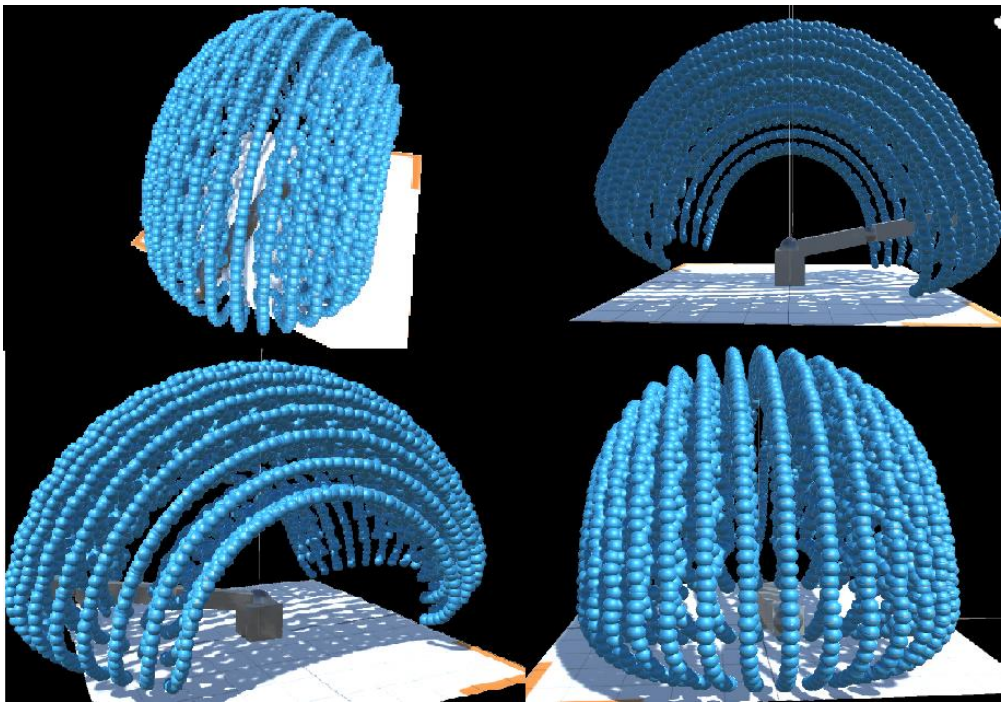


Figura 64 - Área de Trabalho do Robô Desenvolvido.

Fonte: Arquivo Pessoal

O manipulador robótico foi fixado dentro de uma fresadora, modelo Hermle C42U. Ao realizar um comando de posicionamento para o robô verificava-se a posição do efetuador final com a ferramenta da fresadora, dessa forma, obtém-se a precisão e posicionamento do robô. Para os testes de força, utilizou-se o sensor de força da

fabricante Kistler Instruments, modelo 9256C2, Figura 66. Novamente, os testes foram realizados dentro da fresadora, mas desta vez a ferramenta estava conectada o sensor de força. Ao se posicionar a ferramenta em coordenadas específicas, o manipulador foi alocado para praticamente as mesmas posições, com objetivo de posicionamento no “interior” do sensor, desta forma ele ao encostar na superfície do sensor precisava se movimentar mais um pouco para atingir o objetivo, com isso o controle aumentava o torque para completar a movimentação e exercia a força que era medida. A Figura 65 apresenta o sensor de força acoplado na fresadora.

Figura 65 - Sensor de Força (Direita) Robô Aplicando Força (Esquerda) no Sensor.

Fonte: Arquivo Pessoal

Figura 66 - Sensor de Força Utilizado.

A tabela a seguir complementa a Figura 67, servindo de legenda para o resultado dos testes de posicionamento do efetuador final.

