UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MATHEUS LIMA SAMPAIO

MODELAGEM, IDENTIFICAÇÃO E CONTROLE PID DE UM ROBÔ PLANAR

FORTALEZA

2021

MATHEUS LIMA SAMPAIO

MODELAGEM, IDENTIFICAÇÃO E CONTROLE PID DE UM ROBÔ PLANAR

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientadora: Prof. Dr[a]. Laurinda Lúcia Nogueira dos Reis

Coorientador: Prof. Me. Darielson Araújo de Souza

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S184m Sampaio, Matheus Lima.
Modelagem, identificação e controle PID de um rôbo planar / Matheus Lima Sampaio. – 2021.
61 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2021.
Orientação: Profa. Dra. Laurinda Lúcia Nogueira dos Reis.
Coorientação: Prof. Me. Darielson Araújo de Souza.

1. Robótica. 2. Manipulador robótico. 3. Controle PID. I. Título.

CDD 621.3

MATHEUS LIMA SAMPAIO

MODELAGEM, IDENTIFICAÇÃO E CONTROLE PID DE UM ROBÔ PLANAR

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Drª. Laurinda Lúcia Nogueira dos Reis (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Me. Darielson Araújo de Souza (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Me. Josias Guimarães Batista
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

À Prof. Dr. Laurinda Lúcia Nogueira dos Reis por me orientar no projeto final de curso.

Aos Doutorandos em Engenharia Elétrica, Josias Guimarães Batista e Darielson Araújo de Souza pelo grande auxílio no projeto.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

“Seja Feliz”

(Todos)

RESUMO

Este trabalho introduz os conceitos de controladores Proporcional Integral Derivativo (PID), técnica de controle de processos que une as ações derivativa, integral e proporcional, conceitos de identificação de sistemas pelo método de mínimos quadrados e conceitos de manipuladores robóticos. Além disso, são apresentados a conceituação de cinemática direta, inversa e o Método de Newton para sistemas não-lineares, com o objetivo de resolver a cinemática inversa de um manipulador robótico com 2 Graus de Liberade (GDL). Todos os conceitos são aplicados com o objetivo de controlar o manipulador robótico e ajustar sua trajetória em um plano cartesiano. Por fim, buscando comprovar a eficácia do controlador e o gerador de trajetórias, foi realizada a trajetória horizontal e trapezoidal no plano cartesiano, obtendo resultados satisfatórios bem próximos das referências.

Palavras-chave: Robótica. Manipulador Robótico. Controle PID.

ABSTRACT

This work introduces the concepts of controllers gls PID, a process control technique that unites as derivative, integral and proportional actions, concepts of system identification by the combined data method and concepts of robotic manipulators. In addition, they are responsible for a conception of direct and inverse kinematics and the Newton Method for non-linear systems, with the objective of solving an inverse kinematics of a robotic manipulator with 2 GDL. All concepts are designed to control the robotic manipulator and adjust its trajectory on a Cartesian plane. Finally, in order to prove the effectiveness of the controller and trajectory generator, the horizontal and trapezoidal trajectory was performed in the Cartesian plane, obtaining satisfactory results very close to the references.

Keywords: Robotics. Robotic manipulator. PID control.

LISTA DE FIGURAS

Figura 1 – Tipos de Garras. 16
Figura 2 – Sensor potenciômetro. 18
Figura 3 – Estrutura do manipulador articulado 18
Figura 4 – Estrutura do manipulador Esférico 19
Figura 5 – Estrutura do manipulador SCARA 20
Figura 6 – Estrutura do manipulador cilíndrico 20
Figura 7 – Estrutura do manipulador cartesiano 21
Figura 8 – Matriz de transformação de um manipulador de 3 elos. 22
Figura 9 – Robô planar. 23
Figura 10 – Robô planar. 24
Figura 11 – Robô planar. 25
Figura 12 – Diagrama de blocos PID. 26
Figura 13 – MPU6050 30
Figura 14 – PWM MG996R 31
Figura 15 – Zona morta MG996R 31
Figura 16 – Zona morta MG996R 32
Figura 17 – Zona morta MG996R 32
Figura 18 – Manipulador utilizado 33
Figura 19 – Resposta ao degrau elo final 34
Figura 20 – Resposta ao degrau elo final com controlador 36
Figura 21 – Esquema Geral. 37
Figura 22 – Resposta ao degrau do elo final com controlador. 38
Figura 23 – Resposta ao degrau elo da base com controlador. 39
Figura 24 – Junta da base. 40
Figura 25 – Junta Final. 40
Figura 26 – Trajetória Vertical. 41
Figura 27 – Trajetória Trapezoidal. 42

LISTA DE TABELAS

Tabela 1 – Especificações Arduino Uno R3 29
Tabela 2 – Índices de desempenho 39

LISTA DE ABREVIATURAS E SIGLAS

GDL Graus de Liberade

PID Proporcional Integral Derivativo

LISTA DE SÍMBOLOS

Ae	Área efetiva da antena
B	Largura de faixa em que o ruído é medido em Hertz
d	Distância em metros
E	Campo elétrico
FA	Fator da antena
Gr	Ganho de recepção
h	Altura efetiva ou comprimento efetivo de uma antena
I	Corrente elétrica
k	Constante de Boltzmann's
K	Eficiência de irradiação
M	Variação do patamar de ruído em função da RBW
N	Condutor de neutro
NF	Figura de ruído
Ni	Potência do ruído na entrada
No	Potência do ruído na saída
P	Potência
R	Resistência
Si	Potência do sinal na entrada
So	Potência do sinal na saída
t	Tempo
V	Tensão
ZL	Impedância da antena
Zo	Impedância de referência (50Ω)
λ	Comprimento de onda
Γ	Coeficiente de reflexão

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Justificativa 14
1.2 Objetivos 15
1.2.1 Objetivos específicos 15
1.3 Organização do texto 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Componentes da tipologia de manipuladores 16
2.1.1 Ponta ou garra (End-Effector) 16
2.1.2 Atuadores 17
2.1.3 Sensores 17
2.2 Manipulador robótico 17
2.2.1 Tipos de manipuladores 18
2.2.1.1 Articulado *18*
2.2.1.2 Esférico *19*
2.2.1.3 SCARA *19*
2.2.1.4 Cilíndrico *20*
2.2.1.5 Cartesiano *20*
2.3 Cinematica direta 21
2.3.1 Representação Denavit Hartenberg 22
2.4 Cinemática inversa 24
2.4.1 Exemplo de um manipulador RR planar 24
2.5 Controlador PID 26
2.5.1 Controle proporcional 27
2.5.2 Controle proporciona mais integral 27
2.5.3 Controlador PID aplicado em um sistema de segunda ordem 27
3 METODOLOGIA 29
3.1 Microcontrolador: Arduino UNO R3 29
3.2 MPU 6050 30
3.3 GNU Octave 30
3.4 Servo MG996R 30

3.5 **Manipulador** 33
3.6 **Identificação de cada junta** 34
3.7 **Controle PID** 35
3.8 **Cinemática inversa aplicada para gerar a trajetória** 36
3.9 **Sistema do manipulador** 37
4 **RESULTADOS** 38
4.1 **Resultados ao degrau** 38
4.2 **Resultados dos controladores atuando simultaneamente** 39
4.3 **Resultado trajetória vertical** 41
4.4 **Resultado trajetória Trapezoidal** 42
5 **CONCLUSÕES E TRABALHOS FUTUROS** 43
REFERÊNCIAS 44
APÊNDICES 46
APÊNDICE A – Códigos-fontes utilizados para controle do braço robótico 46
ANEXOS 61

1 INTRODUÇÃO

A definição de robô dada pelo "Robotics Institute of America" (RIA): "Um robô é um manipulador multifuncional reprogramável projetado para mover o material, peças, ferramentas ou dispositivos especializados com movimentos programados de variável bruta para o desempenho de uma variedade de tarefas.". Portanto, os robôs podem efetuar várias atividades realizadas por humanos e também atividades que não são executáveis por pessoas, sendo possível programar e montá-los pra várias aplicações(bélicas, industriais, entretenimento, entre outras) (ROBÓTICA, 1997).

A industrialização está intimamente ligada aos robôs e manipuladores, com isso, o desenvolvimento de um alavanca o outro. As indústrias precisam ser competitivas para oferecer melhores preços e produtos de alta qualidades, assim, precisam de equipamentos com eficiência e precisão. Para alcançar os objetivos citados, foram desenvolvidos várias técnicas de controle de robôs, controlando as posições trajetórias e desenvolvendo robôs específicos para cada necessidade, tanto para atender exigências industriais como necessidades comerciais (CRAIG, 2009).

Com o desenvolvimentos das técnicas e difusão dos conhecimentos, foram criados vários robôs, que ajudam e facilitam atividades do cotidiano. Com tal difusão, o contato com os robôs tornou frequente, sendo cada vez mais indispensáveis para as atividades gerais do dia a dia. E certamente existem várias lacunas que ainda podem ser aproveitadas no mercado para o desenvolvimento de outros tipos de robôs.

Este trabalho apresenta a aplicação das técnicas de identificação, controle e aplicação nos robôs. Sendo o enfoque nos manipuladores que são compostos por juntas e elos. Serão aplicadas as técnicas em um robô articulado ou também chamados de angulares, sendo o formado semelhante a um braço humano.

1.1 Justificativa

A grande presença de robôs nas atividades gerais da sociedade atual, mostra a grande importância que eles têm e o grande potencial de crescimento que ainda é possível obter com os robôs.

A grande janela de possibilidades de aplicação, mostra a necessidade de conhecer, desenvolver e programá-los. Com esse conhecimento é possível desenvolver tanto produtos

como novas possibilidades para a indústria, ajudando no desenvolvimento da indústria 4.0 que procuram menor custo, operações em tempo real e a otimização dos processos (INDUSTRIA, 2017).

1.2 Objetivos

Este trabalho tem como objetivo integrar um microcontrolador com software matemático para o controle de posição e trajetória de um manipulador com 2 graus de liberdade. Utilizando técnicas clássicas de cinemática e controle. Visando desenvolver as técnicas aplicadas, facilitando o conhecimento sobre a área e utilizando equipamentos de fácil acesso.

As especificações requeridas foram baseadas no artigo "*Modeling, Simulation and Control of a Robotic Arm*"(EBRAHIMI, 2019). A ultrapassagem máxima aceita será de 5% e o tempo de subida de 2 segundos.

1.2.1 Objetivos específicos

Os objetivos principais do trabalho são:

- Modelagem matemática de um manipulador;
- Implementar controle PID em um manipulador real;
- Controlar posição e trajetória do manipulador real;

1.3 Organização do texto

No capítulo 1, foi dada uma abordagem geral sobre o tema, tendo a função de deixar claro a importância da discussão do assunto tratado neste trabalho e a justificativa para tal abordagem.

No capítulo 2, é mostrada a fundamentação teórica utilizado no trabalhando, definindo termos utilizados e desenvolvendo modelos utilizados.

No capítulo 3, é desenvolvida a metodologia, esse tópico mostra passo a passo como foi realizado o projeto, mostrando os equipamentos, sensores e atuadores utilizados e como foram utilizados.

No capítulo 4, é mostrado e discutido os resultados e os índices de desempenho do controlador. As trajetórias utilizadas foram a horizontal e a trapezoidal. Também foi mostrada a resposta ao degrau de cada junta.

2 FUNDAMENTAÇÃO TEÓRICA

Essa capítulo tem como objetivo apresentar os fundamentos básicas utilizados no projeto, mostrando a fundamentação teórica sobre cinemática direta e inversa, controle PID, os tipos de manipuladores, sensores e representação de Denavit-Hartenberg e avançar nos conhecimentos sobre robôs. Logo, o robô pode ser definido como:

> Mecanicamente, um robô é composto por uma série de elementos ou elos unidos por juntas que permitem um movimento relativo entre cada um dos dois elos. A constituição física da maioria dos robôs industriais guarda certa semelhança com a anatomia do braço humano, razão pela qual às vezes, se referem aos diferentes elementos que compõem o robô, termos como corpo, braço, cotovelo e pulso (ROBÓTICA, 1997).

2.1 Componentes da tipologia de manipuladores

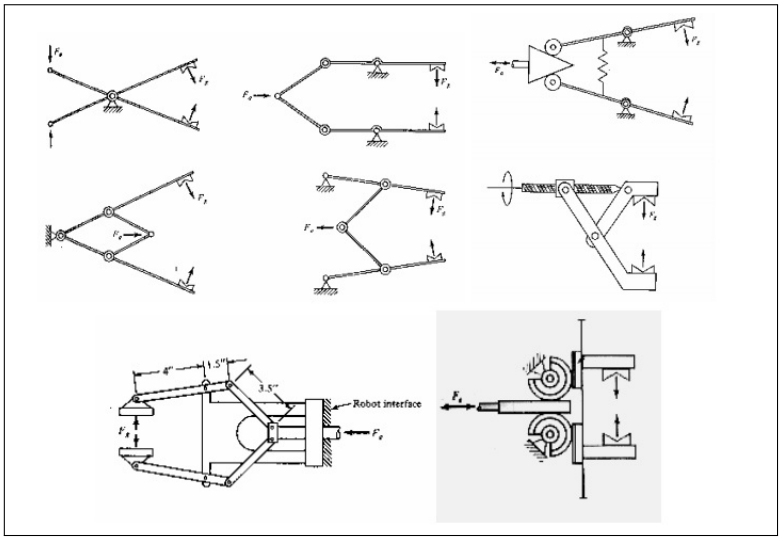
2.1.1 Ponta ou garra (End-Effector)

O primeiro componente a ser definido do manipulador será a ponta ou garra, a definição formal pode ser dada por:

> Componente ligado à extremidade do braço, isto é, ligado ao último elo do manipulador, e que tem funções adicionais (agarrar ou prender um objeto, ou ainda um dispositivo com funções adicionais mais específicas). A ponta ou *end-effector* pode ser do tipo garra (*gripper*) ou uma ferramenta (*tool*). (SANTOS, 2004)

A Figura 1 um mostra alguns tipos de garras que podem ser utilizadas em manipuladores.

