MODEL AND CONTROL OF DOUBLE LINK FLEXIBLE ROBOTIC MANIPULATOR SYSTEM FOR SPATIAL TRAJECTORY TRACKING FOLLOWING TASK

MOHAMAD IQHMANNABIL BIN KARIM

B.ENG (HONS.) ELECTRICAL ENGINEERING (ELECTRONICS)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: MOHAMAD IQHMANNABIL BIN KARIM
Date of Birth	: 25//01/1997
Title	: MODEL AND CONTROL OF DOUBLE LINK FLEXIBLE ROBOTIC MANIPULATOR SYSTEM FOR SPATIAL TRAJECTORY TRACKING FOLLOWING TASK
Academic Session	: 20211/2022

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

[illegible]	[illegible]
(Student's Signature)	(Supervisor's Signature)
MOHAMAD IQHMANNABIL BIN KARIM	ASSOC PROF DR NORMANIHA BINTI ABD GHANI
970125305105 Date:23/06/2022	Date:

NOTE: * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

[illegible]
(Supervisor's Signature)

Date:

Stamp: ASSOC. PROF. TS. DR. NOR MANIHA ABDUL GHANI
ASSOCIATE PROFESSOR
DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING
UNIVERSITI MALAYSIA PAHANG
LEBUHRAYA TUN RAZAK
26300 GAMBANG, KUANTAN, PAHANG
TEL : +609-424 6087 FAX : +09-549 2689

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Electrical Engineering (Electronics).

[illegible]
____
(Supervisor's Signature)

Full Name	:	ASSOC. PROF. TS. DR. NOR MANIHA ABDUL GHANI
Position	:	ASSOCIATE PROFESSOR DEPARTMENT OF ELECTRICAL ENGINEERING COLLEGE OF ENGINEERING UNIVERSITI MALAYSIA PAHANG LEBUHRAYA TUN RAZAK 26300 GAMBANG, KUANTAN, PAHANG TEL : +609-424 6087 FAX : +09-549 2689
Date	:	

____
(Co-supervisor's Signature)

Full Name	:	
Position	:	
Date	:	

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

[illegible]

____

(Student's Signature)

Full Name	: MOHAMAD IQHMANNABIL BIN KARIM
ID Number	: EA18052
Date	: 23 JUNE 2023

MODEL AND CONTROL OF DOUBLE LINK FLEXIBLE ROBOTIC MANIPULATOR SYSTEM FOR SPATIAL TRAJECTORY TRACKING FOLLOWING TASK

MOHAMAD IQHMANNABIL BIN KARIM

Thesis submitted in fulfillment of the requirements
for the award of the
B.Eng (Hons.) Electrical Engineering (Electronics)

College of Engineering

UNIVERSITI MALAYSIA PAHANG

JUNE 2022

ACKNOWLEDGEMENTS

Alhamdulillah, praise be to Allah S.W.T, our creator. With Allah's blessings, I have successfully completed my Final Year Project session. I would want to thank Him for granting me excellent health and the capacity to complete my Final Year Project peacefully and well.

I want to offer my heartfelt gratitude to everyone who made it possible for me to finish this project. I owe a special thanks to my supervisor, Assoc. Prof. Dr. Normaniha Binti Abd Ghani, for her stimulating suggestions and encouragement, which aided me in completing my project, as well as for their careful and valuable guidance, which was extremely beneficial to my study both theoretically and practically.

The project provided an excellent opportunity for learning and professional growth. As a result, I consider myself a very fortunate student for having been given the opportunity to participate in it. This chance is a significant turning point in my academic career.

Finally, I want to express my gratitude to my family for their love, sacrifice, and unwavering support during my academic career.

ABSTRAK

Dalam kebanyakan sektor semasa, penggunaan manipulator robotik dengan struktur dua pautan mempunyai kesan yang ketara. Penggunaan pengawal dalam manipulator fleksibel baru-baru ini mendapat banyak perhatian dalam pelbagai bidang penyelidikan. Ramai ahli akademik sedang berusaha membangunkan pengawal untuk mengelakkan kesilapan manipulator. Pemodelan dan kawalan manipulator robot fleksibel pautan dua dibentangkan dalam kajian ini. Mengawal pergerakan manipulator pautan berkembar, sebaliknya, telah terbukti sebagai tugas yang mencabar, terutamanya apabila rangka kerja yang fleksibel digunakan. Selain itu, kebanyakan model sistem manipulator fleksibel pautan berkembar tidak dibangunkan berdasarkan perkakasan sebenar. Oleh itu, projek ini bertujuan untuk membangunkan reka bentuk Solidworks untuk manipulator robotik fleksibel pautan berkembar (DLFRM) serta prototaip perkakasan sebenar. Prestasi kedudukan kawalan DLFRM telah dianalisis dan sistem kawalan akan diuji pada prototaip perkakasan. Projek ini dimulakan dengan simulasi kedua-dua sistem kawalan iaitu PID dan FLC. Simulasi adalah reka bentuk daripada Solidworks dan dieksport ke Simulink dan akan ditukar sebagai Simscape. Selebihnya simulasi telah ditambah untuk melengkapkan sistem. Selepas selesai bahagian simulasi, perkakasan untuk setiap sistem kawalan akan dibangunkan. Sambungan untuk manipulator robot adalah reka bentuk dalam Solidwork dan dibina menggunakan cetakan 3D. Hasilnya, dalam simulasi, PID mempunyai prestasi yang lebih baik untuk kedua-dua pautan dalam mengurangkan masa menetap dan masa meningkat iaitu 0.5053s, 0.5308s untuk pautan 1 dan 0.2344s, 0.3799s untuk pautan 2. Walau bagaimanapun, FLC mengatasi sistem kawalan PID untuk kedua-dua pautan dalam mengurangkan overshoot dan ralat keadaan tetap iaitu 14.07%, 0.005 untuk pautan1 dan 11.9817%, 0.0005 untuk pautan2. Bagi hasil perkakasan, PID menunjukkan prestasi yang lebih baik dalam mengurangkan overshoot dan masa menetap untuk kedua-dua pautan iaitu 2.2222%, 1.088s untuk pautan 1 dan 12.2211%, 2.504s untuk pautan 2 . Sebaliknya, FLC mengatasi sistem kawalan PID untuk kedua-dua pautan dalam mengurangkan masa naik dan ralat keadaan tetap iaitu 0.44s, 0 untuk pautan1 dan 0.1531s,1 untuk pautan2 .