	Atende Totalmente a Expectativa do Projeto	
Cor	Repetibilidade (< 0,05mm)	Precisão (< 0,1mm)
Verde	Sim	Sim
Branca	Não	Sim
Azul	Sim	Não
Vermelha	Não	Não

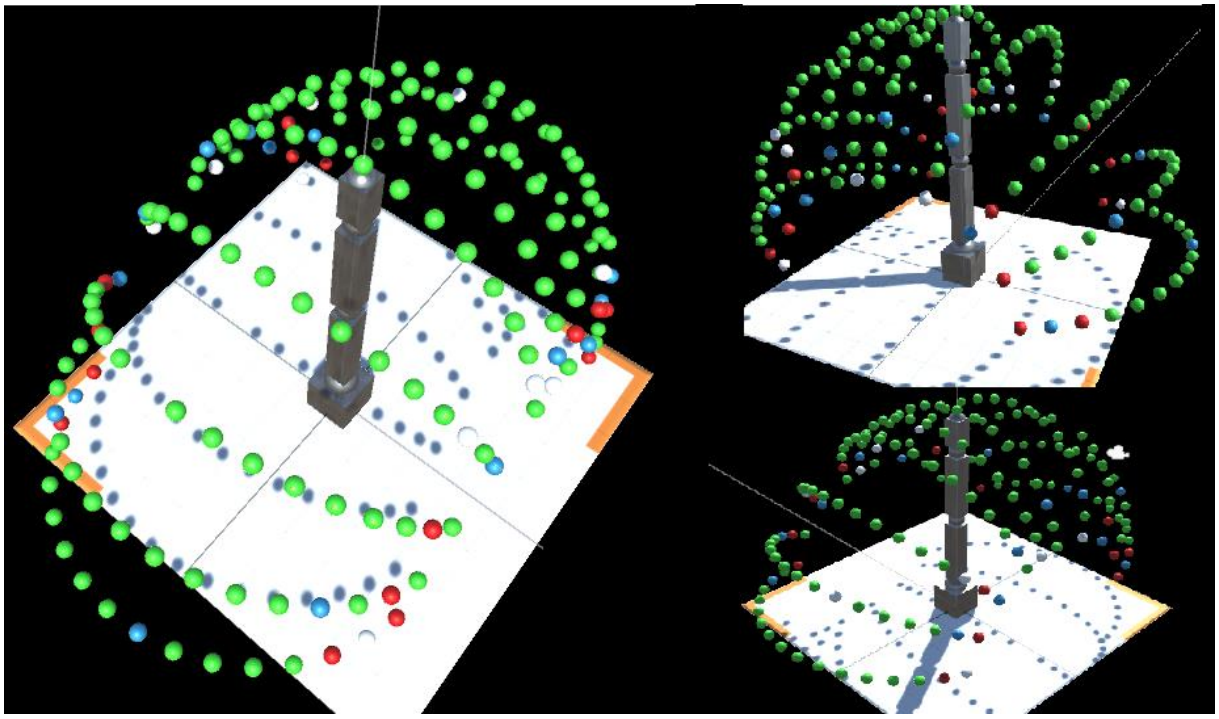


Figura 67 - Resultado dos Testes de Posicionamento.

Fonte: Arquivo Pessoal

Durante os testes o sistema se mostrou bastante estável. Houve ainda uma fase de testes com integrantes externos ao projeto, onde utilizavam o sistema e avaliar a facilidade de utilização e simplicidade da interface, algumas dificuldades foram apresentadas no editor de manipuladores, mas o problema era maior em usuários sem uma noção básica em robótica, o que dessa forma era esperado apresentarem dificuldades em abstrair robô para simples blocos, como é o caso da atual versão do editor. De modo geral, o sistema comportou-se como esperado, sem apresentar

maiores problemas, atendendo os requisitos iniciais para a confecção desta prova de conceito.

8 – CONSIDERAÇÕES FINAIS

Este capítulo dedica-se a fazer uma síntese dos resultados obtidos com a solução desenvolvida, tanto em relação aos requisitos de projeto, como em relação aos objetivos do trabalho.