Figura 1 – Tipos de Garras.



Fonte: (SANTOS, 2004).

2.1.2 Atuadores

Componentes que transformam energia hidráulica, elétrica e pneumática em energia mecânica. Esses componentes usam a energia mecânica para que o manipulador se mova (LEE, 2018).

Os atuadores podem ser classificados em:

- Hidráulicos: geralmente é acionado por fluido em movimento, geralmente óleo pressurizado. São utilizados para transporte de carga pesada e tem de média para alta precisão de controle de posição e velocidade.
- Pneumáticos: geralmente é acionado por fluido em movimento, geralmente ar comprimido. São utlizados no transporte de cargas médias e pequenas, tendo uma alta velocidade, mas uma baixa precisão, porém, um baixo custo.
- Eletromagnético: são atuadores usados costumeiramente em robôs, como motores de corrente contínua e motores de passo. São utilizados para o transporte de cargas médias e pequenas, possuem uma alta precisão, porém ocupam muito espaço.

2.1.3 Sensores

Os sensores têm grande importância na robótica, eles fazem com que o robôs recebam informações do ambiente, a definição pode ser dada por:

> Fornecem informação ao controlador, nomeadamente em que local estão as diversas juntas do manipulador. Além destes sensores internos há também os interruptores de fim de curso que delimitam as deslocações extremas das juntas. Existem também os sensores externos dedicados a recolher informação adicional sobre o ambiente (SANTOS, 2004).

Os sensores podem ser potenciômetros, codificadores, acelerômetros, etc. Cada um deles podendo ser utilizados de diversas formas ou combinados para uma maior precisão, a Figura 2 mostra um modelo de potenciômetro.

2.2 Manipulador robótico

Existem várias composições de manipuladores e tipos de manipuladores, a maior parte dos manipuladores são classificados como:

> A maioria dos manipuladores industriais atualmente tem seis ou menos graus de liberdade. Esses manipuladores são geralmente classificados cinematicamente com base nas três primeiras juntas do braço, com o pulso sendo descrito

Figura 2 – Sensor potenciômetro.

Fonte: (WB SENSORS, 2004).

separadamente. A maioria desses manipuladores caem em um dos cinco tipos geométricos: articulado (RRR), esférico (RRP), SCARA (RRP), cilíndrico (RPP) ou cartesiano (PPP) (SPONG; VIDYASAGAR, 2008a).

2.2.1 Tipos de manipuladores

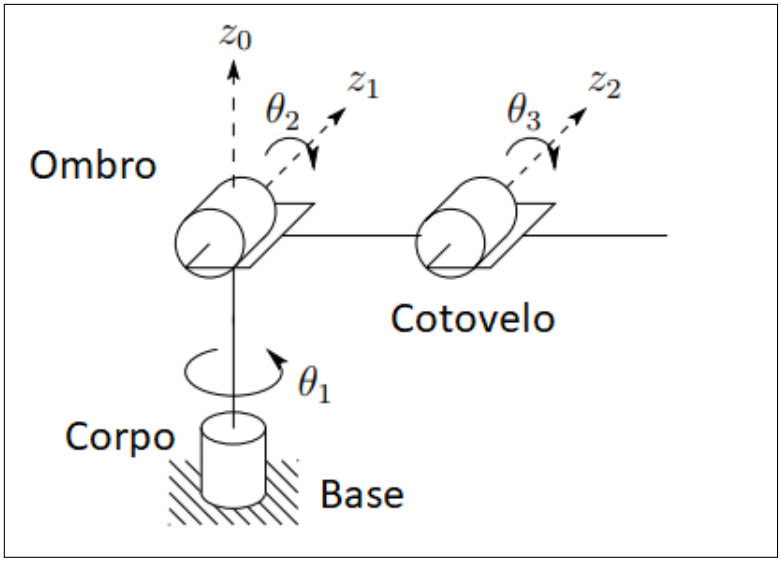
2.2.1.1 Articulado

O primeiro tipo de manipular é o manipulador antropormófico que é definido como :

> O manipulador articulado é chamado de manipulador de revolução ou manipulador antropomórfico, nessa configuração existem pelo menos três juntas de rotação. O eixo de movimento da junta de rotação da base é ortogonal às outras duas juntas de rotação, que são simétricas entre si. Tal configuração é a que permite maior mobilidade aos robôs. O volume de trabalho apresenta um geometria mais complexa em relação às outras (TRONCO, 2017).

Na Figura 3 é mostrada a estrutura do manipulador robótico de revolução:

Figura 3 – Estrutura do manipulador articulado

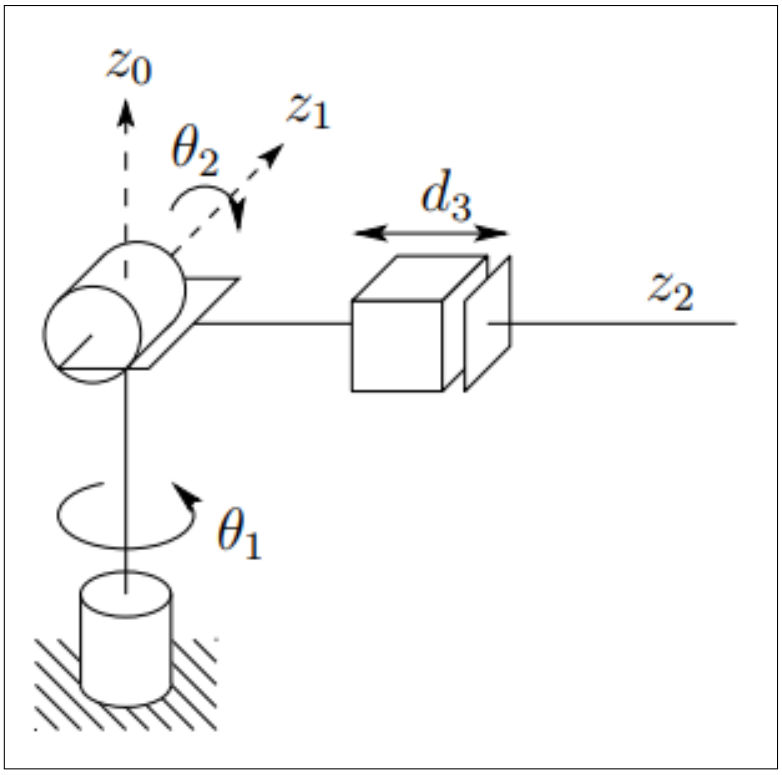


Fonte: (SPONG; VIDYASAGAR, 2008b).

2.2.1.2 Esférico

O manipulador esférico é criado ao substituir a terceira articulação do manipulador antropomórfico por uma articulação prismática (RRP), como mostrado na Figura 4. O manipulador tem as três juntas(z0, z1, z2), mutuamente, perpendiculares.

Figura 4 – Estrutura do manipulador Esférico



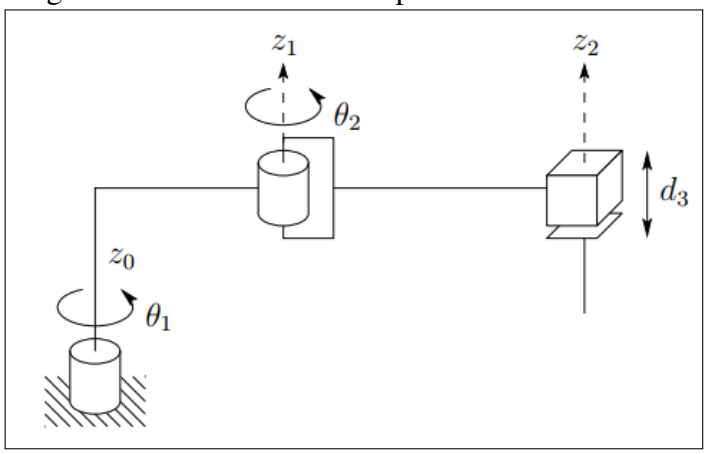
Fonte: (SPONG; VIDYASAGAR, 2008a).

O manipulador esférico consegue ter um grande volume de atuação, porém tem um modelo cinemático complexo.

2.2.1.3 SCARA

O SCARA (Robô Articulado Compatível Seletivo para Montagem) é uma configuração popular, que, como o próprio nome sugere, é adaptada para montagem operações. Embora o SCARA tenha uma estrutura RRP, é bastante diferente do configuração esférica na aparência e na sua gama de aplicações. Ao contrário do desenho esférico, que tem z0, z1, z2 mutuamente perpendiculares , o SCARA tem z0, z1, z2 paralelos. A Figura 5 apresenta a estrutura do manipulador SCARA.

Figura 5 – Estrutura do manipulador SCARA



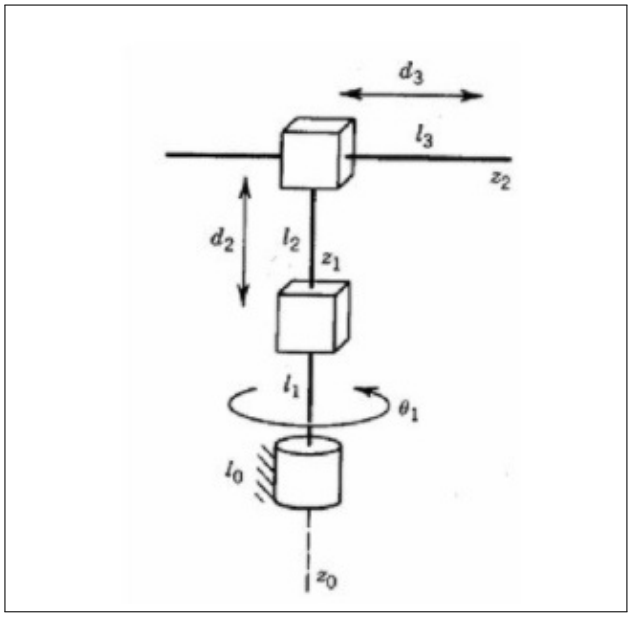
Fonte: (SPONG; VIDYASAGAR, 2008a).

2.2.1.4 Cilíndrico

A primeira junta é de rotação e produz uma rotação em torno da base, enquanto a segunda e a terceira juntas são prismáticas, logos, o manipulador cilíndrico é RPP. Conforme o nome, sugere que as variáveis das juntas são as coordenadas cilíndricas do *End-Effector* em relação a base.

A Figura 6 mostra com pode ser representado o manipulador cilíndrico.

Figura 6 – Estrutura do manipulador cilíndrico



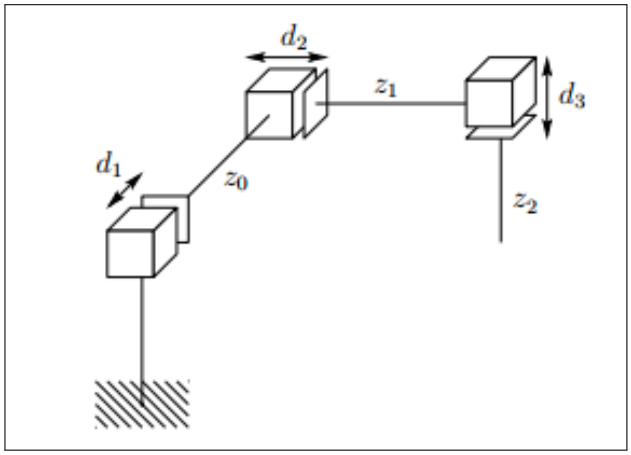
Fonte: (LEE, 2018).

2.2.1.5 Cartesiano

O manipulador cartesiano é definido como:

> Um manipulador cujas três primeiras articulações são prismáticas é conhecido como um manipulador cartesiano (PPP). Para o manipulador cartesiano, as variáveis das juntas são as coordenadas cartesianas do "*End-Effector*"em relação à base. Como seria de se esperar, a descrição cinemática de este manipulador é a mais simples de todas as configurações.(SPONG; VIDYASAGAR, 2008a)

Figura 7 – Estrutura do manipulador cartesiano



Fonte: (SPONG; VIDYASAGAR, 2008a).

As configurações cartesianas são úteis para aplicações de montagem de mesa e, como robôs de pórtico, para transferência de material ou carga. Na Figura 7 é mostrada a estrutura do manipulador cartesiano.

2.3 Cinematica direta

Uma das definições sobre cinemática é dada por:

> A cinemática de um manipulador é o estudo do conjunto de relações entre as posições, velocidades e acelerações dos seu elos (SANTOS, 2004).

A definição para Craig é dada por :

> Cinemática é a ciência do movimento que trata do assunto sem considerar as forças que o causam. Nessa ciência estudamos a posição, a velocidade, a aceleração e todas as derivadas de ordem mais elevada das variáveis de posição (com respeito ao tempo ou quaisquer outras variáveis). Assim, o estudo da cinemática dos manipuladores refere-se a todas as propriedades do movimento que sejam geométricas e baseadas no tempo (CRAIG, 2009).

O principal objetivo da cinemática direta é analisar a posição de cada efetuador a partir da posição angular de cada uma das articulações.