ABSTRACT

In most current sectors, the use of a robotic manipulator with a double-link structure has a significant impact. The use of controllers in flexible manipulators has recently gained a lot of attention in a variety of fields of research. Many academics are working on developing controllers to prevent manipulators' errors. The modelling and control of a double-link flexible robot manipulator are presented in this study. Controlling the movement of a double-link manipulator, on the otherhand, has proved to be a challenging task, especially when a flexible framework is used. Moreover, most model of double link flexible manipulator system is not developed based on real hardware. Hence, this project aim is to develop Solidworks design for double link flexible robotics manipulator (DLFRM) as well as real hardware prototype. The control position performance of DLFRM is analysed and the controllers is tested on hardware prototype. This project started with simulation of both of the controllers which is PID and FLC. The simulation was design from Solidworks and exported to Simulink and will converted as Simscape. Then, the hardware for each controller will be validated by using the control parameter in simulation. The joints for robot manipulator are design in Solidwork and built using 3D printing. As a result, in simulation, PID has better performance for both links in reducing settling time and rise time which is 0.5053s, 0.5308s for link1 and 0.2344s, 0.3799s for link 2 respectively. However, FLC outperforms PID controllers for both links in reducing overshoot and steady state error which is 14.07%, 0.005 for link1 and 11.9817%, 0.0005 for link2 respectively. As for hardware results, PID shows better performance in reducing overshoot and settling time for both links which is 2.2222%, 1.088s for link1 and 12.2211%, 2.504s for link2 respectively. On the other hand, FLC outperform PID controller for both links in reducing rise time and steady state error which is 0.44s, 0 for link1 and 0.1531s,1 for link2 respectively.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FLC	Fuzzy Logic Controller
PWM	Pulse Width Modulation
PID	Proportional, Integral, Derivative
SISO	Single Input Single Output
ILA	Iterative Learning Algorithm
SCAR	Selective Compliance Assembly Robot
ADOF	Arm Degree of Freedom
PSO	Particle Swarm Optimization
ABC	Artificial Bee Colony
MATLAB	MATrix LABoratory
SCAR	Selective Compliance Assembly Robot
BZN	Ziegler-Nichols
Tr	Rise Time
Ts	Settling Time
Os	Overshoot
PRB	Pseudo-rigid-Body
PPR	Pulse Per Rotation

CHAPTER 1

INTRODUCTION

1.1 Project Background

Robotic manipulators are a major component in the manufacturing industry. They are used for many reasons including speed, accuracy, and repeatability[1]. A robotic manipulator arm is a programmable electronic and multipurpose mechanical device that performs wide range of function by manipulating materials, components, objects, or tools through predetermined movements. These robotic devices are made up of several jointed sections that create a manipulator in the shape of an arm. A robotic manipulator can move or handle items independently based on the amount of degrees of freedom it has. Axes are another name for degrees of freedom. The value of actuators within a robotic manipulator corresponds to each axis. The number of axis on a robotic manipulator can vary between two to 10 or even more. The majority of industrial robotshave between four and six axis. The most common axis is six since the scope of movements is equivalent to human arm. This gives robots the flexibility they need to automate numerous industrial operations. A robotic manipulator's general structure is made up of rigid links attached by joints. One side of the manipulator is attached to a base, and the other is independent and may be utilized for a variety of robotic tasks. Robotic manipulators come in a variety of types and differ depending on the joint such as Cartesian, cylindrical, polar, articulated, SCARA, and delta configurations [2].

Formerly, robotic manipulator designs were huge and heavy, resulting in inflexible arm and stiff joint designs as a result, their use is restricted to light weights, and their movement is limited. Hence, the traditional design is unfavorable in today's sectors sinceit is inefficient in terms of speed, productivity, and power consumption. Furthermore, a lighter construction has become a prerequisite for every engineering system. Hence, to overcome the problem flexible manipulator was composed. Flexible manipulator is mainly composed of manipulators consisting of flexible and lightweight materials. Lightweight flexible manipulators are widely used in various applications because to the higher payload carrying capacity, lower energy consumption, lower cost of manufacturing, quicker motions, and greater reach as compared to typical industrial rigid robots. Due to a lot of flexibility in its application, the two-link flexible manipulator is more preferable [3]. The behavioral model of a flexible manipulator is determined by the modelling approach and control technique used to regulate a certain parameter. There are numerous control systems available to control the manipulators. According previous studies, the most common method to control the flexible manipulator is PID control, which is used to regulate the motion of the flexible manipulators & take to a present location and also to track the set-points [4]. Another method to control the flexible manipulator are also have been implemented in another studies which is FLC. In controlling nonlinear systems, the approach of fuzzy logic has found a distinguished position with its systematic capability of inclusion of human linguistic knowledge in the controller design [5]. However, the methods are not commonly used to control the manipulators especially for a double link manipulator.

1.2 Problem statement

In the manufacturing industry, flexible manipulators have been used widely to assists and reduce manual labour. However, there are several questions concerning the performance of flexible manipulators. The performance of the flexible manipulators is crucial in the industry. Hence, there are several issues in improving the performance of the system.

The first issue that obstruct the process of improving the performance of flexible manipulators is most model of double link flexible manipulator system is not developed based on real hardware. The model is developed based on simulation only which mathematical dynamics are sometimes neglected and also the nonlinearities, thus not represent real system. According to previous paper, the flexibility of the flexible manipulator model will be compromised if all nonlinear components are disregarded [6].

The second issue is the flexible manipulator is much more difficult to control and operate than their rigid counterparts due to their flexibility. This issue has become particularly prevalent in the field of space and industrial robots with light and flexible linkages. The flexible manipulators however lead to vibration on the robot controllers. Based on previous research, the constant strain brought on by the vibration can cause structural damage, failure, imbalance, and a decline in performance. The constant strain brought on by the vibration can cause structural damage, failure, imbalance, and a decline in performance [7].

Last but not least, the flexible manipulator system shows results in low accuracy and low precision of the system. Hence, with low accuracy of the system may affect the effectivity of the robot controller. Flexible manipulators can be difficult to manipulate in order to obtain and maintain precise positioning. The concern of this system is the need for exact position, the system's flexibility which causes vibration, the difficulty of generating an accurate model of the system, and the system's non-minimum phase characteristics.

1.3 Objective

This section contains a description of the project's primary objectives. The study's research objectives are as follows:

I. To develop Solidworks design for Double Link Flexible Robotics Manipulator (DLFRM) as well as real hardware prototype.

II. To develop a controller and analyse the control position performance of DLFRM and maximize stiffness of flexible link.

To achieve the aforementioned objective, numerous approaches were examined while investigating other journals to determine the best strategy to developing the system. A numerous researchers have devised several methods for reducing vibration in flexible manipulators.

1.4 Scope of project

I. The mechanical design for this project is limited for double link flexible manipulator.

II. The position and speed control using 2 method which is Matlab Simscape for Simulation and Arduino for Hardware.

III. The system response will be analysed based on Overshoot, Rise time, Settling time and Steady State Error.

1.5 Thesis outline

There are five chapters in this paper. An overview for each chapter is as follows:

Chapter 1: Contains the introduction, project background, objectives and the scope of the project.

Chapter 2: The literature review focuses on the studies and research of the associated previous case study on double link flexible robotic manipulator.

Chapter 3: The methodology of the works is discussed; the flow of the works overall project.

Chapter 4: The results collected based on all required parameters. All recorded data will be summarized and then will be presented.

Chapter 5: Conclusion for overall project. The project achieved objectives and results are concluded. Suggestion for future improvement is also discussed in this section.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter covers the basics of flexible manipulators andthe control strategy that typically employ. Imprecision and vibration suppression as a result of flexibility and other factors are also discussed. There is also a study of the literature on modelling flexible and rigid manipulators. Since it is such a large topic, theresearch reviewed here is only appropriate to modelling double link flexible systems.

2.1.1 Classical Control

In classical control systems, knowledge of the controlled system (plant) is required in the form of a set of algebraic and differential equations, which analytically relate inputs and outputs [8]. The Laplace transform is used in classical control theory to represent systemsand signals. These approaches are heavily reliant on the mathematical model of the controlled item, which might take the shape of a function or an equation group. The increasing complexity of today's system makes them difficult to employ.