8.1 – Aos Requisitos de Projeto

Todos os requisitos foram atendidos pela solução apresentada ao final do projeto com exceção de alguns pontos na área de trabalho do manipulador, como apresentado anteriormente. Em relação ao posicionamento dos motores, a precisão para aplicações desejadas a serem desenvolvidas na empresa é suficiente, porém, o que foi estabelecido pela equipe não foi atendido totalmente, há pontos em que a precisão da movimentação ultrapassa a meta que seria de $\pm 0{,}1\ mm$ e repetibilidade de $\pm 0{,}05\ mm$. Porém, como analisado anteriormente, o manipulador ainda se trata de um protótipo e sua implementação sofrerá modificações, refinamento no controle e, possivelmente, em sua estrutura e etc. Além do que com a inserção de mais cinco graus de liberdade, adicionaram ao manipulador uma flexibilidade na hora do posicionamento da ferramenta, dessa forma, é possível evitar pontos onde o robô apresenta imprecisão do movimento.

Em relação a movimentação a velocidade atingida com o robô atualmente, em trajetórias complexas e contínuas foi superior a 120º/s, em cada junta, o que já atende perfeitamente os requisitos do projeto, pois robôs industriais possuem velocidades semelhantes em suas juntas finais e nas iniciais a velocidade é menor, a grosso modo a média se iguala tornando o protótipo um competidor a altura.

Os testes iniciais superaram a capacidade esperada, atendendo a expectativa e superando muito a demanda de 15kg no efetuador final, para o protótipo, esse esforço excessivo causou uma folga no mecanismo que liga os motores a estrutura e agora a estrutura será projetada para uma carga maior. É importante mencionar que está “avaria” na estrutura foi projetada, ou seja, a equipe responsável pelo projeto criou o mecanismo de conexão com o objetivo de romper se as forças atingissem o limite de segurança, tudo isso para evitar danos nos motores.

A implementação do software tanto de comunicação com os motores (driver de comunicação) como o ambiente de simulações, superaram a expectativa de toda a

equipe. Em relação a monitoração de variáveis de segurança e controle, é efetuada de forma dinâmica. O usuário pode escolher que tipo de valor deseja observar durante o processo de trabalho, essa abordagem se mostrou bastante útil, uma vez que a amostragem de todas as variáveis de uma só vez, causa um pequeno gargalo na atualização do modelo virtual do robô. Outro fator positivo nisso é que a apresentação de tantos dados em tela causa confusão e a sensação de uma tela “poluída”.

Embora ainda sem controle do usuário, um histórico é registrado a cada seção de utilização do robô. Ainda há melhorias que podem ser implementadas neste ponto, atualmente os dados escolhidos pelo usuário são registrados e arquivados acompanhado de um histórico das trajetórias realizadas. Uma implementação melhorada poderia abranger, não apenas dados escolhidos pelo usuário, mas também informações críticas, como temperatura por exemplo. Gráficos poderiam ser criados com esses dados para que fosse visualmente mais agradável e simples de fazer a inspeção dos arquivos. Limitar o acesso à plataforma, ou então, criar um *login* para cada usuário. Criar um banco de dados de informações de manutenção do robô, como manter um histórico do erro em relação as trajetórias desejadas, assim seria possível prever danos aos motores e / ou folgas nas juntas e acionar a manutenção necessária, ou mesmo criar rotinas de calibração para minimizar problemas com seguimento de referência dos motores.

8.2 – Aos Objetivos do Trabalho

Em relação ao objetivo do trabalho, citado na introdução, considerou-se que a solução desenvolvida, mesmo sendo apenas parte do manipulador, atende bem a necessidade do cliente, e tudo indica que, se a mesma qualidade for atingida na segunda fase do projeto, o manipulador será capaz de realizar as tarefas para qual está sendo implementado. As especificações quantitativas foram atendidas, ainda que apenas em um protótipo, mas há muitas melhorias que podem ser aplicadas no sistema, fisicamente e computacionalmente.