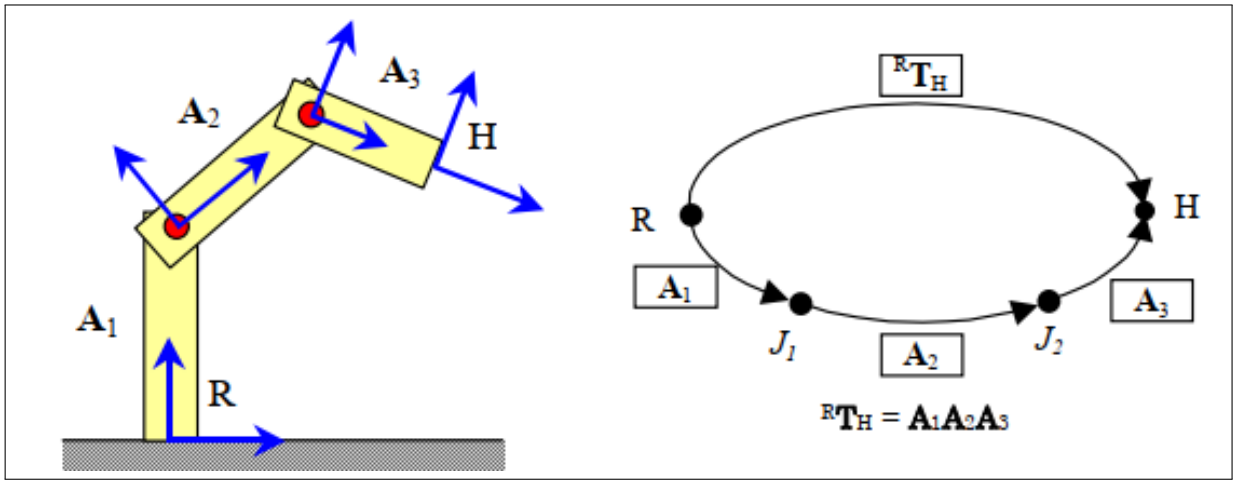
Para conseguir o objetivo da cinemática direta existe a transformação ${}^{R}T_{H}$, porém, essa relação só dá informação entre a origem e o "*End-Effector*", logo, é necessário uma relação

para as juntas intermediárias, dada por $A_1, A_2...A_n$. Sendo:

$$^{R}T_H = A_1A_2...A_n \tag{2.1}$$

A Figura 3 esquematiza as transformações associadas a um manipulador de 3 elos.

Figura 8 – Matriz de transformação de um manipulador de 3 elos.



Fonte: (SANTOS, 2004).

O algoritimo da cinemática direta consiste em 7 passos, sendo:

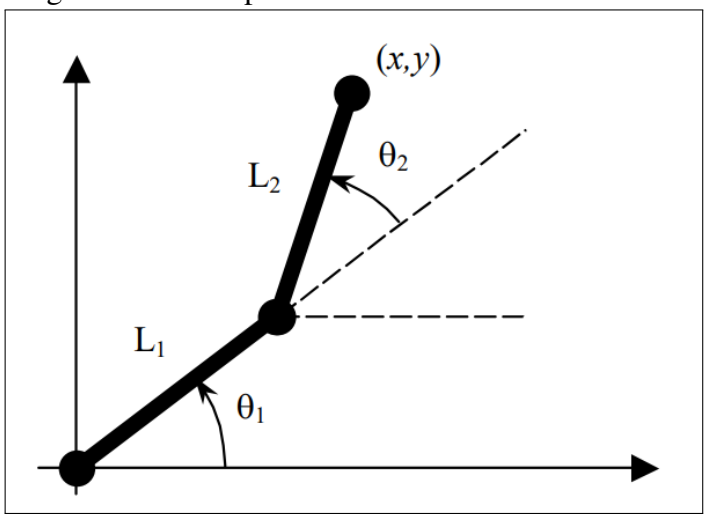
1. Colocar o robô na posição inicial;
2. Atribuir um sistema de coordenada para cada elo;
3. Descrever as relações(translações e rotações) entre as variáveis das juntas e dos elos;
4. Determinar as matrizes de tranformação A_i dos diversos elos;
5. Multiplicar os A_i dos diversos elos;
6. Obter as coordenadas de posição;
7. Obter as coordenadas de orientação.

2.3.1 Representação Denavit Hartenberg

Embora seja possível realizar análises com estrutura arbitrária anexado a cada link, é útil ser sistemático na escolha desses quadros. Assim, a convenção usada para selecionar quadros de referência em aplicações robóticas é Denavit-Hartenberg, ou convenção D-H. Nesta convenção, cada transformação homogênea A_i é representado como um produto de quatro transformações básicas (SPONG; VIDYASAGAR, 2008a).

Considere agora um manipulador robótico planar Figura 9.

Figura 9 – Robô planar.



Fonte: (SANTOS, 2004)

Considerando a Figura 9, a descrição do subespaço (**??**) T_W^B, ou T_0^2, para o manipulador em questão pode ser dada por:

$$T_W^B = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & x \\ \sin\phi & \cos\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Em que x e y informam a posição e a orientação do efetuador final. Logo, o subespaço é gerado à medida em que são atribuídos valores a essas variáveis. De acordo com os parâmetros dos elos do robô, as equações cinemáticas são dadas da seguinte forma:

$$T_0^2 = \begin{bmatrix} \cos\theta_1+\theta_2 & -\sin\theta_1+\theta_2 & 0 & L_1\cos\theta_1+L_2\sin\theta_1+\theta_2 \\ \sin\theta_1+\theta_2 & \cos(\theta_1+\theta_2) & 0 & L_1\sin\theta_1+L_2\cos(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

A partir das equações dados foram encontradas as seguintes equações:

$$x = L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2) \tag{2.4}$$

$$y = L_1\sin\theta_1 + L_2\sin(\theta_1+\theta_2) \tag{2.5}$$

2.4 Cinemática inversa

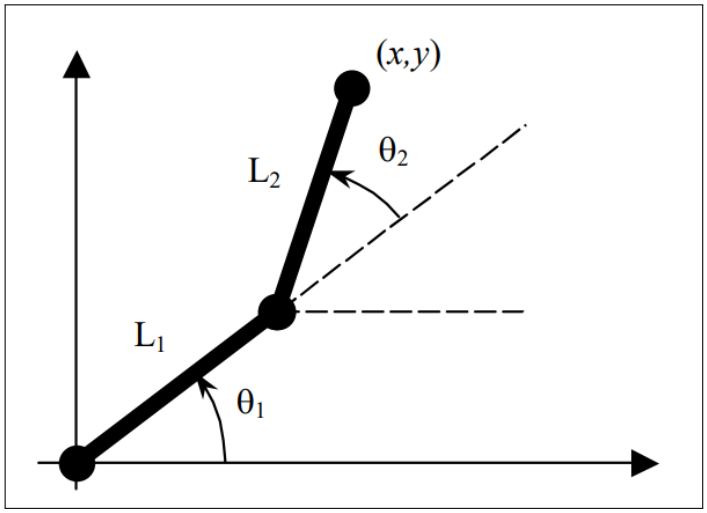
A cinemática direta tem o ângulo de cada junta e encontra a posição do manipulador, na cinemática inversa é o inverso, a posição é dada e quer encontrar o conjunto de ângulos para essa posição. Para Santos a cinemática inversa é dada por:

> A cinemática inversa procura determinar o conjunto de valores das juntas que se adéquam a uma dada configuração do espaço operacional ou cartesiano. A cinemática inversa pode ser vista conceitualmente como o conjunto de processos para determinar as funções inversas do sistema das expressões da cinemática direta (SANTOS, 2004).

2.4.1 Exemplo de um manipulador RR planar

Nesse tópico será demonstrado a resolução da cinemática inversa para um manipulador 2-DOF planar. Na Figura 10 será mostrado o modelo que será resolvido.

Figura 10 – Robô planar.



Fonte: (SANTOS, 2004)

Para começar a solução é utilizada a solução da cinemática direta, dada por :

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \tag{2.6}$$

$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \tag{2.7}$$

Sendo $L_1 = 0{,}12$ m e $L_2 = 0{,}20$ m. Elevando as expressões ao quadrado e somando-as:

$$\begin{aligned} x^2 + y^2 = {} & L_1^2 \cos^2\theta_1 + L_1^2 \sin^2\theta_1 + L_2 \sin^2(\theta_1 + \theta_2) + L_2^2 \cos^2(\theta_1 + \theta_2) \\ & + 2L_1 L_2 \cos\theta_1 \cos(\theta_1 + \theta_2) + 2L_1 L_2 \sin\theta_1 \sin(\theta_1 + \theta_2) \end{aligned} \tag{2.8}$$

Simplificando ficará:

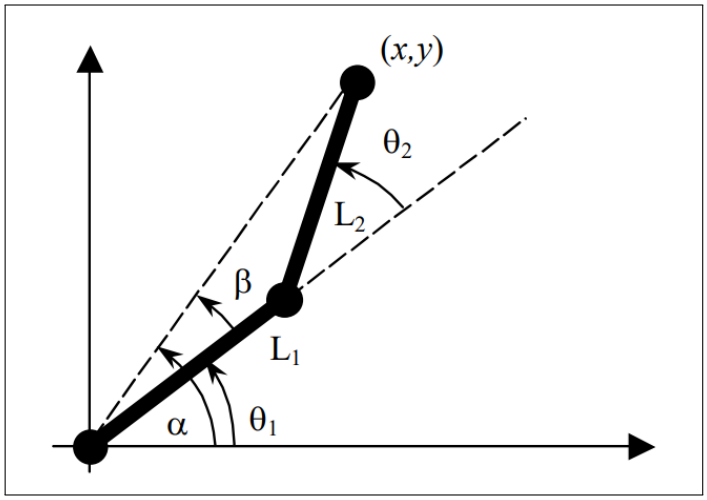
$$x^2 + y^2 = L_1^2 + L_2^2 + 2L_1 L_2 \cos\theta_2 \tag{2.9}$$

Assim θ será:

$$\theta_2 = \pm \arccos \frac{x^2 + y^2 - L_1^2 - L_1^2}{2L_1 L_2} \tag{2.10}$$

Portanto, é possível notar que existem duas soluções possíveis pelo $\pm$ na fórmula. Para encontra θ_1 é preciso utilizar o recurso da tangente trigonométrica.

Figura 11 – Robô planar.



Fonte: (SANTOS, 2004)

Sendo:

$$\tan(A - B) = \frac{\tan A - \tan B}{1 - \tan A \tan B} \tag{2.11}$$

$$\tan\beta = \frac{L_2 \sin\theta_2}{L_1 + L_2 \cos\theta_2} \tag{2.12}$$

$$\tan\alpha = \frac{y}{x} \tag{2.13}$$

Como na Figura 11, $\theta_1 = \alpha - \beta$, logo, pela relação da tangente da diferença da Fórmula 2.12:

$$\theta_1 = \arctan\left(\frac{y(L_1 + L_2\cos\theta_2 - xL_2\sin\theta_2)}{x(L_1 + L_2\cos\theta_2) + yL_2\sin\theta_2}\right) \tag{2.14}$$

Portanto, foi possível encontrar uma relação para ambos os ângulos.

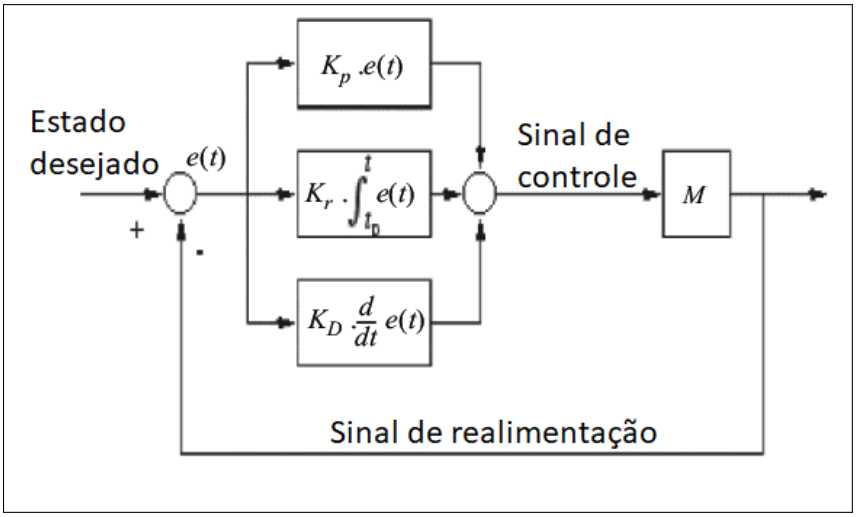
2.5 Controlador PID

O controlador PID é formado por três termos, proporcional, integral e derivativo, cada um interferindo de forma diferente no sistema. O controle PID tem a função de transferência dada por:

$$D(s) = k_p + \frac{k_I}{s} + K_D s \tag{2.15}$$

Sendo K_p o controle proporcional,K_I o controle integral e K_D o controle derivativo. O diagrama de blocos do controlador PID pode ser representado pela Figura 12.

Figura 12 – Diagrama de blocos PID.



Fonte: (FRANKLIN *et al.*, 2013).

2.5.1 *Controle proporcional*

A definição dada para a parte proporcional é dada por:

> Quando o sinal de controle realimentado é linearmente proporcional ao erro do sistema, chamamos o resultado de realimentação proporcional (FRANKLIN *et al.*, 2013)

.

A função de transferência de controlador proporcional e dada por:

$$\frac{U(s)}{E(S)} = D_{cl}(s) = k_p \tag{2.16}$$

No caso de aplicar um controlado proporcional em uma planta de segunda ordem:

> O projetista pode controlar o termo constante nesta equação, o qual determina a frequência natural, mas não pode controlar o coeficiente de amortecimento. O sistema é Tipo 0 e, se fizermos k_p grande o suficiente para obtermos o erro em regime permanente pequeno, o coeficiente de amortecimento pode ser muito pequeno para uma resposta transitória satisfatória usando apenas o controlador proporcional (FRANKLIN *et al.*, 2013)

.

2.5.2 *Controle proporciona mais integral*

O controle PI(proporcional-integral) é o controle proporcional adicionado do integral de equação dada por:

$$\frac{U(s)}{E(S)} = D_{cl}(s) = k_p + \frac{k_I}{s} \tag{2.17}$$

> A introdução do termo integral aumenta o tipo do sistema e, portanto, pode rejeitar totalmente perturbações constantes (FRANKLIN *et al.*, 2013).

2.5.3 *Controlador PID aplicado em um sistema de segunda ordem*

Por fim é adicionado o controlador derivativo , com isso a equação de controle ficará a Equação 2.15. O controle derivativo tem como objetivo suavizar a resposta.