2.1.2 Intelligent Control

Artificial intelligence-based control, often known as intelligent control, is a promising path of modern control strategy. Intelligent control is a computationally efficient procedure of directing a complex system with incomplete and inadequate representationand under incomplete specifications of how to do this in an uncertain environment towarda certain goal [9]. Intelligent control is using abstract modelling which is the user gives the system the behaviour, and the system attempt to broadly define it in which the attemptis to mimic significant aspects of human intelligence. Among these features include adaptability and learning, planning under high uncertainty, and dealing with massive volumes of data. Presently, intelligent control refers to anything that is not classified as conventional control. Intelligent control is multidisciplinary in the sense that it incorporates and expands concepts and approaches from several disciplines like as control, computer science, and systems engineering. Intelligent control methodologies are being applied to robotics and automation, communications, manufacturing, and trafficcontrol, to mention but a few areas of application [9].

2.2 Controller Method

Based on previous studies, control system often incorporates 2 types of techniques whichis PID controller and FLC. In PID controller there are various method of tuning that havebeen implemented in previous study such as P-type, ILA and ABC

2.2.1 PID controller

The (PID) Controller is a feedback-based controller that generates an output depending on the features of the error and produces a favourable solution. In a closed loop, PID is employed. It consists of three elements which is P, I, and D. In the process control sector,the PID controller seems to be the most often implemented Controller Strategy. PID controllers are commonly employed in industrial control systems, and they are made up of proportional, integral, and derivative control actions. The PID controller can be utilized as a pneumatic, hydraulic, or mechanical controller, or it might have a simple system formanual tuning [10]. PID stands for proportional integral derivation, and it is used by the controller to assess the regulated variables. From there, the controller compares analyses the feedback to the specified point and sends commands to the controller. The response is determined by the proportional constant based on the present error, the integral constant depending on the total of earlier error, and the derivative variable depending on the pace at which the errors have been evolving.

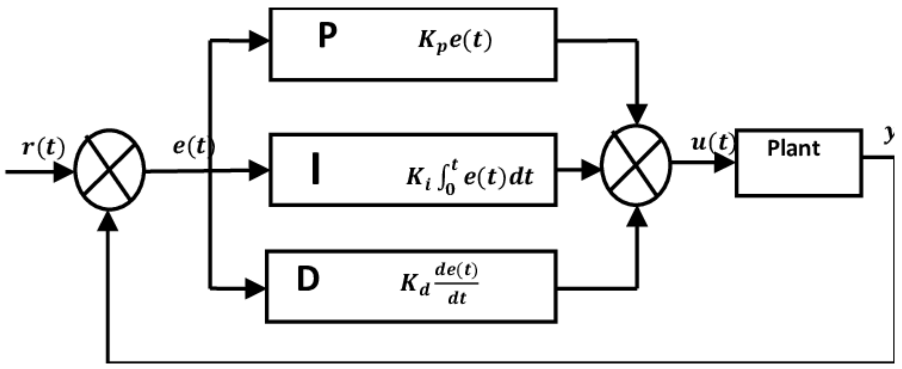


Figure 2-1: PID Controller Configuration [11]

2.2.1.1 P-type ILA

Iterative learning algorithm is a scheme that uses information in previous repetitions to improve the control signal which ultimately enabling a suitable control action [12]. ILA provides consistent results without relying too heavily on control loop feedback. The ILA to boost the effectiveness of the PID control framework. As a result, ILA functions as an adaptive technique that improves the performance of a dynamic system as time passes depending on error criteria. This is classified as an adaptive controller, which overcomes the fixed controller problem. According to [7], the approach proposed that ILA self-tune the PID controller settings in order to reduce the overall error in the systemand improve performance. ILA acts to estimate the parameters for which the system output approaches an appropriate value as time passes. However, as time passed, there was a risk of overlearning among the learning modes. This condition might lead to framework precariousness when it enters a perilous zone [13].To address this weakness,the ILA has a halting condition. Figure 2.1 shows the schematic diagram of ILA tuned by PID controller. The equation used in P-type ILA is as below:

$$K_P(k+1) = K_P(k) + \varphi_1 \times e(k) \tag{2.1}$$

$$K_I(k+1) = K_I(k) + \varphi_2 \times e(k)$$

$$K_D(k+1) = K_D(k) + \varphi_3 \times e(k)$$

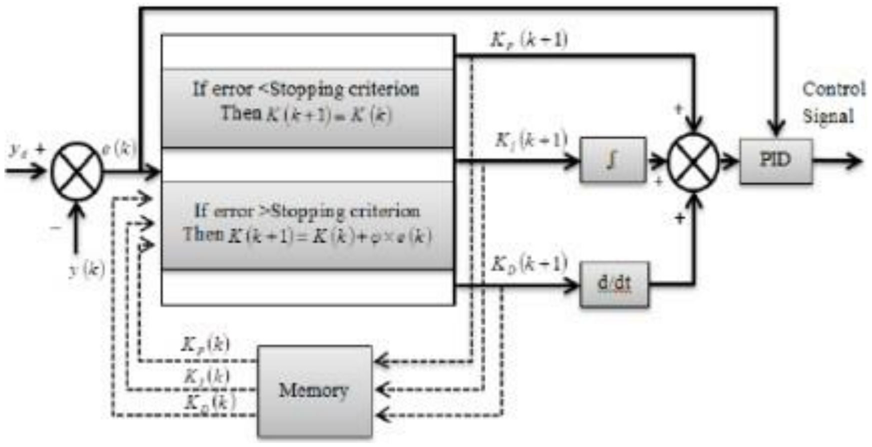

If error <Stopping criterion Then K(k+1) = K(k)
If error >Stopping criterion Then K(k+1) = K(k)+φ×e(k)
$K_P(k+1)$
$K_I(k+1)$
$K_D(k+1)$
y_d
$e(k)$
$y(k)$
∫
d/dt
PID
Control Signal
Memory
$K_P(k)$
$K_I(k)$
$K_D(k)$

Figure 2-2: P-type ILA with PID controller [12]

2.2.1.2 PSO

PSO algorithm was first introduced by Eberhart and Kennedy in 1995. The origin of thePSO inspired of the behaviour of a flock of birds or a school of fish while searching for prey [14]. In this system, the particles move around about in a multi-dimensional searchspace, altering their position based on their own experiences and the experiences of theirsurroundings. The objective is to effectively explore the optimum solution by swarmingthe particles through the best-fitting solution found in earlier rounds, with the purpose ofdiscovering best systems along the way. This method is a numerical solution or algorithm for determining the maximum or minimum of a function working under particular restrictions. According to [15], PSO is able to reduce the error generated during the control process runs. Figure 2.3 shows the block diagram of PSO in PID controller. The particle's velocity and location are adjusted using the equation below:

$$V_{id}^{k+1} = W \times V_{id}^{k} + C_1 \times R_1 \times \left(Y_{id}^{k} - X_{id}^{k}\right) + C_2 \times R_2 \times \left(Y_{id}^{k} - X_{id}^{k}\right) \qquad (2.2)$$

where V= particle velocity, X= particle position, W= Inertia weight,

R1, R2= random number and C1, C2 = learning factors

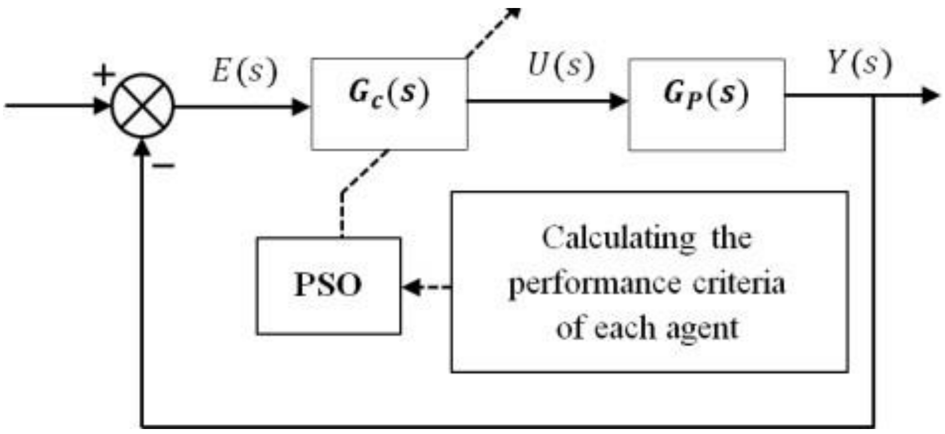


Figure 2-3: PSO in tuning PID controller [16]