8.3 – Perspectivas futuras

Após a apresentação de entrega ao cliente, se iniciará uma nova fase no desenvolvimento, de uma forma básica, será nada mais que uma nova iteração no desenvolvimento com os mesmos passos da anterior. Um acréscimo de mais três motores já estava previsto para esta segunda etapa, mas há possibilidades de acontecer uma integração com mais cinco totalizando oito motores ao total.

O acréscimo de mais motores ao robô, estruturalmente falando, haverá grande trabalho, pois haverá todo um planejamento agora de se passar cabos pelo interior do robô para a efetuação das várias tarefas que ele poderá executar, Na versão atual, apenas cabos de alimentação e comunicação dos motores estavam presentes.

A programação do ambiente de simulação pouco será afetada com o acréscimo dos motores, sua implementação já levou em consideração esse acréscimo, no máximo, algumas mudanças na interface com a adição de campos para os novos motores.

Os algoritmos de movimentação sofreram uma mudança maior. Terão que supervisionar muito mais variáveis, resolver redundâncias e singularidades. Importante mencionar que está tarefa já foi iniciada, utilizando o editor de manipuladores presente no ambiente de testes, o robô atualmente rodando os novos testes de algoritmos é o mesmo apresentado na Figura 55.

REFERENCIAS

1. CURRIE, A. **The History of Robotics**.1999. Disponível em: <https://web.archive.org/web/20060718024255/http://www.faculty.ucr.edu/~currie/robooadam.htm>. Acessado 10 jun. 2018.

2. NEEDHAM, J. **Science and Civilization in China: Volume 2, History of Scientific Thought**".1991. Cambridge University Press.

3. ANGELOVA, K. **How Robots Have Evolved Over The Last 200 Years**. 2011. Disponível em: <http://www.businessinsider.com/robots-evolution-photos-2011-6?op=1>. Acessado em 15 abr. 2018

4. OXFORD DICTIONARIES. **robotics**. Disponível em: <https://en.oxforddictionaries.com/definition/robotics>. Acessado 4 abr. 2018.

5. AKINS, C. **5 jobs being replaced by robots**. Disponível em: <http://excelle.monster.com/benefits/articles/4983-5-jobs-being-replaced-by-robots >. Acessado 15 abr. 2018.

6. NOF, S, Y. **Handbook of Industrial Robotics**, 2. ed. John Wiley & Sons, 1999.1378 p.

7. SCIAVICCO, L; SICILIANO, B. **Modelling and Control of Robot Manipulators. Measurement Science and Technology**. Springer, 2000.

8. ROTH, B. **Principles of Automation, Future Directions in Manufacturing Technology**, Baseado no simpósio da Divisão de Engenharia e Pesquisa da Unilever realizado em Port Sunlight, Unilever Research, 1983.

9. BROOKS, R. **Flesh and Machines**. Nova York: Pantheon Books, 2002.

10. FEDERACAO INTERNACIONAL DE ROBOTICA, E NACOES UNIDAS. **World Robotics 2001, Estatísticas, Análises de Mercado, Previsões, Estudos de Casos e Rentabilidade do Investimento em Robótica**. Nova York e Genebra: Nações Unidas, 2001.

11. WIKIPEDIA. Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Servi%C3%A7o_Nacional_de_Aprendizagem_Industrial>. Acessado em 20 jun. 2018.

12. WIKIPEDIA. Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/SENAI_Santa_Catarina>. Acessado em 20 jun. 2018.

13. SENAI. Instituto Senai de Inovação em Sistemas Embarcados. Disponível em: <http://www.fiescnet.com.br>. Acessado em 20 jun. 2018.

14. ESTADÃO. **Indústria instala 1,5 mil robôs por ano**. Disponível em: https://economia.estadao.com.br/noticias/geral,industria-instala-1-5-mil-robos-por-ano,70001935172. Acessado em 21 jun. 2018.

15. 320VOLT. **Encoder, Angle Measurement**. Disponível em: <http://320volt.com/en/encoder-kullanimi-aci-olcumu-ve-ccs-c-pic16f628-ornek-uygulama/>. Acessado em 15 jun. 2018.