Para exemplificar os efeitos do controle PID, pode-se aplicar em um sistema de segunda ordem. A equação do sistema de segunda ordem aplicado o PID é dado por :

$$s^3 + (a_1 + Ak_d)s^2 + (a_2 + Ak_p) + Ak_I = 0 \tag{2.18}$$

> O ponto aqui é que nesta equação, na qual as três raízes determinam a natureza da resposta dinâmica do sistema, os três parâmetros kp, kI e kD selecionados apropriadamente determinam, em teoria, as raízes arbitrariamente. Sem o termo derivativo, existiriam apenas dois parâmetros de ajuste, mas ainda existiriam três raízes; assim, a escolha das raízes da equação característica seria restrita. Para ilustrar o efeito deste termo de forma mais concreta, um exemplo numérico é utilizado.(FRANKLIN *et al.*, 2013)

3 METODOLOGIA

O capítulo metodologia mostrará passo a passo do projeto do manipulador robótico, começando pelos utilitários físicos utilizados, como, sensores e microcontroladores até a parte de programação e controle utilizados.

3.1 Microcontrolador: Arduino UNO R3

O microcontrolador escolhido para o projeto foi o Arduino Uno R3, ele foi escolhido pela sua extensa lista de bibliotecas e por ser um dos mais utilizados para pequenos projetos. O arduino tem as especificações necessárias para o projeto, são elas:

Especificações	
Microcontrolador	ATmega328
Tensão de operação	5V
Input(Recomendado)	7 - 12 V
Input(Limite)	6 - 20 V
Pinos Digital I/O	6
Pinos Analógicos Input	6
Corrente (DC) por pino input e output	20 mA
Corrente (DC) para pino 3.3 V	50 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Frequência de Clock	16 MHz

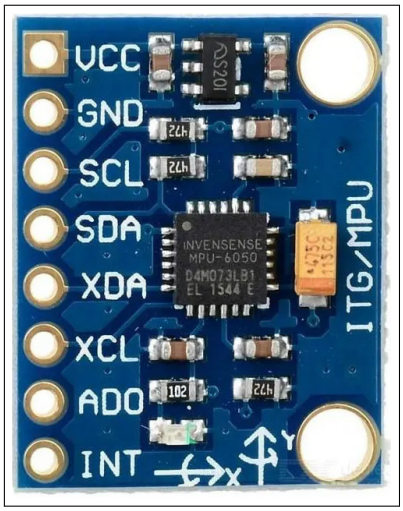
Tabela 1 – Especificações Arduino Uno R3

O Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em Wiring, e é essencialmente C/C++. Ele foi desenvolvido na Itália em 2005 com o objetivo de diminuir custo de pequenos projetos, principalmente os projetos escolares.

3.2 MPU 6050

O MPU6050 é um sensor acelerômetro e giroscópio 3 Eixos e com 6 graus de liberdade utilizado em projetos de robótica, com comunicação serial I2C, com as portas SDA e SCL. O sensor foi utilizado no manipulador robótico com o objetivo de medir a posição angular de cada junta. O sensor é mostrador na Figura 13.

Figura 13 – MPU6050



Fonte: (IVENSENSE SEMICONDUCTORS, 2013)

Para conseguir essa medição foi necessário utilizar o método de fusão sensorial e o Filtro de Kalman, com essas ferramentas matemáticas foi possível medir com alta precisão cada junta.

3.3 GNU Octave

GNU Octave é um software interativo de alta performance voltado para cálculo numérico. O software foi utilizado para gerar e enviar o sinal de entrada para o microcontrolador por uma comunicação serial USB, além de ser utilizado como coletor de dados.

A comunicação foi feita em I2C utilizando uma porta USB entre o computador utilizando o Octave e o Arduino com código inserido.

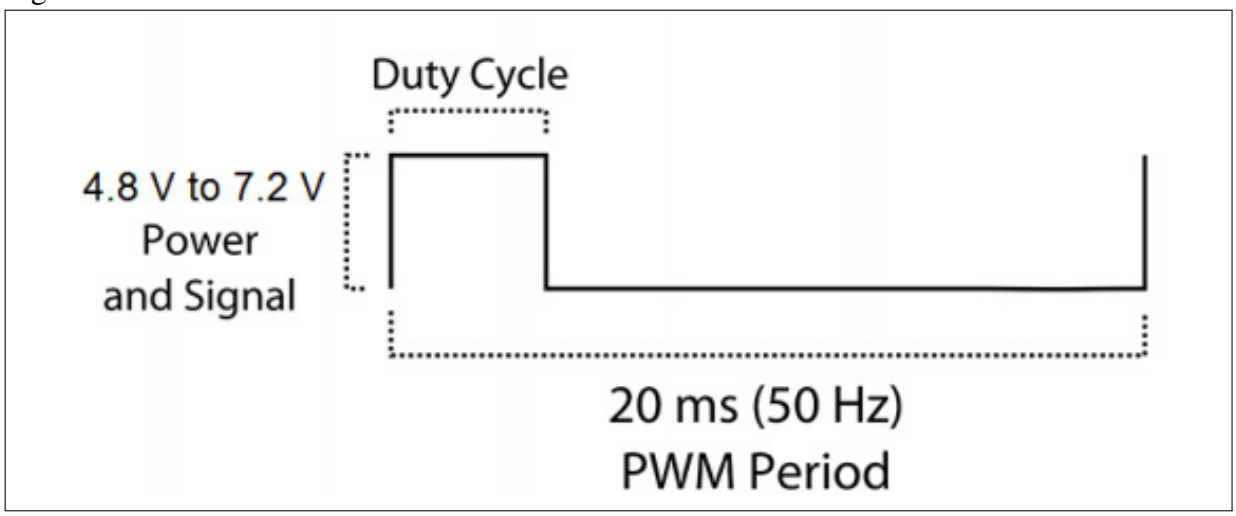
3.4 Servo MG996R

O servo motor é uma maquina de corrente contínua, que apresenta um controlador de posição em malha fechada. Esse manipulador é muito utilizado em pequenos projetos de

robótica por ter preço acessível e de fácil manipulação.

Para manipular o servo é necessário 4.8V a 7.2V para a alimentação e um sinal PWM, como mostrado na Figura.

Figura 14 – PWM MG996R



Fonte: (TOWERPRO, 2010).

O sinal de pwm varia de 1ms a 2ms em alta, sendo, 1ms 0 graus e 2ms 180 graus, os intervalos entre eles apresentam uma relação linear entra milissegundos e graus.

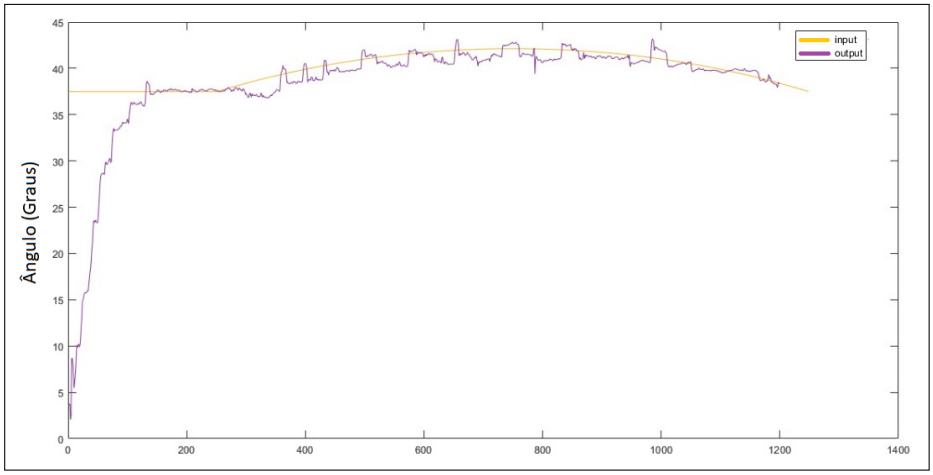
O servo utilizado no projeto foi o MG996R da *TowerPower*, mostrado na Figura 15.

Figura 15 – Zona morta MG996R

Fonte: Autor.

O servo utilizado apresenta um alto torque, no entanto, possui muitas zonas mortas, como apresentada na Figura a seguir.

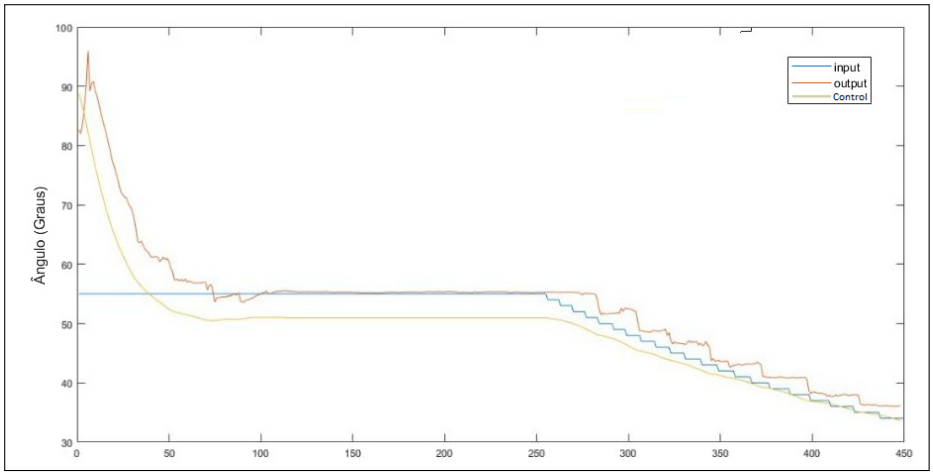
Figura 16 – Zona morta MG996R



Fonte: Autor.

Para ratificar essa possibilidade é possível ver na Figura 17 o sinal de controle, entrada e saída, é possível perceber a continuidade do sinal de controle, no entanto, a saída demora para responder.

Figura 17 – Zona morta MG996R



Fonte: Autor.

Como é possível notar na Figura 16 e 17, sendo x as amostras e y o ângulo em graus, o servo não responde a pequenas variações no sinal de comando, respondendo apenas quando o sinal de comando variar 0,5 graus. Logo, quando o servo responde, acontece uma mudança rápida de ângulo.

3.5 Manipulador

O manipulador é um RR em movimento planar, ele é constituído por 3 servos motores MG996R, 2 servos na Base e um para a segunda junta. A junta da base tem comprimento de 12 centímetro e a junta final tem comprimento de 20 centímetros. O manipulador é mostrado na Figura 18.

Figura 18 – Manipulador utilizado

Fonte: Autor

Aplicando o comprimento do manipulador nas Equações 2.6 e 2.7:

$$x = 0,12\cos\theta_1 + 0,20\cos(\theta_1 + \theta_2) \tag{3.1}$$

$$y = 0,12\sin\theta_1 + 0,20\sin(\theta_1 + \theta_2) \tag{3.2}$$

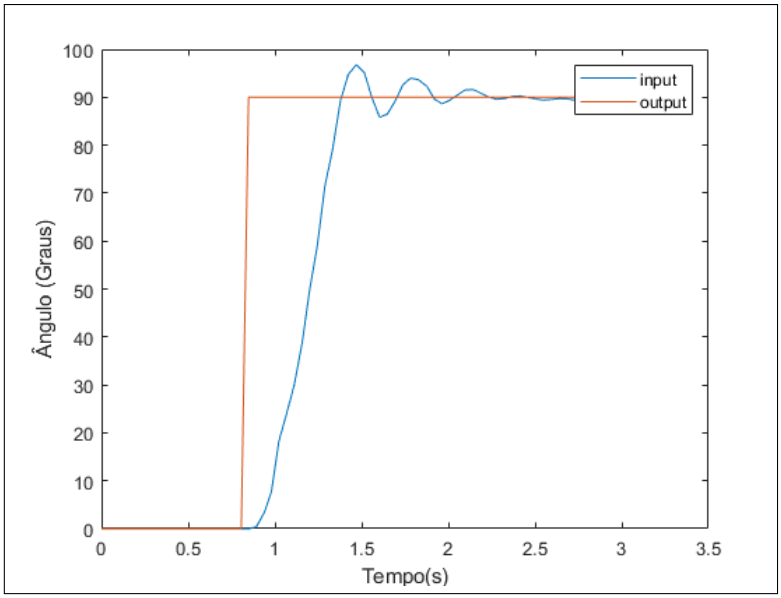
3.6 Identificação de cada junta

Para fazer a identificação da planta foi utilizado o sensor MPU6050 junto com o Arduino para enviar sinais à planta e colher os dados de saída. O processo de identificação do sistema foi feito utilizando o modelo ARX(autoregressivo com entradas exógenas)(AGUIRRE, 2004),que é caracterizado pela equação de diferenças apresentada na Equação 3.3. Foi utilizado o método dos mínimos (CHAPRA; CANALE, 2016) quadrados para fazer a estimação dos parâmetros do moelo ARX..

$$A(q)y(t) = B(q)u(t - nk) + e(t) \tag{3.3}$$

Para identificação foi enviado o sinal de degrau de Heaviside, conhecido também como degrau. O sinal foi de 0 a 90 graus, o sinal de saída para a junta final e da base foram as seguintes:

Figura 19 – Resposta ao degrau elo final



Fonte: Elaborado pelo autor (2021).

Como é possível notar na Figura 19, a forma do sinal de saída se aproxima de uma equação de se segunda ordem. Logo, foi utilizado o método dos mínimos quadrados para a identificação. As equações de cada junta foi :

$$\frac{0.03035z - 0,00075}{z^2 - 1,759z + 0,7895} \tag{3.4}$$

$$\frac{0,03943z - 0,02598}{z^2 - 1,864z + 0,8776} \tag{3.5}$$

Após a a transformação para o contínuo:

$$JuntaFinal = \frac{0,4225s + 20,83}{s^2 + 5.909s + 21.17} \tag{3.6}$$

$$JuntaBase = \frac{0,8703s + 8,981}{s^2 + 3,263s + 8,934} \tag{3.7}$$

3.7 Controle PID

Como o modelo encontrado foi um de segunda ordem e a ultrapassagem é um fator muito prejudicial para manipuladores robótico, foi necessário que o controlador PID retirasse a ultrapassagem e diminuísse a constante de tempo.