2.2.1.3 ABC

Artificial Bee Colony (ABC) is inspired by intelligent behaviour of honey bees developed by Dervis Karaboga in in 2005. Three types of bees: employed bees, onlooker bees, and scout bees [15]. The ABC system combines local search techniques, which arecarried out by worker and onlooker bees, with global search techniques, which are supervised by admirers and scouts, in order to strike a balance between exploration and exploitation. The employed bees make up half of the population in the ABC algorithm, while the onlooker bees make up the other half. It makes use of standard control parameters like colony growth and number of cycles count. The number of employed bees or onlooker bees in the swarm is equal to the number of solutions. The ABC createsan initial population of swarm size (SN) responses that is random selection dispersed. SNdenotes the number of employed or onlooker bees that are assumed equal till the algorithm is completed. Following this initialization, the algorithm's primary cycle is runfor a set number of iterations or until a termination requirement is fulfilled. In ABC, a specific system was designed that permits just one of the parameters of the present solution to be changed. The block diagram of ABC in PID controller are as figure below basedon [17] It should be reminded that at each cycle, only one artificial bee is allowed to become scout and perform the search as follows:

$$x_i^j = x_{min}^j + \varphi(x_{max}^j - x_{min}^j) \tag{2.3}$$

Where φ is a random number in domain [0,1].

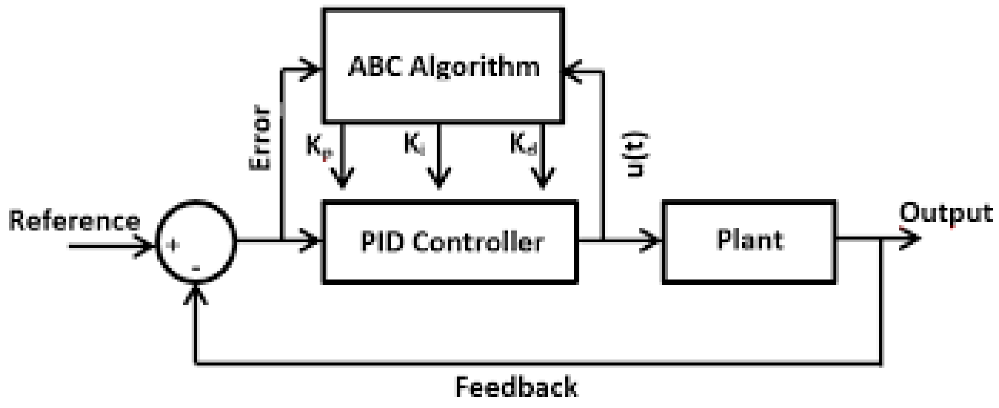


Figure 2-4: ABC in PID controller [18]

2.2.2 Fuzzy Logic Controller (FLC)

Fuzzy logic control is a heuristic approach that easily embeds the knowledge and key elements of human thinking in the design of nonlinear controllers. Qualitative and heuristic considerations, which cannot be handled by conventional control theory, can beused for control purposes in a systematic form, applying fuzzy control concepts [19]. Qualitative and heuristic elements, which may not be solved by traditional control systems, can be utilized in a systematic way for control applications by implementing fuzzy control approaches. Fuzzy logic control would not require a mathematical model, it perform with inaccurate inputs, tolerate non - linearity, and has a higher sensitivity thanother control systems techniques. Fuzzy logic controllers usually outperform other controllers in complex, nonlinear, or undefined systems for which a good practical knowledge exists [19]. Traditional logical systems are significantly closer in approach tological cognition and clear language than fuzzy logic, since that is the logic on which fuzzy control is built. Fundamentally, it is a useful tool for portraying the imprecise, ambiguous quality of the real world. Fuzzy Logic Controller usually takes the form of an iteratively adjusting model.

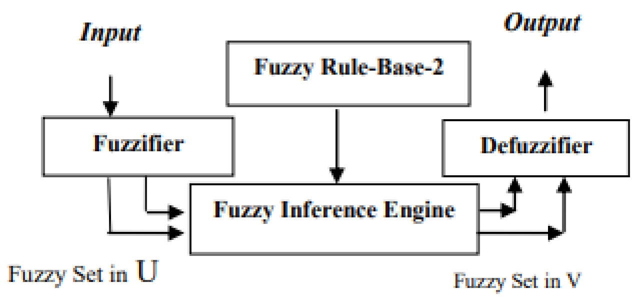


Figure 2-5: Fuzzy Logic Controller [1]

The block diagram for fuzzy logic controller is as in Figure 2.5 above. There are four basic components in the design of fuzzy logic controllers: fuzzifier, knowledge base, inference mechanism, and defuzzifier [20]. Fuzzifier transforms a crisp input signal into fuzzified signals that are identifiable by membership functions. The inference system determines which control rules are applicable at the time and then determines what the input should be. Finally, the defuzzification procedure turns the fuzzily generated control signal into a crisp signal. In addition, the efficiency of fuzzy sets is based on the development of proper membership functions [21]. A membership function is usually limited to a specific given set of data that can be written with a limited number of parameters. The most frequent membership functions are triangular, trapezoidal, and Gaussian.

2.3 Gap analysis

In this subchapter, the previous paper research will be analysed to find the optimum method used for robot manipulator. The paper will be summarized for a better view on each paper. Hence, the result for this project ca be expected from the previous research.

No	Paper Title & Year of Publication	Methodology	Result	Conclusion
1	PID Control of a Double Link (2-link) Flexible Robotic Manipulator (2-DOF) in the 3 DE Space Year: 2018	Models are developed in the Simulink-Matlab environment. Development of the simulink model for a dual link flexible manipulator case PID control, which is used to regulate the motion of the flexible manipulators & take to a present location and also to track the set-points	the base motor – 1 to which link 1 is connected (more control effort needed in this case) because need to carry weight of link 1 and link 2 the base motor – 2 to which link 2 is connected (less control effort needed in this case) because carry only link 2	To conclude, controller was designed for a double link flexible manipulator for the tracking control of the base motor using the set-point & proper tuning of the P, I & D values so that the tracking is done with least error, which can be seen from the output simulated results.
2	Implementation of PID based controller tuned by Evolutionary Algorithm for Double Link Flexible Robotic Manipulator Year: 2018	PSO and ABC to tune the PID controllers' parameters. (Hybrid PID-PID controller)	Lower overshoot using proposed method that is with the improvement of 80.85% and 69.89% respectively as compared with theZN. The settling time is very much faster thatis from 2.97 s to 0.05 s for hub 1 and from 1.46s to 0.043 s for hub 2. Meanwhile, the vibration reduction shows great improvement that is about 96.01% for link 1 and 90.67% forlink 2. Overall, it is revealed that PSO controllers offer the best outcomes compared to ABC	Result simulation divided into 2 sections - hub angle control • PSO and ABC controller achieved a very significance improvement in term rise time, steady state errorand overshoot as compared to Ziegler-Nichols (ZN). - Flexible motion control • The vibration can be further suppressed by employing the ABC and PSO controller as compared to ZN. Overall PSO show superiority over ABC