16. DYNAPAR ENCODERS. **Dynapar: o que é encoder**. Disponível em: <https://www.dynaparencoders.com.br/blog/index.php/encoders/como-funciona-encoder/>. Acessado em: 15 jun 2018.

17. CANON. **New - Rotary Encoder**. Disponível em <http://www.canon.com/bctv/faq/rotary.html>. Acessado em 15 jun. 2018.

18. EITEL, E. **Basics of rotary encoders: Overview and new technologies**, Machine Design Magazine, 2014. Disponível em: http://www.machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0. Acessado em 15 jun. 2018.

19. MCMILLAN,G. K., CONSIDINE, D.M. **Process Instruments and Controls Handbook**, 5 ed, 1999, McGraw Hill.

20. MITCHELL, L. .**TI-5000EX Serial/Incremental Encoder Test System User Manual**, Mitchell Electronics, Inc.

21. IFPR. **Comunicação Serial e Paralela**. Disponível: <http://wiki.foz.ifpr.edu.br/wiki/index.php/Comunicação_serial_e_paralela>. Acessado em: 15 jun 2018.

22. STEMMER, M. R. et al. DAS5305 – **Informática Industrial I Programação C Avançada / Comunicação PC x Periféricos**. Florianópolis: S2i – Sistemas Industriais Inteligentes, [200-]. 46 slides, color.

23. WIKIPEDIA. **Universal Serial Bus**. Disponível em: <https://pt.wikipedia.org/wiki/Universal_Serial_Bus>. Acessado em 13 jun. 2018.

24. UFMG. **Universal Serial Bus**. Disponível em: <http://homepages.dcc.ufmg.br/~adrianoc/usb/>. Acessado em 13 jun. 2018.

25. FLESCH, R. C. C. **Configurações de Sistemas de Aquisição de Sinais**. Florianópolis: das, [2011-]. 72 slides, color.

26. MAXIMINTEGRATED. **How Far and How Fast Can You Go with RS-485?**, Disponível em: https://www.maximintegrated.com/en/app-notes/index.mvp/id/3884. Acessado em 14 mai. 2018.

27. SOLTERO, M; ZHANG, J; COCKRIL, C; ZHANG, K; KINNARIRD, C; KUGELSTADT, T. **RS-422 and RS-485 Standards Overview and System Configurations, Application Report (pdf)**. 2002Texas Instruments. Disponível em: <http://www.ti.com/lit/an/slla070d/slla070d.pdf>. Acessado em: 16 jun. 2018.

28. ELETRONIC INDUSTRIES ASSOCIATION. **Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems**. 1983

29. TEXAS INSTRUMENTS. **DS3695,DS3695A,DS3695AT,DS3695T,DS96172, DS96174, DS96F172MQML, DS96F174MQML: Application Note 847 FAILSAFE Biasing of Differential Buses**. Texas Instruments. 2011. Disponível em: <http://www.ti.com/lit/an/snla031/snla031.pdf >. Acessado em: 7 abr. 2018.

30. GATES CORPORATION. **How belt drives impact overhung load**. Disponível em: https://www.gates.com/resources/resource-library/white-papers/how-belt-drives-impact-overhung-load. Acessado em 28 abr. 2018.

31. MORTENSEN, S. H.; BECKWITH, S. **General Picture of a Synchronous Machine**. A.E. Standard Handbook for Electrical Engineers, 8 ed. McGraw-Hill. p. 646-647.

32. HAMEYER, K. **Electrical Machine I: Basics, Design, Function, Operation**. RWTH Aachen University Institute of Electrical Machines.

33. LEE, N. C. (2006). Practical Guide to Blow Moulding. iSmithers Rapra Publishing.

34. KIM, S. (2017). Electric Motor Control: DC, AC, and BLDC Motors.

35. EUROPUMP, **Variable Speed Pumping, A Guide to Successful Applications, Executive Summary**. 2004. p. 9.

36. BOSE, B. K. **Power Electronics and Motor Drives: Advances and Trends**. Academic Press. 2006, pag 328, 397, 481.

37. HAMEYER, K.**Electrical Machine I: Basics, Design, Function, Operation.** 2011, RWTH Aachen University Institute of Electrical Machines.