Com as Equações 3.6 e 3.7, o controlador escolhido para a junta Final foi um PID e para a Junta da Base apenas um integrador.

O método utilizado foi a alocação de polos, portanto foi necessário transformar o sistema em criticamente amortecido ou superamortecido.

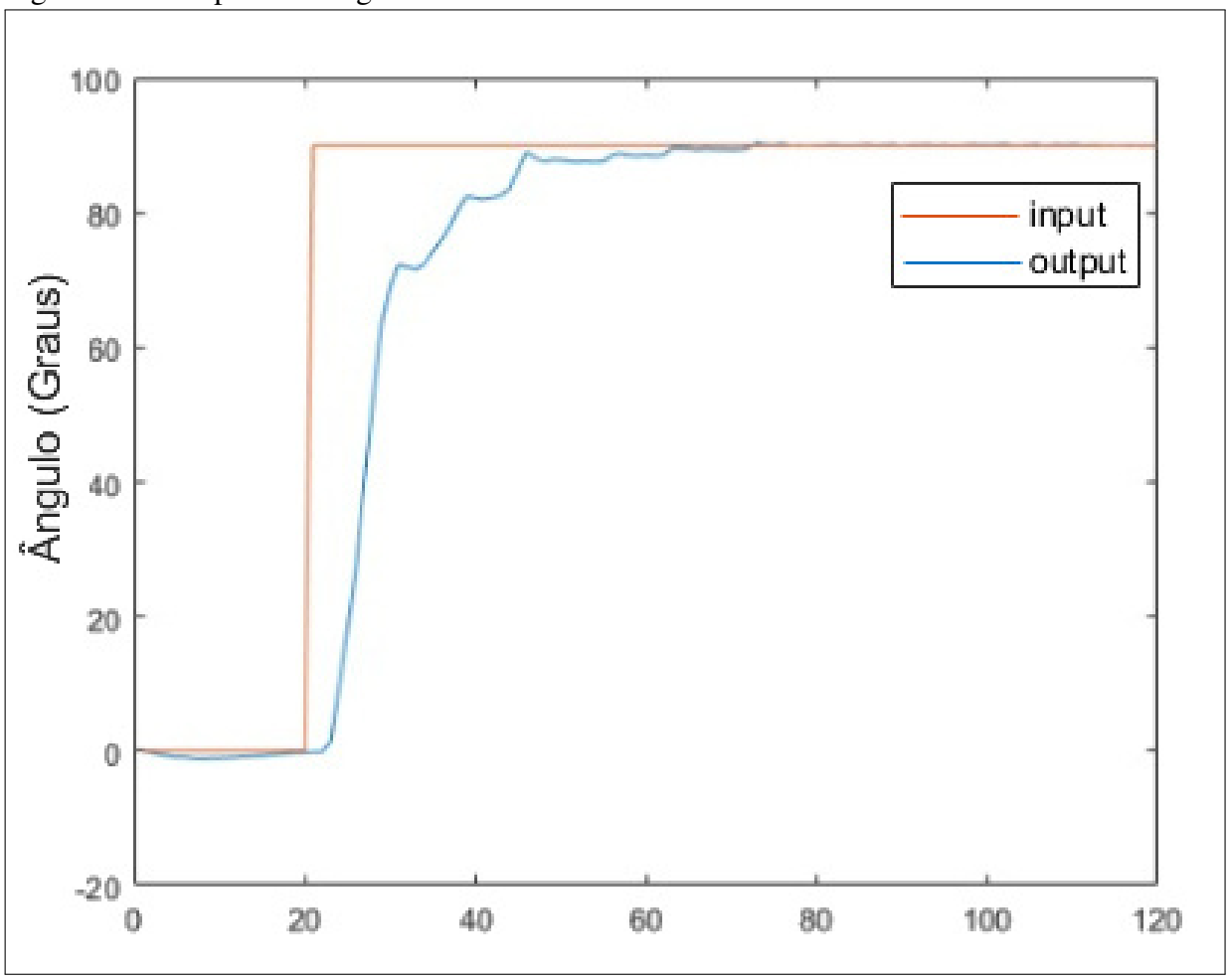
Os controlares para cada junta foram:

$$ControladorBase = \frac{0.9508}{s} \tag{3.8}$$

$$ControladorFinal = 0.379 + \frac{2.62}{s} + 0.0137s \tag{3.9}$$

Com os controladores projetados, foram aplicados no como outro degrau para validar o controlador, como mostrado na Figura 20.

Figura 20 – Resposta ao degrau elo final com controlador



Fonte: Elaborado pelo autor (2021).

3.8 Cinemática inversa aplicada para gerar a trajetória

Com o intuito de encontrar a posição final do manipulador, foi utilizado o método de newton para sistemas não lineares para encontrar os ângulos da junta final e da base para cada posição desejada. Foram utilizadas as Equações 3.1 e 3.2, depois aplicado o Método de Newton (CHAPRA; CANALE, 2016), dessas equações, logo, o sistema foi:

$$F(x) = \begin{bmatrix} 0,12\cos\theta_1 + 0,20\cos(\theta_1+\theta_2) - x \\ 0,12\sin\theta_1 + 0,20\sin(\theta_1+\theta_2) - y \end{bmatrix} \tag{3.10}$$

Sendo x e y as posições a serem encontradas, então, é necessário calcular o jacobiano da função, que é dado por:

$$J_F(x) = \begin{bmatrix} -0,12\sin\theta_1 - 0,20\sin(\theta_1+\theta_2) & -0,20\sin(\theta_1+\theta_2) \\ 0,12\cos\theta_1 + 0,20\cos(\theta_1+\theta_2) & 0,20\cos(\theta_1+\theta_2) \end{bmatrix} \tag{3.11}$$

Então, foi programada no Octave utilizando de forma interativa a seguinte equação:

$$x_{n+1} = x_n - J_F^{-1}(x_n)F(x_n) \tag{3.12}$$

Após a função pronta e aplicada para as Equações 3.10, 3.11 e 3.12, foi gerada a matriz de trajetória com as seguintes rotas: vertical e trapezoidal.

3.9 Sistema do manipulador

O esquema geral do manipulador conquiste na junção dos itens mencionados nos tópicos acima. Então, o esquema começa pela geração da trajetória pelo Octave e calculando o erro angular entre a trajetória e cada junta.

Na parte do gerador de trajetória, são gerados os ângulos das juntas da base e da junta final para a garra do manipulador se encontrar na posição x e y desejada.

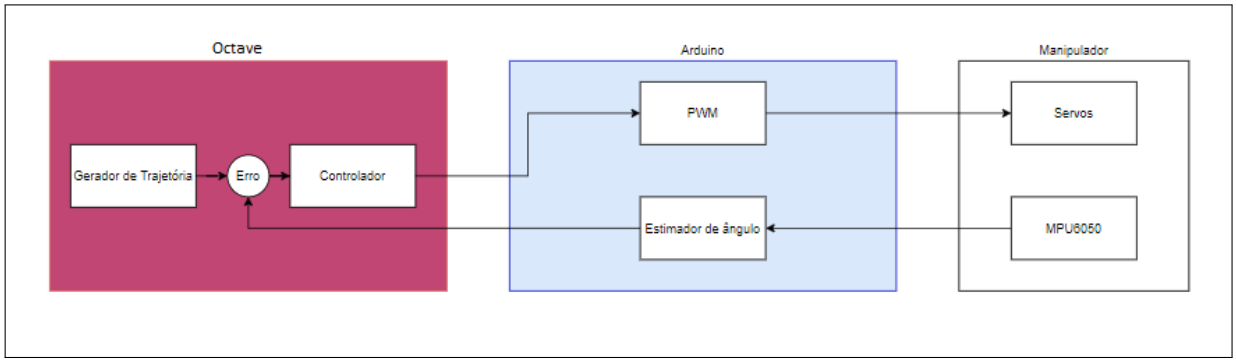
O erro é calculado, sendo a diferença entra o ângulo gerado e o ângulo medido, esse erro passa pela função de controle PID, pela parte proporcional, derivativa e integrativa. O sinal gerado é o sinal de controle que é enviado por USB para o Arduino.

O Arduino recebe por porta serial o sinal de controle de cada junta, logo, ele transforma o sinal de controle em um sinal PWM para enviar para os servos motores.

O MPU6050, que está alocado em cada junta, envia os dados do acelerômetro e giroscópio para o arduino para estimar o ângulo que cada servo se encontra. Após isso, o microcontrolador estima os ângulos por fusão sensorial e filtro de Kalman.

Por fim, é enviado os ângulos estimados para o Octave por porta serial e o processo se repete continuamente. O esquema geral pode ser visualizado na Figura 21.

Figura 21 – Esquema Geral.



Fonte: Elaborado pelo autor (2021).

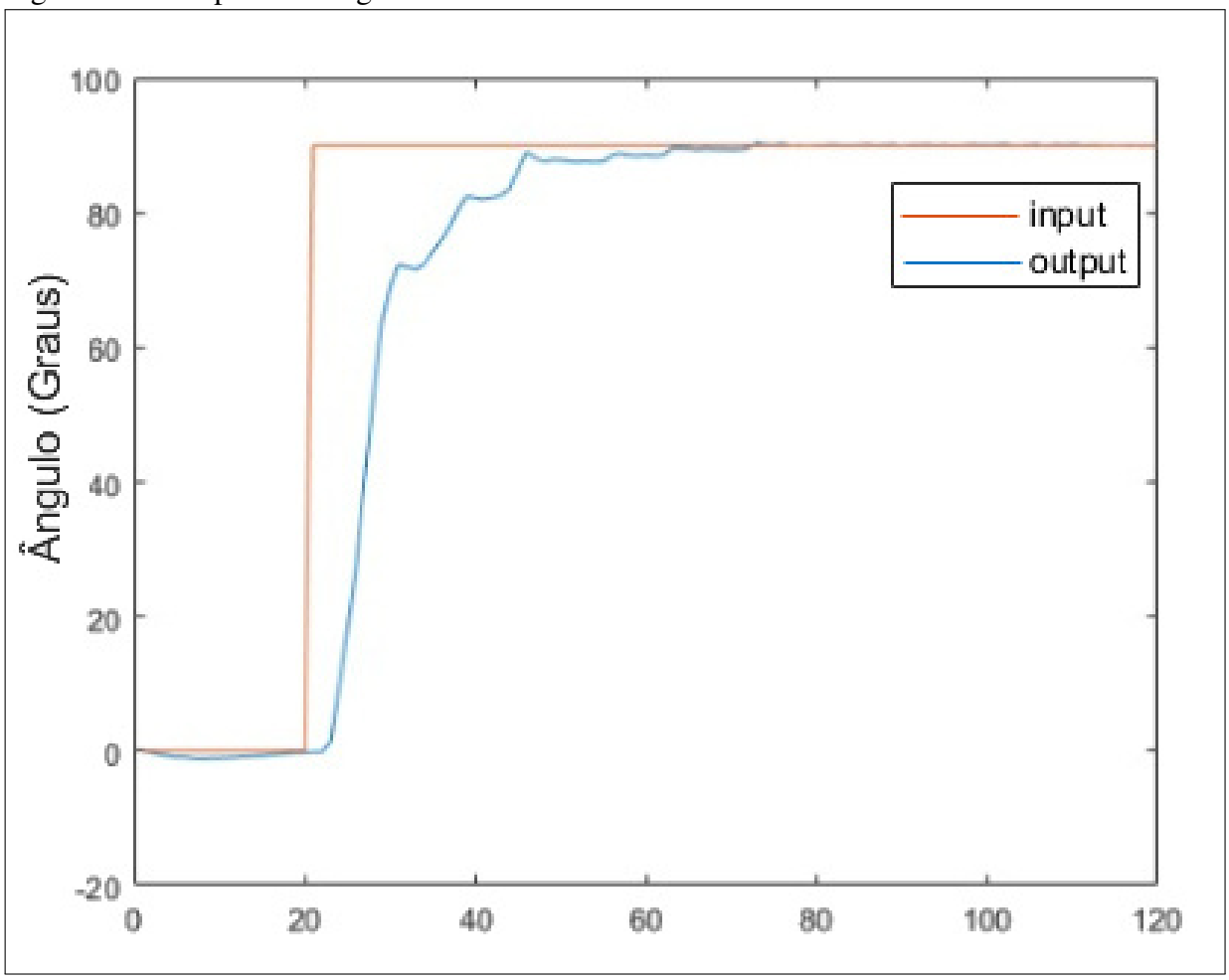
4 RESULTADOS

Nesse capítulo é mostrado os resultados práticos da implementação do controle de ambas as juntas, além disso, é mostrado as trajetórias para mostrar a eficiência do controle . As trajetórias mostradas são as de trajetória vertical e trapezoidal. Também será discutido os resultados, explicando possíveis diferenças do esperado.

4.1 Resultados ao degrau

Neste capítulo é mostrado o resultado ao degrau, além disso é analisado os índices de desempenho. Foi realizado um degrau de 90 graus e 45 graus nas juntas final e base, respectivamente. Sendo, x as amostras e o ângulo em graus.