3	Utilizing P-Type ILA in tuning Hybrid PID Controller for Double Link Flexible Robotic Manipulator Year: 2018	Using tuning Iterative learning algorithm (P-Type ILA) to improve the performance of PID control structure	Hub angle Motion • As a numerical result. It can be noted that PID-ILA control structure for link 1 and 2 were able to track the desired hub-angle of DLFRM. Have Improvement in PID- ILA. • The percentage of improvement achieved by PID-ILA controller compared with PID-PSO controller for tr, ts and Mp are 86.2 %, 44.94 % and 86.21 % for link 1 and 80.95 %, 10.278 % and 17.91 % for link 2 Flexible Motion • The results show that PID tuning throughILA managed to improve the performance of vibration suppression than those obtained by the PSO method	It was noted that PID-ILA control structure performed well as compared to those fixed PID control structure specifically PID-PSO manages to give a good response. The performance of the proposed adaptive PID-ILA control scheme is better than the fixed PID and PID-PSO controller.
4	Optimization Of PID Controller For Double-Link Flexible Robotic Manipulator Using Metaheuristic Algorithms Year:2019	Tuning of PID controller using PSO and ABC	Hub angle control • PSO and ABC controller achieved a very significance improvement in term rise time, steady state error and overshoot in comparison to Ziegler-Nichols (ZN) Flexible motion control • the responses from ABC and PSO tuning methods are having almost the same amplitude of vibration	Results showed that PSO and ABC- based control can achieve satisfactory hub angle response and provide a better suppression in comparison to Ziegler-Nichols (ZN)

5	Controlling The Non-parametric Modelling of Double Link Flexible Robotic Manipulator using Hybrid PID Tuned by P- Type ILA. Year:2018	(P-type ILA) tuning to improve the performance of PID control structure.	Hub angle Motion The rate of enchantment attained by PID-ILA controller compared with PID-PSO controller for tr, ts and Mp are 86.2 %, 44.94 % and 86.21% for link 1 and 80.95 %, 16.95 % and 17.91 % for link 2. The percentage of improvement for flexible body control achieved by PID-ILA controller compared to PID-PSO controller for MSE is 54.15 % and 6.05 % for link 1 and 2 respectively	The outcome appears to be that PID tuning through ILA managed to improve the performance of vibration suppression than those obtained by the PSO strategy
6	Cascade fuzzy logic control of a single-link flexible-joint manipulator Year:2013	Cascade FLC structure in Simulink The controller includes 4 blocks: the measurement block, input block, output block, and controller block The experiments can be grouped into 3 main parts: position control, trajectory tracking control, and robustness testing Compare with PID controller	FLC yields better results than the PID controller Cascade FLC structure was able to suppress link vibrations in a short time (maximum of 2 s). When a longer link was used, more overshoot and oscillations occurred in θ and α Fast trajectory tracking and precise position control were obtained with cascade FLCs in the flexible-joint robot manipulators.	The best response for the end-point deflection angle (reduced overshoot and vibration, and no steady-state error) was attained by using all of the state variables. The settling time of the motor rotation angle was increased. Compared to the PID controllers, the FLC eliminates vibrations in a short time in α angle and no overshoot occurs in the θ angle.

7	Modelling and Control of two link Flexible Space Manipulator Using the Waved-Based Method Year:2017	PRB (Pseudo-rigid-Body) model is used in the description of flexible link. The basic principle of waved based control is to regard the motion as the transmission of wave.	For a single link flexible manipulator, the steady state error due to the symmetry of the joint torque is very small and can be neglected. But for the two link flexible manipulator, due to the interaction between the two links the joint tracking error is relatively large.	The motion of the flexible manipulator war regarded as the propagation of wave and wave-based control method was used to suppress the residual vibration of the flexible manipulator.
8	Performance Analysis Of Pid, Pd And Fuzzy Controllers For Position Control Of 3-Dof Robot Manipulator Year: 2019	Fuzzy logic controller (FLC) using two input triangular membership function.	In link 1, it was discovered that, the PID (RT=0.3149) (ST=0.6361), outperform FLC (RT=1.8507) (ST=3.3064), and FLC (OS=2.22.364e-04) (SS=- 0.001) outperform PID (OS=0.1395) (SS=-0.002). In link 2, it was discovered that, the PID (RT=0.3011) (ST=0.5264) outperform FLC (RT=1.1373) (ST=2.4064), the PID (OS=0.1790) and FLC (OS=7.2402e-04) while the Steady State Error are approximately the same (SS≈-0.002).	When compared to the convectional PD and PID, the FLC performs better in terms of overshoot. However, as compared to the FLC, the PD and PID performed better in terms of Rise Time and Settling Time. Both the convectional PID and the FLC can function effectively in a lower order system.

2.4 Comparison on previous study

Based on the table above, there are various method was proposed to compare the techniques for control system. According to, J. Annisa, I. Z. M. Darus, M. O. Tokhi, and A. S. Z. Abidin, rise time, settling time and overshoot are the lowest when the P-type ILA was implemented to tuning the PID controller compared to PSO and ZN. Hence, theP-type ILA managed to suppress the vibration in contrast with PSO [10]. However, based on previous study A. Jamali, there are more than 70% improvement of vibration suppression from PSO and ABC as compared to ZN [16]. On comparison with ABC andPSO, PSO provide better suppression than ABC. Overall, the best tuning method in PIDcontroller is P-type followed by PSO and ABC. Although this method achieved significant result in suppressing the vibration, the control system inn these days are morecomplex than before. Hence, an improvement from this method needs to be implemented.According to I. H. Akyüz, Z. Bingül, and S. Kizir, fuzzy logic was implemented and shows a better result in flexible manipulator [5]. In the study, with using FLC method tocontrol the flexible manipulator, a fast trajectory tracking, and precise position were obtained. It also shows that the FLC was able to suppress the vibration in a shorter timewhich is in 2 seconds.

In the study by Usman Kabir, Mukhtar Fatihu Hamza, Ado Haruna and Gaddafi Sani Shehu, performance analysis of PID PD and Fuzzy Controller for position control of 3-DOF robot manipulator was presented [22]. According to their findings, PID and PD controller outperforms FLC based on rise time and settling time. However, in terms of overshoot and steady state error, the FLC have better performance in link 1, meanwhile steady state error in link 2 and 3 are approximately the same.

According to paper by Joshi Shubangi Milind, Dr. Arunkumar G.and Dr. T.C. Manjunath on PID control of a double link flexible manipulator in the 3E space, the main objective is to create complex control algorithms for flexible manipulator systems, focusing on motor trajectory control and tip position precision. Flexibility of the links and joints plays a key role in these systems [4]. From their findings, a controller was created utilising the set-point & suitable tuning of the P, I, & D values so that the tracking is done with the least amount of error, as can be seen from the output simulated results. Additionally, it can be observed that the Ziegler-Nicolas approach may be used to properly tune the PID controller's KP, KI, and KD constants such that the control effect is smooth, and the set-points are tracked in a way that reaches the intended target.

In the study by Annisa J., I.Z Mat Darus, M.O Tokhi and S. Mohamaddan, implementation of PID based controller tuned by evolutionary algorithm for double link flexible manipulator was presented [23]. From their study, the method used to tune the PID controller is using PSO and ABC. The results show the set-point and suitable tuning of the P, I, and D values to regulate the base motor's tracking in a way that minimises error, as demonstrated by the output simulated results. Based on another study by the same authors, utilizing P-type ILA in tuning hybrid PID controller for double link flexible robotic manipulator as presented [12]. From the result, it was shown that PID-ILA control structures outperformed fixed PID control structures, particularly PID-PSO, which manages to respond well. In comparison to fixed PID and PID-PSO controllers, the suggested adaptive PID-ILA control method performs better.