38. STOLTING, Hans-Dieter. **Motor Systematics**. Handbook of Fractional-Horsepower Drives. 2008. Springer. Pag. 5, Tabela 1.1.

39. IONEL, D.M. (2010). "**High-Efficiency Variable-Speed Electric Motor Drive Technologies for Energy Savings in the US Residential Sector**". 12 ed Conferência Internacional "Optimization of Electrical and Electronic Equipment". IEEE. Pag. 1403–1414.

40. ALGER, Philip L. **AC Commutator Motors**. A.E. Standard Handbook for Electrical Engineers, 8 ed. 1949. McGraw-Hill. Pag. 755–763.

41. KRISHNAN, R. **Permanent Magnet Synchronous and Brushless DC Motor Drives**. Pag. xvii. (2008)

42. NOZAWA, T (2009). "Tokai University Unveils 100W DC Motor with 96% Efficiency". Tech-On -- Nikkei Electronics.

43. SUH S. , KANG S. K.; CHUNG D.; STROUD I. **Theory and Design of CNC Systems**. Springer Science & Business Media. 2008.

44. JACEK F. G. **Permanent Magnet Motor Technology: Design and Applications**, 3 ed. CRC Press.2011.

45. RALF D.; MARTIUS G.. **The Playful Machine: Theoretical Foundation and Practical Realization of Self-Organizing Robots**. Springer Science & Business Media.

46. ZUNT,D. **Who did actually invent the word "robot" and what does it mean?.** The Karel Čapek website.

47. ZILLIE, S. do R. **A robótica educacional no ensino fundamental: perspectivas e prática**. Repositório Institucional da Universidade Federal de Santa Catarina. Acessado em 12 de mar. 2018.

48. BENITTI,F. B. V.; VAHLDICK, A.; URBAN, D. L., KRUEGER, M. L. e HALMA. **Experimentação com Robótica Educativa no Ensino Médio: ambiente, atividades e resultados**. Worshop de Informática na Escola. Acessado em 15 mai. 2018.

49. GONÇALVES, P. C. **Protótipo de um robô móvel de baixo custo para uso educacional**. Biblioteca Central da Universidade Estadual de Maringá. Acessado em 15 mai. 2018.

50. CRAIG, J. J.; **Robótica**, tradução Heloísa Coimbra de Souza, revisão técnica Reinaldo Augusto da Costa Bianchi. – 3. ed. – São Paulo: Pearson Education do Brasil, 2012. Título original: Introduction to robotics: mechanics and control ISBN 978-85-8143-128-4.

51. WIKIPEDIA. **Parâmetros Denavit-Hartenberg**. Disponível em: <https://pt.wikipedia.org/wiki/Par%C3%A2metros_de_Denavit-Hartenberg>. Acessado em 06 jun. 2018.

52. SPONG, M. W.; VIDYASAGAR, M. **Robot Dynamics and Control**. New York: John Wiley & Sons. ISBN 9780471503521.1989.

53. LEGNANI, G; CASOLO, F; RIGHETTINI, P; ZAPPA, B. .**A homogeneous matrix approach to 3D kinematics and dynamics — I. Theory**. Mechanism and Machine Theory. doi:10.1016/0094-114X(95)00100-D.

54. LEGNANI, G; CASOLO, F; RIGHETTINI, P; ZAPPA, B. **A homogeneous matrix approach to 3D kinematics and dynamics—II. Applications to chains of rigid bodies and serial manipulators**. Mechanism and Machine Theory. doi:10.1016/0094-114X(95)00101-4.

55. RICHARD, P.. **Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators**. MIT Press, Cambridge, Massachusetts. ISBN 978-0-262-16082-7.

56. MCCARTHY,J. M. , 1990, **Introduction to Theoretical Kinematics**, MIT Press, Cambridge, MA.

57. UICKER, J. J., PENNOCK, G. R., e SHIGLEY, J. E., 2003, **Theory of Machines and Mechanisms**, Oxford University Press, New York.