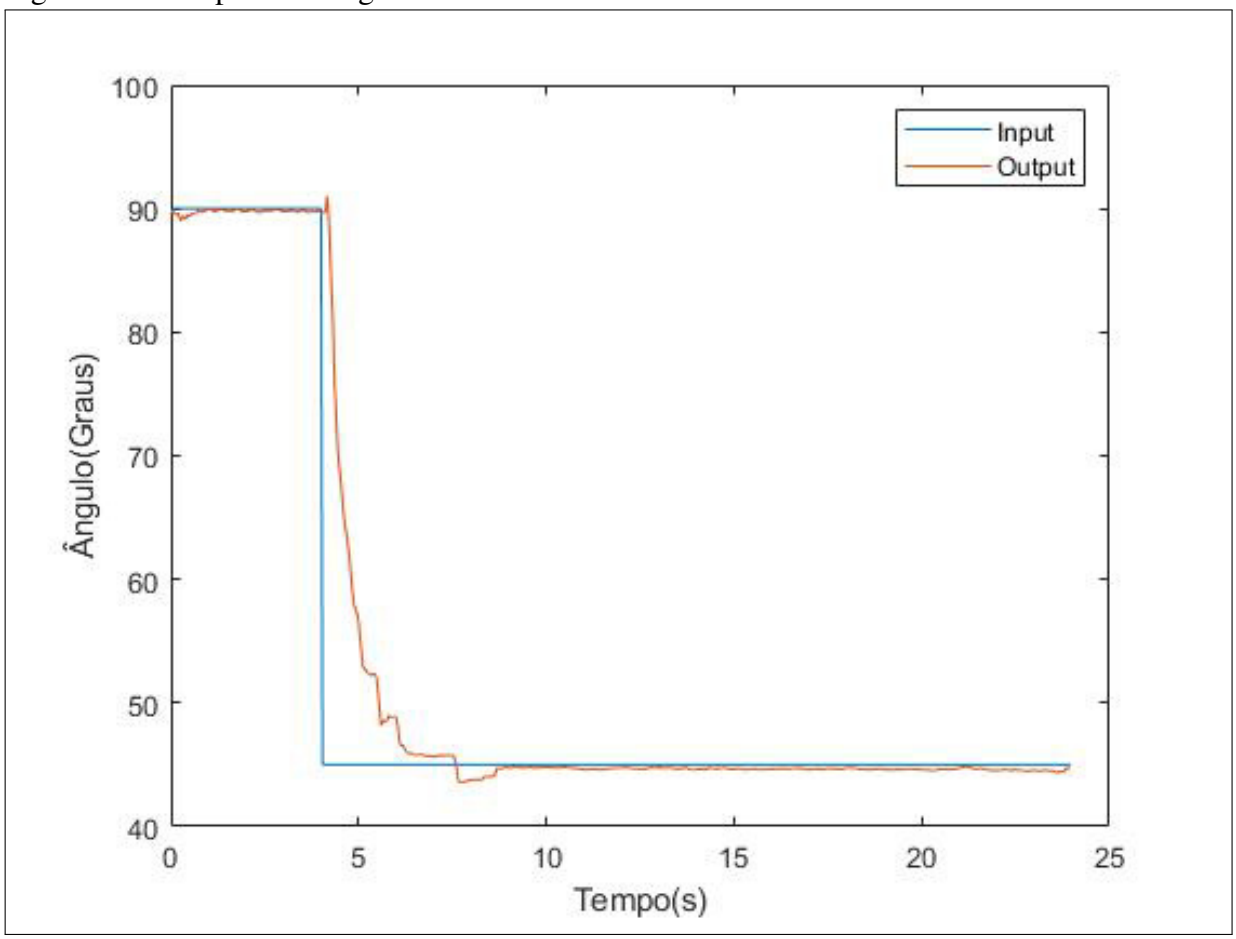
Figura 22 – Resposta ao degrau do elo final com controlador.



Fonte: Elaborado pelo autor (2021).

o *Input* é o sinal de referência, e o *Output* é o resultado experimental. Com isso, é possível notal que o experimental seguir a referência com baixo erro e com velocidade

Figura 23 – Resposta ao degrau elo da base com controlador.



Fonte: Elaborado pelo autor (2021).

Tabela 2 – Índices de desempenho

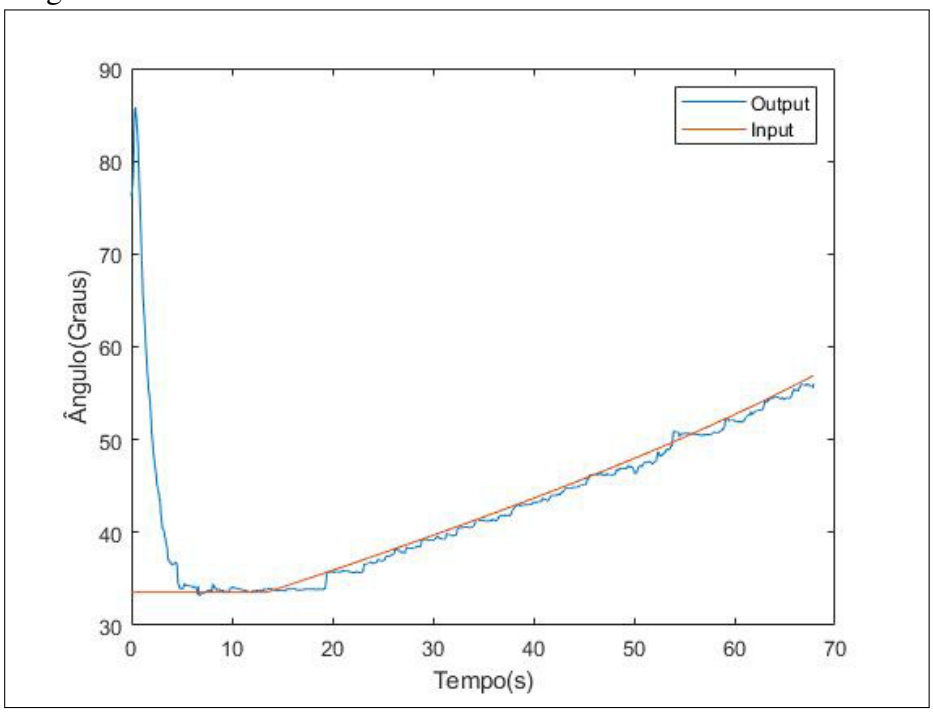
Tempo de subida (s)	Tempo de assentamento (s)	Ultrapassagem(%)
0.361	1.84	0
0.86	3.44	3.3

razoável. O elo final não obteve ultrapassagem, porém o elo da base obteve uma pequena ultrapassagem de 3.44% e um tempo de subida máximo de 0.361s e 0.86s, que segundo o artigo "*Modeling, Simulation and Control of a Robotic Arm*"(EBRAHIMI, 2019) o tempo de subida e a ultrapassagem máximos aceitáveis são de 2 segundos e 5%, respectivamente. O experimento obteve um bom assentamento, conseguiu com alta precisão se mantendo no ângulo certo.

4.2 Resultados dos controladores atuando simultaneamente

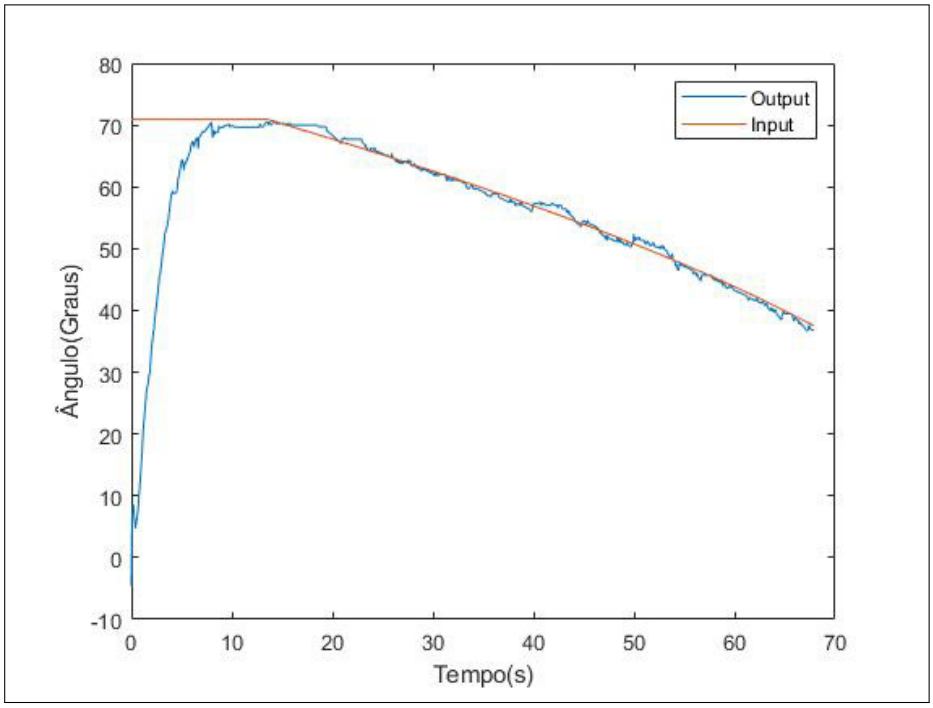
Nesse tópico é mostrado o resultado de ambas as juntas funcionando conjuntamente. Como a junta final fica após a junta da base, movimentos da junta da base interferem na junta final.

Figura 24 – Junta da base.



Fonte: Elaborado pelo autor (2021).

Figura 25 – Junta Final.



Fonte: Elaborado pelo autor (2021).

Os sinais de saída seguiram as referências, no entanto, existiu uma demora para responder aos sinais. A figura mostra que o sinal para de responder para pequenas variações no sinal de saída, essa demora é causa pela zona morta do servo morto, como foi mostrada no capítulo anterior. Apesar desse diferença, a junta seguiu a referência.

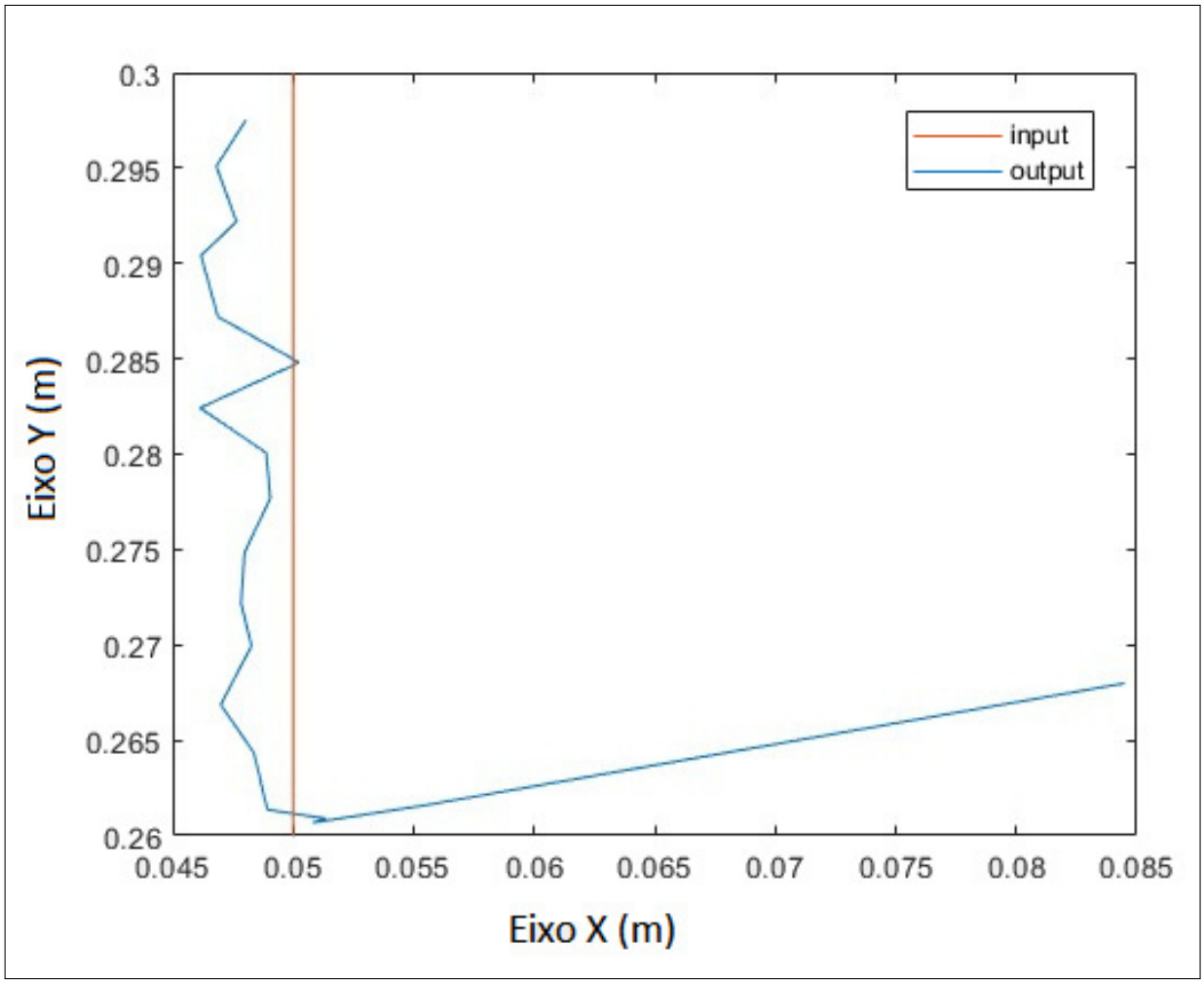
Com ambas as juntas simultaneamente, existiram certas ondulações em torno da referência, No entanto, seguiu a referência.

4.3 Resultado trajetória vertical

Nesse tópico é mostrado a trajetória vertical no plano XY de coordenadas. O *Input* é a trajetória de controle, e o *Output* é o *setpint*, que é a trajetória experimental obtida. A trajetória é uma vertical com X constrante e Y variando.

O resultado foi bem próximo ao esperado, sendo a distância máxima de erro de 5 milímetros da referência. A oscilação e causada, em parte, em razão da zona morta, pois apenas um os motores ficam se movendo em momentos diferentes, ocasionando uma divergência.

Figura 26 – Trajetória Vertical.



Fonte: Elaborado pelo autor (2021).

O centro das variações está um pouco deslocado, ocasionando esse erro de 2.5mm na referência. O *setpint* começa em (0,085m, 0,27m), pois é o ponto inicial que se encontrava a

ponta do braço, então, o controle direcionou a ponto para o inicio da trajetória.