According to the study entitled Controlling The Non-parametric Modelling of Double Link Flexible Robotic Manipulator using Hybrid PID Tuned by P- Type ILA by Anissa J., Mat Daru I. Z, M.O. Tokhi and. S.Z.Abidin, a (P-type ILA) tuning is applied to improve the performance of PID control structure [7]. The output shows that PID tuning using ILA was able to enhance vibration suppression performance in comparison to results attained by the PSO technique.

Based on previous study by LI Yanan, MENG Deshan, LIU Houde, WANG Xueian and LIANG Bin, modelling, and control of two link flexible space manipulator using the wave-based method was presented. Flexible links are described using the PRB (Pseudo-rigid-Body) concept [24]. The motion of the flexible manipulator was thought of as the wave propagating, and the residual vibration of the flexible manipulator was suppressed using a wave-based control approach. The efficiency of the suggested approach is demonstrated by simulation results that investigate the wave propagation in the two-link flexible manipulator.

According to the study by S.R.Vaishnav and Z.J. Khan. in design and performance of PID and fuzzy logic controller with smaller rule set for higher order system is comparing Ziegler-Nichol’s method with FLC [25]. From their findings, the simulation results show that compared to the Ziegler Nichols adjusted PID controller and the fine-tuned PID controller, the suggested fuzzy logic controller with its straightforward design method and smaller rule base may give higher performance.

Based on the study [20], [22], [26], when compared to fuzzy logic controllers, PID controllers give responses with shorter delays and rising times, but they also have very long settling times because of their oscillating transient nature. The system performance is damaged by its strong oscillations, which have a very high peak overshoot. These harmful oscillations may be efficiently eliminated by the suggested fuzzy logic controller, which also offers smooth operation throughout the transient period.

Therefore, in this study, a control system using PID controller and FLC will be developed. The system will consist of double link flexile manipulator that will be design in Solidwork. The system will be able to follow the spatial trajectory with minimize overshoot and can lessen the vibration in each link. Subsequently, the result for PID controller and FLC will be analyze on response parameters.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will go through the simulation software that have been used, the design and the flowchart of the system. The PID controller and FLC techniques will be used for controlling the flexible robot manipulator. The techniques will be compared, andthe result will be analysis to know which the best techniques that can be implemented isin the real life.

3.2 Project Management

In project management, a brief summary of the whole research process will be shown below. The significance of project management in a research endeavor cannot beemphasized. When done correctly, it makes every aspect of the project operate more smoothly to focus on the important job without being distracted by projects that go off course or budgets that spiral out of control.

3.3 Gantt Chart

From this Gantt chart, it can be clearly seen that all tasks given or done need to be first decided in a form of graphic table so that all tasks will be done in appropriate manner. One of the most crucial components in making Gantt chart is, deciding the daysallocated for each task and planning the kick-off in doing the task need to be refined carefully so that it creates a mind-set towards the research and a systematic workflow could be accomplished.

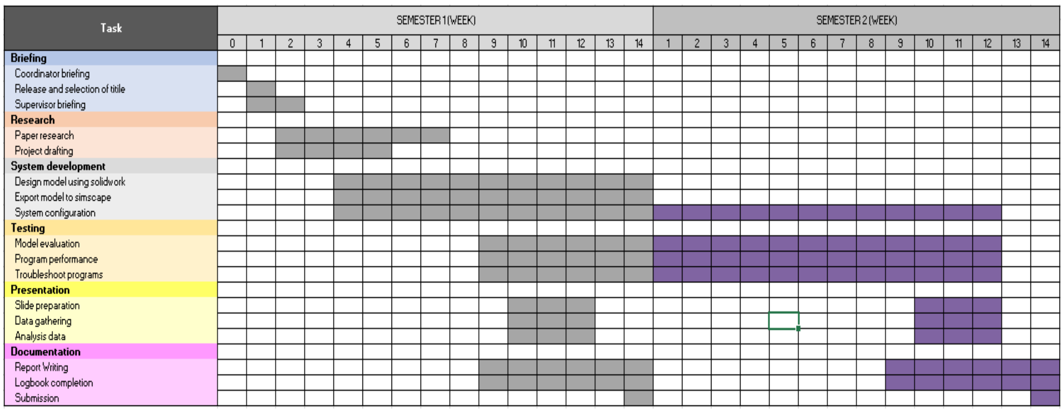


Figure 3-1: Gantt Chart

From figure above, there were about 14 weeks to complete given task by superiorand massive component needed more days to be completed. In general, creating Gantt chart needs to follow these basic steps:

i. Make a task list of all the tasks required to accomplish the project.

ii. Set the start and finish dates for each assignment.

iii. Create a project timetable based on task duration.

iv. Determine task dependencies.

v. Fill in the blanks on the bar chart timeline with the tasks.

vi. Set goals and objectives.

vii. Determine the critical path.

3.4 System Development

3.4.1 Simulation Software

3.4.1.1 Solidworks

Solidworks is a famous software to use for designing any robotic elements. This is due to the feature that have been provided by the software that can make the designing faster.This software was chosen as it had previously been used by other researchers and was able to help shorten robot development time, increase the productivity of the robot designer and improve the speed and quality of robot design [27]. Manually documentingnumerous features of a mechanical component is a time-consuming operation that needshigh degrees of precision in traditional drawing methods. SOLIDWORKS is a simulationmodelling software that allows to alter the design at any point throughout the design process. It can generate advanced photo realistic renderings and animation and the graphics allows to see the design in real time. This will provide an idea of final design will appear before it is completed and will be easier to troubleshoot. This software is also adaptable and user friendly as there is various platform to learn the software. It alsoprovided many libraries that can help in developing any system.

Figure 3-2: Solidworks

3.4.1.2 MATLAB Simulink

Simulink is a graphical extension of MATLAB that allows you to model and simulate systems. Simulink's capacity to describe a nonlinear system, which a transfer function cannot, is one of its key features. Simulink's capacity to take on beginning circumstancesis another benefit [28]. The Simulink was used to produce the graphical output and the analysis of the system.

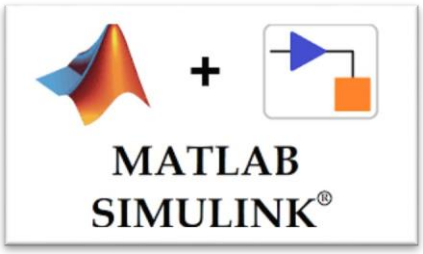


Figure 3-3: Matlab Simulink

3.4.1.3 Simscape

Simscape is a tool inside the Simulink that enables to design physical part of the system.It also compatible with a wide range of platforms. As a result, other design can easily beexported to the environments without the need to reconstruct the design. Simscape aids in the development of control systems as well as the testing of system-level performance.Simscape uses block diagram approach hence make it easier to develop the system without using mathematical approach.