58. MCCARTHY,J. M. e SOH, G. S. 2010, **Geometric Design of Linkages**, Springer, New York.

59. SICILIANO, Bruno, KHATIB, Oussama; (2016), Handbook of Robotics, editora Springer.

60. INTECHOPEN. **Accurate Numerical Methods for Computing 2D and 3D Robot Workspace**,(2011). Disponível em <http://cdn.intechopen.com/pdfs/24606/InTech-

Accurate_numerical_methods_for_computing_2d_and_3d_robot_workspace.pdf>. Acessado em 18 jun. 2018.

61. SOMMERVILLE, I. Engenharia de software. São Paulo: Pearson Prentice Hall, 2011.

62. PHYS ORG. **The Adept Quattro robot beats iPhone game 1to50 in 6.67 seconds**. Disponível em: <https://phys.org/news/2011-03-adept-quattro-robot-iphone-game.html.> Acessado em 15/06/18.

63. PALMER, D.; GUZMAN, S. C.; AXINTE, D.; **Real-time method for tip following navigation of continuum snake arm robots**. Em Robotics and Autonomous Systems, Volume 62, Issue 10, 2014, pag 1478-1485.

64. BEAL, V. **SDK - software development kit**. Webopedia. Acessado em 7 abr. 2018.Disponível em: < https://www.webopedia.com/TERM/S/SDK.html >. Acessado em 13 jun. 2018.

65. MICROSOFT. **SDK Definition**. Tech Terms. 15 de Abril de 2010. Acessado em 7 de Mai. de 2018.

66. MICROSOFT. **What is a dll**. Disponível em: <https://support.microsoft.com/pt-br/help/815065/what-is-a-dll>. Acessado em 01 jul. 2018.

67. ROBOTICS.**Introduction wiring**. Disponível em: <http://emanual.robotis.com/docs/en/platform/manipulator_h/introduction/#wiring>. Acessado em 15 abr. 2018.

68. ROBOTICS. Disponível em: <http://www.robotis.us/dynamixel-pro-h54-200-s500-r/>. Acessado em 15 abr. 2018.

69. WIKIPEDIA. **Unity Technologies Logo**. Disponível em: <https://commons.wikimedia.org/wiki/File:Unity_Technologies_logo.svg>. Acessado em 20 jun. 2018.

70. NETCODERS. **Visual Studio 15 abertura de pastas**. Disponível em: <http://netcoders.com.br/visual-studio-15-abertura-pastas/>

71. SLIDEPLAYER, **Robot Workspace**. Disponível em: <http://slideplayer.com/slide/4239214/14/images/24/Robot+workspace+Robot+workspace+plays+an+important+role+when+selecting+an+industrial+robot+for+an+anticipated+task..jpg>. Acessado 15/06/18.

72. JASONDAVIES. **Animated Bézier Curves**. Disponível em: <https://www.jasondavies.com/animated-bezier/>. Acessado em: 07 jun. 2018.

73. FACOM. **Filas Encadeadas**. Disponível em: <http://www.facom.ufu.br/~abdala/DAS5102/TEO_HeapFilaDePrioridade.pdf>. Acessado em 07 jul. 2018