4.4 Resultado trajetória Trapezoidal

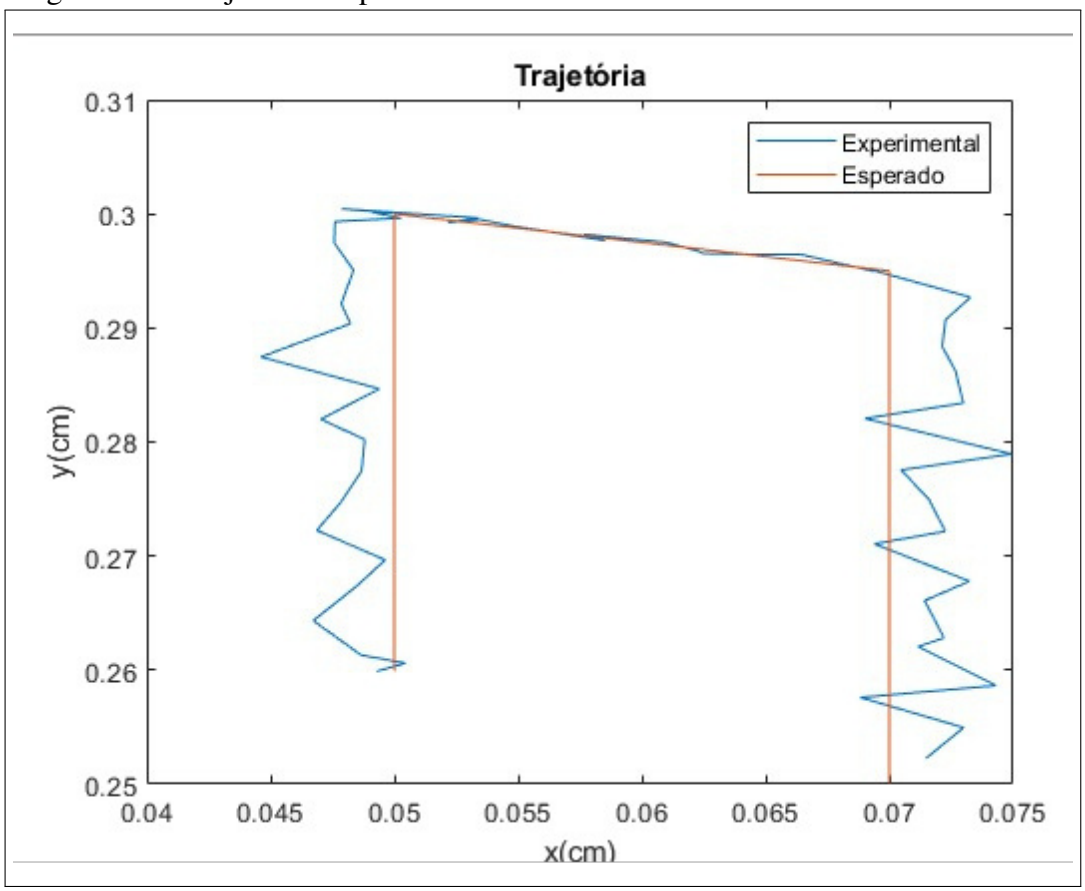
Nesse tópico é mostrado a trajetória trapezoidal no plano XY de coordenadas. O *Input* é a trajetória de controle, e o *Output* é o *setpint*, que é a trajetória experimental obtida.

Na trajetória trapezoidal o resultado foi similar com o da vertical, acontecendo os mesmo desvios por motivos já citados. Pode-se perceber que o controle conseguiu seguir a trajetória, embora tenha algumas pequenas oscilações.

Outro motivo é para a oscilação é a não-linearidades dos servos motores da base do braço robótico, pois são dois servos com direções opostas movimentando apenas um elo, portanto, existem diferenças de movimentos entre os servos que oscilam o elo da base, pois junta as não-linearidades de cada servo para uma única junta.

Entre o intervalo 0.05m e 0.07m ocorreu uma menor variação, seguindo melhor a referência dada

Figura 27 – Trajetória Trapezoidal.



Fonte: Elaborado pelo autor (2021).

5 CONCLUSÕES E TRABALHOS FUTUROS

O trabalho apresentou uma estratégia de identificação e controle de um manipulador robótico articulado com 2 graus de liberdade e utilizando sensores acelerômetros e giroscópios para a estimação angular de cada junta. É importante atentar que as estratégias foram aplicadas em uma planta real, que foi montada com hardware de baixo custo.

Foram realizadas avaliações de índices de desempenhos nos experimentos, a ultrapassagem máxima foi de 3,3%, sendo o máximo permitido 5%, o tempo de subida máximo foi 0,86s e o máximo aceitável é 2s. Os resultados foram satisfatórios, obtendo um controle de ambas as juntas.

Além disso, na parte de controle de posição, foi possível realizar trajetórias com um nível bom de precisão tanto na trajetória vertical quanto na trapezoidal, ambas com erro máximo de 5mm do esperado.

Como proposta de trabalho futuro, objetivando melhorar a precisão do braço e do controle seria interessante realizar um controle não-linear para cada elo para se adaptar as não-linearidades dos servos. Ainda como proposto pode-se adicionar outros controladores com o intuito de fazer uma análise comparativa com os resultados apresentados. Com isso, é possível aperfeiçoar o projeto.

REFERÊNCIAS

AGUIRRE, L. A. **Introdução à identificação de sistemas–Técnicas lineares e não-lineares aplicadas a sistemas reais**. [S.l.]: Editora UFMG, 2004.

BATISTA, J. G. Manipulador pneumático de 2 eixos com clp. Fortaleza, 2009.

BATISTA, J. G. Manipulador robótico didático, descrição: Desenvolvimento, a implementação de um protótipo de um manipulador robótico scara didático para ensino e pesquisas na área de robótico. Fortaleza, 2018.

CHAPRA, S. C.; CANALE, R. P. **Métodos Numéricos para Engenharia-7ª Edição**. [S.l.]: McGraw Hill Brasil, 2016.

COELHO, A. A. R.; COELHO, L. dos S. **Identificação de sistemas dinâmicos lineares**. [S.l.: s.n.], 2004.

CORKE, P. **Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised**. [S.l.]: Springer, 2017. v. 118.

CRAIG, J. J. **Introduction to robotics: mechanics and control, 3/E**. [S.l.]: Pearson Education India, 2009.

EBRAHIMI, N. Modeling, simulation and control of a robotic arm. engrXiv, 2019.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Sistemas de controle para engenharia**. [S.l.]: Bookman Editora, 2013.

INDUSTRIA, C. **La Digitalización y la Industria 4.0**. 2017. Disponível em: <https://industria.ccoo.es/4290fc51a3697f785ba14fce86528e10000060.pdf={28mar.202021},not>.

IVENSENSE SEMICONDUCTORS. **MPU-6000 and MPU-6050 Product Specification transceiver**. [S.l.], 2013. Rev. 3.4.

LEE, J. **Introduction to Robot Design**. 2018. Disponível em: <https://slideplayer.com/slide/17013910/>. Acesso em: 03 fev. 2021.

ROBÓTICA, F. de. Antonio barrientos, luis felipe peñin, carlos balaguer y rafael aracil. **Edición de**, 1997.

SANTOS, V. M. Robótica industrial. **Universidade de Aveito-Departamento de Engenharia Mecânica**, 2004.

SPONG, M. W.; VIDYASAGAR, M. **Robot dynamics and control**. [S.l.]: John Wiley & Sons, 2008.

SPONG, M. W.; VIDYASAGAR, M. **Robot dynamics and control**. [S.l.]: John Wiley & Sons, 2008.

TANGIRALA, A. K. **Principles of system identification: theory and practice**. [S.l.]: Crc Press, 2018.

TOWERPRO. **MG996R High Torque Metal Gear Dual Ball Bearing Servo**. [S.l.], 2010. Rev. 1.

TRONCO, M. L. **Robôs Industriais**. [S.l.]: John Wiley & Sons, 2017.

WB SENSORS. **WDD35D-4**. [S.l.], 2004. Rev. 1.