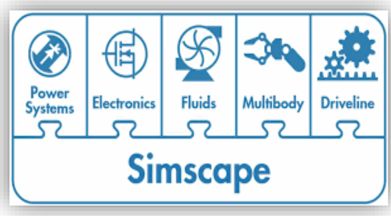


Figure 3-4: Simscape

3.4.2 Design

In this section, the design & development of the simulink model & the control scheme tocontrol / track the speed of the motor for a 2-link flexible manipulator case is being considered [4]. The controller was design in Solidworks software for two link flexible manipulator. Hence, the developed design connected to base of the two-link flexible manipulator using 2 degrees-of-freedom. The use of Solidworks software is to support thecomplex design for the final design. The initial design was adapted from previous study and as an attempt in using the Solidworks to design. The final design will be composed after getting the result of this design. The initial design is a simple design but shows the double link characteristics. Figure 3.5 shows the final design for doublelink flexible robotic manipulator. After the designing step are done, the design was evaluated to make sure that the moveable parts will be able make a move. Next, after ensuring all the parts are in a good state, the design was exported to Simscape toolbox in Simulink to proceed on setup the controller.

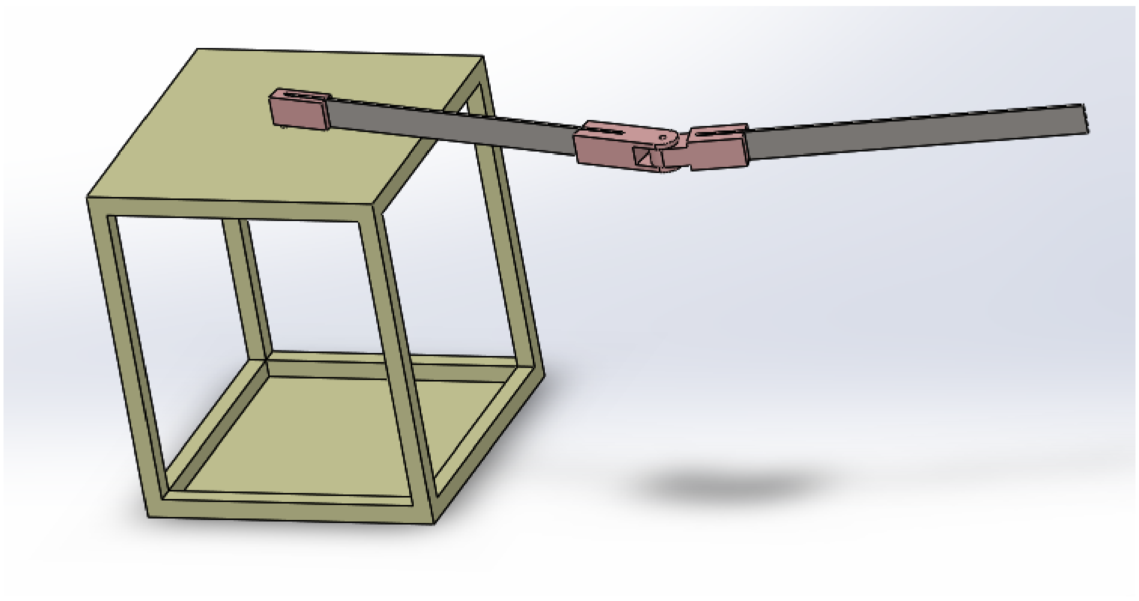
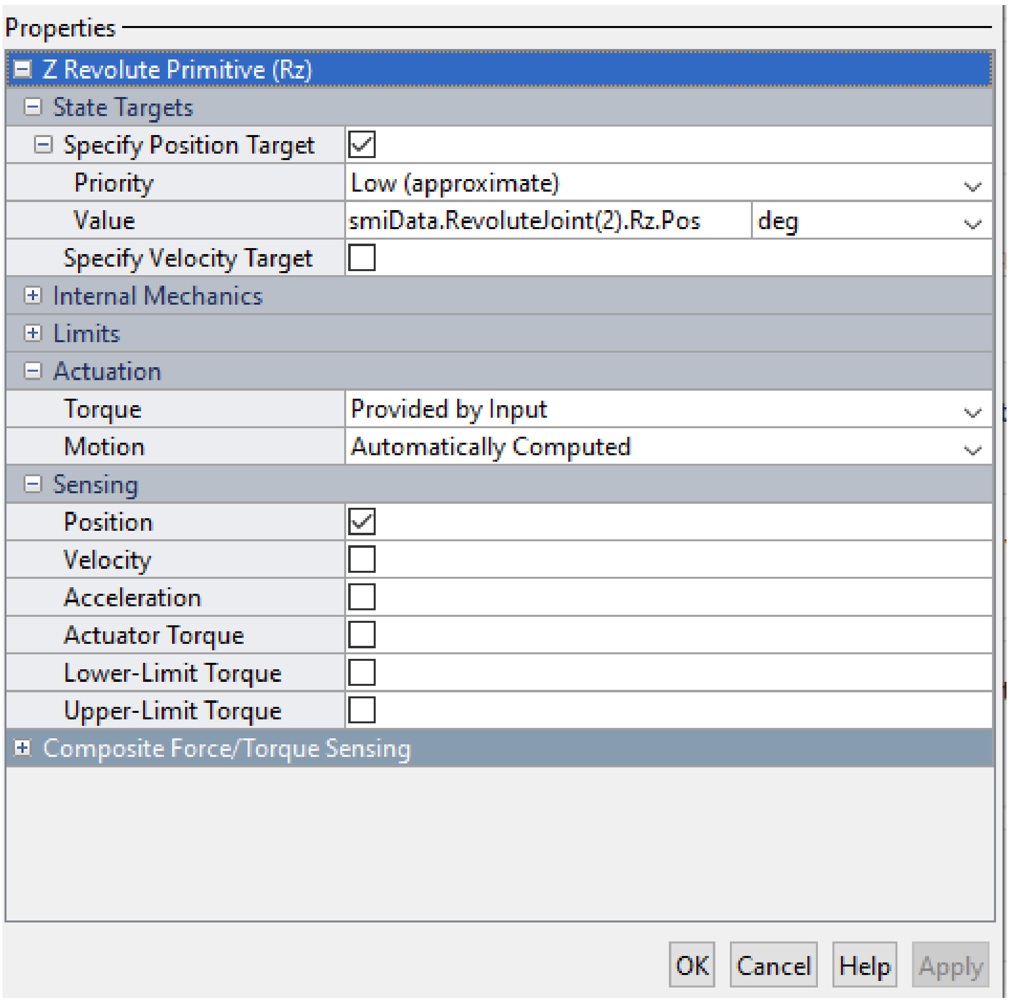


Figure 3-5: Doublelink Robot Manipulator Design

Then, after the design was successfully exported the Simscape, the revolute joint will need to be setting up. For additional information, the revolute joint is consisting of one rotational degree of freedom. Hence, for a double link manipulator, there are two revolute joint that needed to be set up. The use of setting up the revolute joint is to allow the link to follow the input signal given. The input for the revolute joint is actuation torque, t which allow the signal indicates the amount of torque applied to the joint primitive's base and follower frames. The output setting is the position of joint primitive to ensure the output follow the desired position. The setting is changed as Figure 3-6 to compute the desired output.



Properties

⊟ Z Revolute Primitive (Rz)	
⊟ State Targets	
⊟ Specify Position Target	☑
Priority	Low (approximate)
Value	smiData.RevoluteJoint(2).Rz.Pos — deg
Specify Velocity Target	☐
⊞ Internal Mechanics	
⊞ Limits	
⊟ Actuation	
Torque	Provided by Input
Motion	Automatically Computed
⊟ Sensing	
Position	☑
Velocity	☐
Acceleration	☐
Actuator Torque	☐
Lower-Limit Torque	☐
Upper-Limit Torque	☐
⊞ Composite Force/Torque Sensing	

Figure 3-6: Revolute Joint setting

3.4.3 Controller

3.4.3.1 PID Controller

Afterwards, the controller was setting up using Simulink controller. The component thathas been used are signal builder, gain, PID controller and scope. The signal builder wasused to provide an input trajectory signal for the system to follow. Gain was used to movethe link based on gain value. The value of the gain can be equated value of degree angle. PID controller was used to control the movement of both of the links. The output of the system will be given by the scope. Each of the joint are consist of this component to makesure the links can move individually in closed-loop system.