APÊNDICE A – MOVIMENTAÇÃO DA CÂMERA NO AMBIENTE

Algoritmo de controle da câmera dentro do ambiente de testes.

```
using UnityEngine;
using System.Collections;
public class CameraHandler : MonoBehaviour {
	private static readonly float PanSpeed = 20f;
	private static readonly float ZoomSpeedTouch = 0.1f;
	private static readonly float ZoomSpeedMouse = 15;//0.5f;
	private static readonly float[] BoundsX = new float[]{-100f, 40};
	private static readonly float[] BoundsZ = new float[]{-100f, -50f};
	private static readonly float[] ZoomBounds = new float[]{180, 100f};
	public float sensX = 100.0f;
	float rotationX = 0.0f;
	float rotationY = 0.0f;
	private Camera cam;
	private Vector3 lastPanPosition;
	private int panFingerId; // Touch mode only
	private bool wasZoomingLastFrame; // Touch mode only
	private Vector2[] lastZoomPositions; // Touch mode only

	void Awake() {
		cam = GetComponent<Camera>();
	}

	  void Update() {
		if(Input.touchSupported && Application.platform != RuntimePlatform.WebGLPlayer) {
			HandleTouch();
		} else {
			HandleMouse();
		}
	  }

	void HandleTouch() {
		switch(Input.touchCount) {
		case 1: // Panning
			wasZoomingLastFrame = false;
			// If the touch began, capture its position and its finger ID.
			// Otherwise, if the finger ID of the touch doesn't match, skip it.
```

```
                Touch touch = Input.GetTouch(0);
                if (touch.phase == TouchPhase.Began) {
                        lastPanPosition = touch.position;
                        panFingerId = touch.fingerId;
                } else if (touch.fingerId == panFingerId && touch.phase == TouchPhase.Moved) {
                        PanCamera(touch.position);
                }
                break;

        case 2: // Zooming
                Vector2[]newPositions= newVector2 []{Input.GetTouch(0).position,
Input.GetTouch(1).position};
                if (!wasZoomingLastFrame) {
                        lastZoomPositions = newPositions;
                        wasZoomingLastFrame = true;
                } else {
                        // Zoom based on the distance between the new positions compared to the
                        // distance between the previous positions.
                        float newDistance = Vector2.Distance(newPositions[0], newPositions[1]);
                        float oldDistance = Vector2.Distance(lastZoomPositions[0],
lastZoomPositions[1]);
                        float offset = newDistance - oldDistance;
                        ZoomCamera(offset, ZoomSpeedTouch);
                        lastZoomPositions = newPositions;
                }
                break;
                default:
                        wasZoomingLastFrame = false;
                        break;
                }
        }

        void HandleMouse() {
                // On mouse down, capture it's position.
                // Otherwise, if the mouse is still down, pan the camera.
                if (Input.GetMouseButtonDown(2)) {
                        lastPanPosition = Input.mousePosition;
                } else if (Input.GetMouseButton(2)) {
                        PanCamera(Input.mousePosition);
                }
```

```
            if (Input.GetMouseButton (1)) {
                rotationX += Input.GetAxis ("Mouse X") * sensX * Time.deltaTime;
                rotationY -= Input.GetAxis ("Mouse Y") * sensX * Time.deltaTime;
                transform.localEulerAngles = new Vector3 (rotationY, rotationX, 0);
            }
            // Check for scrolling to zoom the camera
            //float scroll = Input.GetAxis("Mouse ScrollWheel");
            //ZoomCamera(scroll, ZoomSpeedMouse);
            float scroll = Input.GetAxis("Mouse ScrollWheel");
            transform.Translate(0, 0, scroll * ZoomSpeedMouse, Space.World);
        }

        void PanCamera(Vector3 newPanPosition) {
            // Determine how much to move the camera
            Vector3 offset = cam.ScreenToViewportPoint(lastPanPosition - newPanPosition);
            Vector3 move = new Vector3(offset.x * PanSpeed, 0, offset.y * PanSpeed);
            // Perform the movement
            transform.Translate(move, Space.World);
            // Ensure the camera remains within bounds.
            Vector3 pos = transform.position;
            pos.x = Mathf.Clamp(transform.position.x, BoundsX[0], BoundsX[1]);
            pos.z = Mathf.Clamp(transform.position.z, BoundsZ[0], BoundsZ[1]);
            transform.position = pos;

            // Cache the position
            lastPanPosition = newPanPosition;
        }
        void ZoomCamera(float offset, float speed) {
            if (offset == 0) {
                return;
            }

            cam.fieldOfView = Mathf.Clamp(cam.fieldOfView - (offset * speed), ZoomBounds[0],
ZoomBounds[1]);
        }
}
```