APÊNDICE A – CÓDIGOS-FONTES UTILIZADOS PARA CONTROLE DO BRAÇO ROBÓTICO

Código-fonte 1 – Código Arduino

```
/* Copyright (C) 2012 Kristian Lauszus, TKJ Electronics.
   All rights reserved.

 This software may be distributed and modified under the
    terms of the GNU
 General Public License version 2 (GPL2) as published by
    the Free Software
 Foundation and appearing in the file GPL2.TXT included in
    the packaging of
 this file. Please note that GPL2 Section 2[b] requires
    that all works based
 on this software must also be made publicly available
    under the terms of
 the GPL2 ("Copyleft").

 Contact information
 -------------------

 Kristian Lauszus, TKJ Electronics
 Web       :  http://www.tkjelectronics.com
 e-mail    :  kristianl@tkjelectronics.com
 */


int32_t Numero = 1500;
char SuaString[10] = "";
int servoF = 1600;
int servoBM = 1600;
```

```
23 int servoBS = (0.09278*servoBM-149.9)*(-9.661)+1468;
24 int32_t ANGX = 0;
25
26 #include <Wire.h>
27 #include <Kalman.h> // Source: https://github.com/
     TKJElectronics/KalmanFilter
28 #include <Servo.h>
29
30 #define RESTRICT_PITCH // Comment out to restrict roll to
       90deg  instead - please read: http://www.freescale.com/
     files/sensors/doc/app_note/AN3461.pdf
31
32 Kalman kalmanX; // Create the Kalman instances
33 Kalman kalmanY;
34
35 Kalman kalmanX2; // Create the Kalman instances
36 Kalman kalmanY2;
37
38 Servo ServBM;
39 Servo ServBS;
40 Servo ServF;// create servo object to control a servo
41 // twelve servo objects can be created on most boards
42
43 /* IMU Data */
44 double accX, accY, accZ;
45 double gyroX, gyroY, gyroZ;
46 int16_t tempRaw;
47
48 double accX2, accY2, accZ2;
49 double gyroX2, gyroY2, gyroZ2;
50 int16_t tempRaw2;
51
```

```
52 double gyroXangle, gyroYangle; // Angle calculate using the
       gyro only
53 double compAngleX, compAngleY; // Calculated angle using a
     complementary filter
54 double kalAngleX, kalAngleY; // Calculated angle using a
     Kalman filter
55
56 double gyroXangle2, gyroYangle2; // Angle calculate using
     the gyro only
57 double compAngleX2, compAngleY2; // Calculated angle using
     a complementary filter
58 double kalAngleX2, kalAngleY2; // Calculated angle using a
     Kalman filter
59
60 uint32_t timer;
61 uint8_t i2cData[14]; // Buffer for I2C data
62
63 uint32_t timer2;
64 uint8_t i2cData2[14]; // Buffer for I2C data
65
66 // TODO: Make calibration routine
67
68 void setup() {
69
70   ServF.attach(5);  // attaches the servo on pin 6 to the
       servo object
71   ServBM.attach(6);
72   ServBS.attach(7);
73   ServF.writeMicroseconds(servoF);
74   ServBM.writeMicroseconds(servoBM);
75   ServBS.writeMicroseconds(servoBS);
76   Serial.begin(115200);
```

```
77    Wire.begin();
78  #if ARDUINO >= 157
79    Wire.setClock(400000UL); // Set I2C frequency to 400kHz
80  #else
81    TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C
          frequency to 400kHz
82  #endif
83
84    i2cData[0] = 7; // Set the sample rate to 1000Hz - 8kHz
          /(7+1) = 1000Hz
85    i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc
          filtering, 256 Hz Gyro filtering, 8 KHz sampling
86    i2cData[2] = 0x00; // Set Gyro Full Scale Range to
           250deg /s
87    i2cData[3] = 0x00; // Set Accelerometer Full Scale Range
          to  2g
88    while (i2cWrite(0x19, i2cData, 4, false)); // Write to
          all four registers at once
89    while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis
          gyroscope reference and disable sleep mode
90
91    while (i2cRead(0x75, i2cData, 1));
92    if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
93      Serial.print(F("Error reading sensor"));
94      while (1);
95    }
96
97    i2cData2[0] = 7; // Set the sample rate to 1000Hz - 8kHz
          /(7+1) = 1000Hz
98    i2cData2[1] = 0x00; // Disable FSYNC and set 260 Hz Acc
          filtering, 256 Hz Gyro filtering, 8 KHz sampling
```

```
99    i2cData2[2] = 0x00; // Set Gyro Full Scale Range to
          250deg /s
100   i2cData2[3] = 0x00; // Set Accelerometer Full Scale Range
          to 2g
101   while (i2cWrite2(0x19, i2cData2, 4, false)); // Write to
          all four registers at once
102   while (i2cWrite2(0x6B, 0x01, true)); // PLL with X axis
          gyroscope reference and disable sleep mode
103
104   while (i2cRead2(0x75, i2cData2, 1));
105   if (i2cData2[0] != 0x68) { // Read "WHO_AM_I" register
106     Serial.print(F("Error reading sensor"));
107     while (1);
108   }
109
110   delay(100); // Wait for sensor to stabilize
111
112   /* Set kalman and gyro starting angle */
113   while (i2cRead(0x3B, i2cData, 6));
114   accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
115   accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
116   accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
117
118   // Source: http://www.freescale.com/files/sensors/doc/
          app_note/AN3461.pdf eq. 25 and eq. 26
119   // atan2 outputs the value of -  to  (radians) - see
          http://en.wikipedia.org/wiki/Atan2
120   // It is then converted from radians to degrees
121 #ifdef RESTRICT_PITCH // Eq. 25 and 26
122   double roll  = atan2(accY, accZ) * RAD_TO_DEG;
123   double pitch = atan(-accX / sqrt(accY * accY + accZ *
          accZ)) * RAD_TO_DEG;
```

```
124 #else // Eq. 28 and 29
125   double roll  = atan(accY / sqrt(accX * accX + accZ * accZ
        )) * RAD_TO_DEG;
126   double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
127 #endif
128
129   kalmanX.setAngle(roll); // Set starting angle
130   kalmanY.setAngle(pitch);
131   gyroXangle = roll;
132   gyroYangle = pitch;
133   compAngleX = roll;
134   compAngleY = pitch;
135
136   timer = micros();
137
138
139   /* Set kalman and gyro starting angle */
140   while (i2cRead2(0x3B, i2cData2, 6));
141   accX2 = (int16_t)((i2cData2[0] << 8) | i2cData2[1]);
142   accY2 = (int16_t)((i2cData2[2] << 8) | i2cData2[3]);
143   accZ2 = (int16_t)((i2cData2[4] << 8) | i2cData2[5]);
144
145   // Source: http://www.freescale.com/files/sensors/doc/
        app_note/AN3461.pdf eq. 25 and eq. 26
146   // atan2 outputs the value of -   to    (radians) - see
        http://en.wikipedia.org/wiki/Atan2
147   // It is then converted from radians to degrees
148 #ifdef RESTRICT_PITCH // Eq. 25 and 26
149   double roll2  = atan2(accY2, accZ2) * RAD_TO_DEG;
150   double pitch2 = atan(-accX2 / sqrt(accY2 * accY2 + accZ2
        * accZ2)) * RAD_TO_DEG;
151 #else // Eq. 28 and 29
```

```
152   double roll2  = atan(accY2 / sqrt(accX2 * accX2 + accZ2 *
          accZ2)) * RAD_TO_DEG;
153   double pitch2 = atan2(-accX2, accZ2) * RAD_TO_DEG;
154 #endif
155
156   kalmanX2.setAngle(roll2); // Set starting angle
157   kalmanY2.setAngle(pitch2);
158   gyroXangle2 = roll2;
159   gyroYangle2 = pitch2;
160   compAngleX2 = roll2;
161   compAngleY2 = pitch2;
162
163   timer2 = micros();
164
165
166
167 }
168
169 void loop() {
170   /* Update all the values */
171   while (i2cRead(0x3B, i2cData, 14));
172   accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
173   accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
174   accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
175   tempRaw = (int16_t)((i2cData[6] << 8) | i2cData[7]);
176   gyroX = (int16_t)((i2cData[8] << 8) | i2cData[9]);
177   gyroY = (int16_t)((i2cData[10] << 8) | i2cData[11]);
178   gyroZ = (int16_t)((i2cData[12] << 8) | i2cData[13]);;
179
180   double dt = (double)(micros() - timer) / 1000000; //
        Calculate delta time
181   timer = micros();
```

```
182
183   // Source: http://www.freescale.com/files/sensors/doc/
          app_note/AN3461.pdf eq. 25 and eq. 26
184   // atan2 outputs the value of -   to    (radians) - see
          http://en.wikipedia.org/wiki/Atan2
185   // It is then converted from radians to degrees
186 #ifdef RESTRICT_PITCH // Eq. 25 and 26
187   double roll  = atan2(accY, accZ) * RAD_TO_DEG;
188   double pitch = atan(-accX / sqrt(accY * accY + accZ *
          accZ)) * RAD_TO_DEG;
189 #else // Eq. 28 and 29
190   double roll  = atan(accY / sqrt(accX * accX + accZ * accZ
          )) * RAD_TO_DEG;
191   double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
192 #endif
193
194   double gyroXrate = gyroX / 131.0; // Convert to deg/s
195   double gyroYrate = gyroY / 131.0; // Convert to deg/s
196
197 #ifdef RESTRICT_PITCH
198   // This fixes the transition problem when the
          accelerometer angle jumps between -180 and 180 degrees
199   if ((roll < -90 && kalAngleX > 90) || (roll > 90 &&
          kalAngleX < -90)) {
200     kalmanX.setAngle(roll);
201     compAngleX = roll;
202     kalAngleX = roll;
203     gyroXangle = roll;
204   } else
205     kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); //
          Calculate the angle using a Kalman filter
206
```

```
207   if (abs(kalAngleX) > 90)
208     gyroYrate = -gyroYrate; // Invert rate, so it fits the
          restriced accelerometer reading
209   kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
210 #else
211   // This fixes the transition problem when the
        accelerometer angle jumps between -180 and 180 degrees
212   if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 &&
        kalAngleY < -90)) {
213     kalmanY.setAngle(pitch);
214     compAngleY = pitch;
215     kalAngleY = pitch;
216     gyroYangle = pitch;
217   } else
218     kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt); //
          Calculate the angle using a Kalman filter
219
220   if (abs(kalAngleY) > 90)
221     gyroXrate = -gyroXrate; // Invert rate, so it fits the
          restriced accelerometer reading
222   kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); //
        Calculate the angle using a Kalman filter
223 #endif
224
225   gyroXangle += gyroXrate * dt; // Calculate gyro angle
        without any filter
226   gyroYangle += gyroYrate * dt;
227   //gyroXangle += kalmanX.getRate() * dt; // Calculate gyro
        angle using the unbiased rate
228   //gyroYangle += kalmanY.getRate() * dt;
229
```

```
230   compAngleX = 0.93 * (compAngleX + gyroXrate * dt) + 0.07
        * roll; // Calculate the angle using a Complimentary
        filter
231   compAngleY = 0.93 * (compAngleY + gyroYrate * dt) + 0.07
        * pitch;
232
233   // Reset the gyro angle when it has drifted too much
234   if (gyroXangle < -180 || gyroXangle > 180)
235     gyroXangle = kalAngleX;
236   if (gyroYangle < -180 || gyroYangle > 180)
237     gyroYangle = kalAngleY;
238
239   /* Print Data */
240 #if 0 // Set to 1 to activate
241   Serial.print(accX); Serial.print("\t");
242   Serial.print(accY); Serial.print("\t");
243   Serial.print(accZ); Serial.print("\t");
244
245   Serial.print(gyroX); Serial.print("\t");
246   Serial.print(gyroY); Serial.print("\t");
247   Serial.print(gyroZ); Serial.print("\t");
248
249   Serial.print("\t");
250 #endif
251 //
252 ////  Serial.print(roll); Serial.print("\t");
253 ////  Serial.print(gyroXangle); Serial.print("\t");
254 ////  Serial.print(compAngleX); Serial.print("\t");
255 //  Serial.print(kalAngleX); Serial.print("\t");
256
257 //  Serial.print("\t");
258 //
```

```
259 //  Serial.print(pitch); Serial.print("\t");
260 //  Serial.print(gyroYangle); Serial.print("\t");
261 //  Serial.print(compAngleY); Serial.print("\t");
262 //  Serial.print(kalAngleY); Serial.print("\t");
263
264 #if 0 // Set to 1 to print the temperature
265   Serial.print("\t");
266
267   double temperature = (double)tempRaw / 340.0 + 36.53;
268   Serial.print(temperature); Serial.print("\t");
269 #endif
270
271 //  Serial.print("\r\n");
272   delay(2);
273
274  /* Update all the values */
275   while (i2cRead2(0x3B, i2cData2, 14));
276   accX2 = (int16_t)((i2cData2[0] << 8) | i2cData2[1]);
277   accY2 = (int16_t)((i2cData2[2] << 8) | i2cData2[3]);
278   accZ2 = (int16_t)((i2cData2[4] << 8) | i2cData2[5]);
279   tempRaw2 = (int16_t)((i2cData2[6] << 8) | i2cData2[7]);
280   gyroX2 = (int16_t)((i2cData2[8] << 8) | i2cData2[9]);
281   gyroY2 = (int16_t)((i2cData2[10] << 8) | i2cData2[11]);
282   gyroZ2 = (int16_t)((i2cData2[12] << 8) | i2cData2[13]);;
283
284   double dt2 = (double)(micros() - timer2) / 1000000; //
        Calculate delta time
285   timer2 = micros();
286
287   // Source: http://www.freescale.com/files/sensors/doc/
        app_note/AN3461.pdf eq. 25 and eq. 26
```

```
288   // atan2 outputs the value of -   to    (radians) - see
         http://en.wikipedia.org/wiki/Atan2
289   // It is then converted from radians to degrees
290 #ifdef RESTRICT_PITCH // Eq. 25 and 26
291   double roll2  = atan2(accY2, accZ2) * RAD_TO_DEG;
292   double pitch2 = atan(-accX2 / sqrt(accY2 * accY2 + accZ2
         * accZ2)) * RAD_TO_DEG;
293 #else // Eq. 28 and 29
294   double roll2  = atan(accY2 / sqrt(accX2 * accX2 + accZ2 *
          accZ2)) * RAD_TO_DEG;
295   double pitch2 = atan2(-accX2, accZ2) * RAD_TO_DEG;
296 #endif
297
298   double gyroXrate2 = gyroX2 / 131.0; // Convert to deg/s
299   double gyroYrate2 = gyroY2 / 131.0; // Convert to deg/s
300
301 #ifdef RESTRICT_PITCH
302   // This fixes the transition problem when the
         accelerometer angle jumps between -180 and 180 degrees
303   if ((roll2 < -90 && kalAngleX2 > 90) || (roll2 > 90 &&
         kalAngleX2 < -90)) {
304     kalmanX2.setAngle(roll2);
305     compAngleX2 = roll2;
306     kalAngleX2 = roll2;
307     gyroXangle2 = roll2;
308   } else
309     kalAngleX2 = kalmanX2.getAngle(roll2, gyroXrate2, dt2);
            // Calculate the angle using a Kalman filter
310
311   if (abs(kalAngleX2) > 90)
312     gyroYrate2 = -gyroYrate2; // Invert rate, so it fits
         the restriced accelerometer reading
```

```
313   kalAngleY2 = kalmanY2.getAngle(pitch2, gyroYrate2, dt2);
314 #else
315   // This fixes the transition problem when the
         accelerometer angle jumps between -180 and 180 degrees
316   if ((pitch2 < -90 && kalAngleY2 > 90) || (pitch2 > 90 &&
         kalAngleY2 < -90)) {
317     kalmanY2.setAngle(pitch2);
318     compAngleY2 = pitch2;
319     kalAngleY2 = pitch2;
320     gyroYangle2 = pitch2;
321   } else
322     kalAngleY2 = kalmanY2.getAngle(pitch2, gyroYrate2, dt2)
           ; // Calculate the angle using a Kalman filter
323
324   if (abs(kalAngleY2) > 90)
325     gyroXrate2 = -gyroXrate2; // Invert rate, so it fits
           the restriced accelerometer reading
326   kalAngleX2 = kalmanX2.getAngle(roll2, gyroXrate2, dt2);
         // Calculate the angle using a Kalman filter
327 #endif
328
329   gyroXangle2 += gyroXrate2 * dt2; // Calculate gyro angle
         without any filter
330   gyroYangle2 += gyroYrate2 * dt2;
331   //gyroXangle2 += kalmanX2.getRate() * dt2; // Calculate
         gyro angle using the unbiased rate
332   //gyroYangle2 += kalmanY2.getRate() * dt2;
333
334   compAngleX2 = 0.93 * (compAngleX2 + gyroXrate2 * dt2) +
         0.07 * roll2; // Calculate the angle using a
         Complimentary filter
```

```
335   compAngleY2 = 0.93 * (compAngleY2 + gyroYrate2 * dt2) +
          0.07 * pitch2;
336
337   // Reset the gyro angle when it has drifted too much
338   if (gyroXangle2 < -180 || gyroXangle2 > 180)
339     gyroXangle2 = kalAngleX2;
340   if (gyroYangle2 < -180 || gyroYangle2 > 180)
341     gyroYangle2 = kalAngleY2;
342
343   /* Print Data */
344 #if 0 // Set to 1 to activate
345   Serial.print(accX2); Serial.print("\t");
346   Serial.print(accY2); Serial.print("\t");
347   Serial.print(accZ2); Serial.print("\t");
348
349   Serial.print(gyroX2); Serial.print("\t");
350   Serial.print(gyroY2); Serial.print("\t");
351   Serial.print(gyroZ2); Serial.print("\t");
352
353   Serial.print("\t");
354 #endif
355
356 //  Serial.print(roll2); Serial.print("\t");
357 //  Serial.print(gyroXangle2); Serial.print("\t");
358 //  Serial.print(compAngleX2); Serial.print("\t");
359 //  Serial.print(kalAngleX2); Serial.print("\t");
360 //
361 //  Serial.print("\t");
362 //
363 //  Serial.print(pitch2); Serial.print("\t");
364 //  Serial.print(gyroYangle2); Serial.print("\t");
365 //  Serial.print(compAngleY2); Serial.print("\t");
```

```
366 //  Serial.print(kalAngleY2); Serial.print("\t");
367
368 #if 0 // Set to 1 to print the temperature
369 //  Serial.print("\t");
370
371   double temperature2 = (double)tempRaw2 / 340.0 + 36.53;
372   Serial.print(temperature2); Serial.print("\t");
373 #endif
374
375 //  Serial.print("\r\n");
376
377 //LER Octave
378
379   if (Serial.available() > 0) {
380
381      for (int i=0;i<10;i++){
382       SuaString[i] = Serial.read();
383       delay(4);
384    }
385    Numero = atol(SuaString);
386       servoF = Numero/10000;
387       servoBM = Numero - servoF*10000;
388       servoBS = (0.09278*servoBM-149.9)*(-9.661)+1468;
389
390       ANGX = kalAngleX*100;
391      ServF.writeMicroseconds(servoF);
392      ServBM.writeMicroseconds(servoBM);
393      ServBS.writeMicroseconds(servoBS);
394      Serial.println(kalAngleX);
395      Serial.println(kalAngleX2);
396
397    }
```

```
398
399
400 //if (Serial.available() > 0) {
401 //
402 //      for (int i=0;i<5;i++){
403 //       SuaString[i] = Serial.read();
404 //       delay(4);
405 //    }
406 //    Numero = atoi(SuaString);
407 //       myservo.writeMicroseconds(Numero);
408 //       Serial.println(kalAngleX2);
409 //       }
410
411 //FIM LEITURA Octave
412
413   delay(10);
414
415
416 }
```