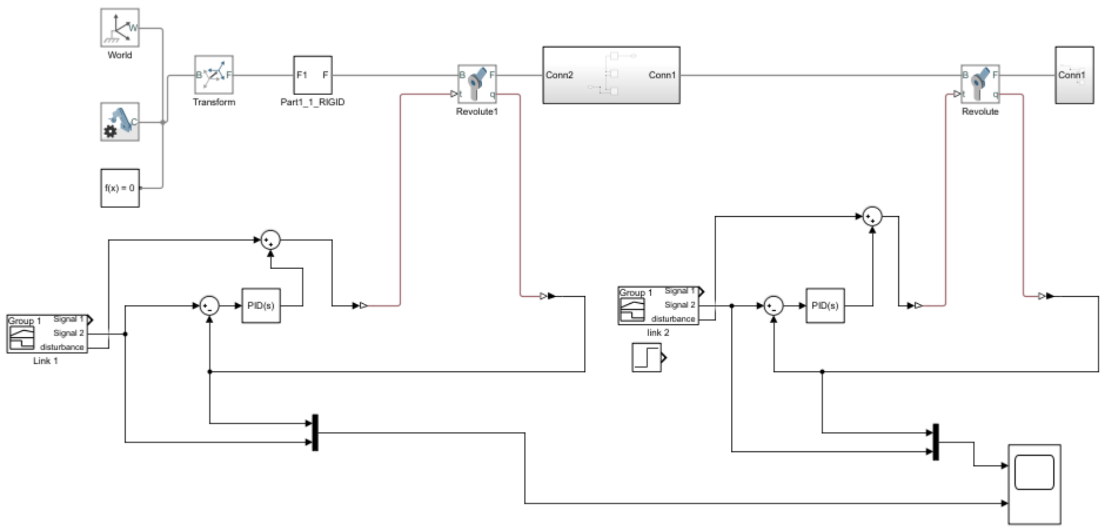


Figure 3-7: Simulink block diagram for PID

3.4.3.2 PID Controller Tuning

Using PID controllers to achieve the spatial trajectory, the controller can be adjusted based on the rate of change of the error signal that can give more accurate and stable control [29]. PID controllers tuning are meant to make things easier by selecting the appropriate tuning settings based on the study of the controlled process's behaviour. Thetuning process that has been implemented is the auto tuning method. The method is using the transfer function as in equation 3.1 and the time response and transient behaviour was tune until graph reference tracking sloping.

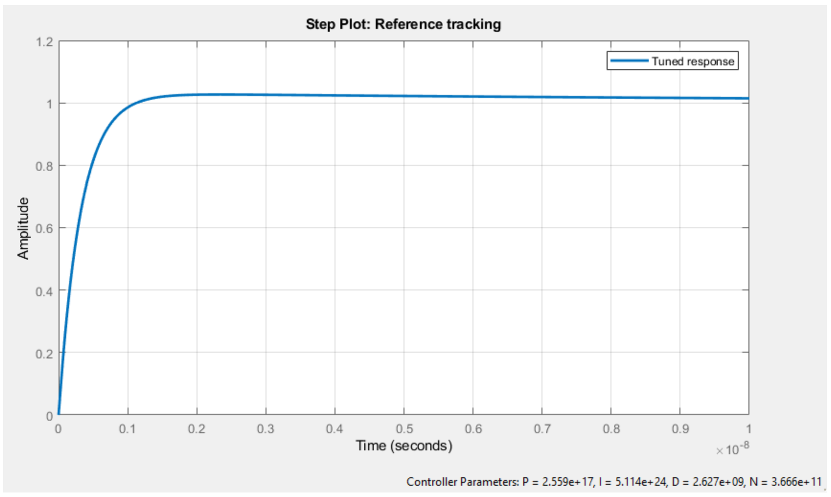


Figure 3-8: Reference tracking

$$C(s) = K_P + \frac{K_I}{s} + sK_D \tag{3.1}$$

Where K_P= Proportional gain, K_I= Integral gain and K_D = Derivative gain and often called as controller parameters.

After tuning into favorable controller parameter, block must be updated to ensure the parameter will be used into the system. This step can be repeated if the output still unstable.

3.4.3.3 Fuzzy Logic Controller, FLC

Generally, block diagram for FLC is as same as PID controller except FLC using fuzzy toolbox. The toolbox helps to describe complicated system behaviors with basic logic rules, which you can subsequently implement in a fuzzy inference system. It can be used independently as a fuzzy inference engine. However, Simulink's fuzzy inference blocks may be utilized to simulate the fuzzy systems as part of a larger model of the overall dynamic system.

Figure 3.9 shows the block diagram for FLC that has been created in MATLAB. From the block diagram. After export the design from the Solidwork, signal builder was added to the system to provide the input as in PID controller design. Next, after the signal builder, fuzzy logic will be implemented as he controller for the system. In fuzzy logic, there are another few things that need to modify such as membership function and fuzzy sets.

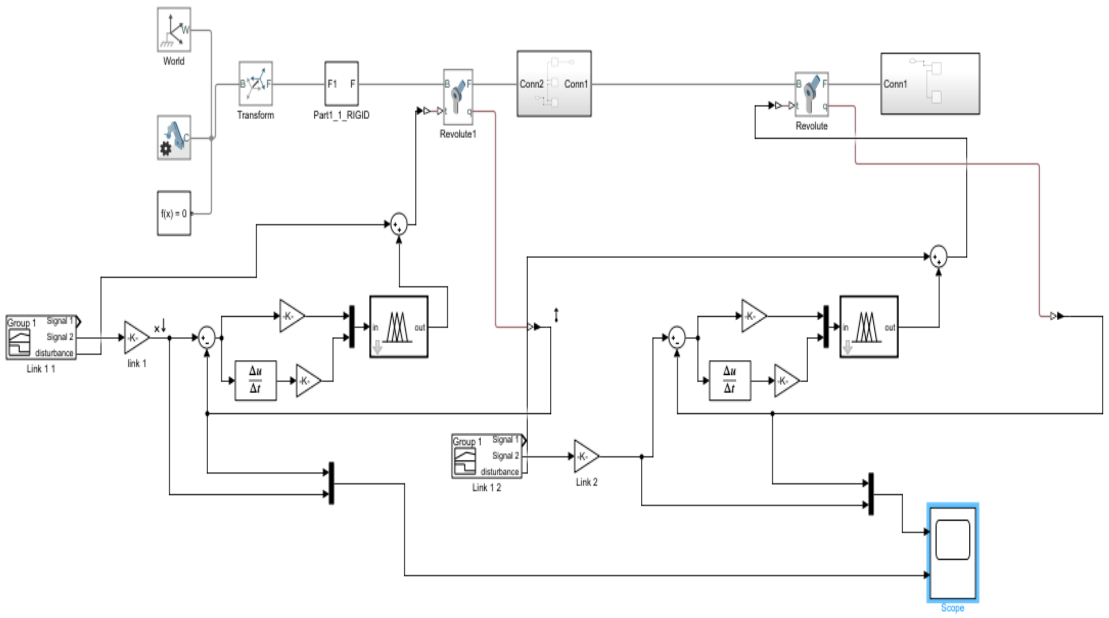
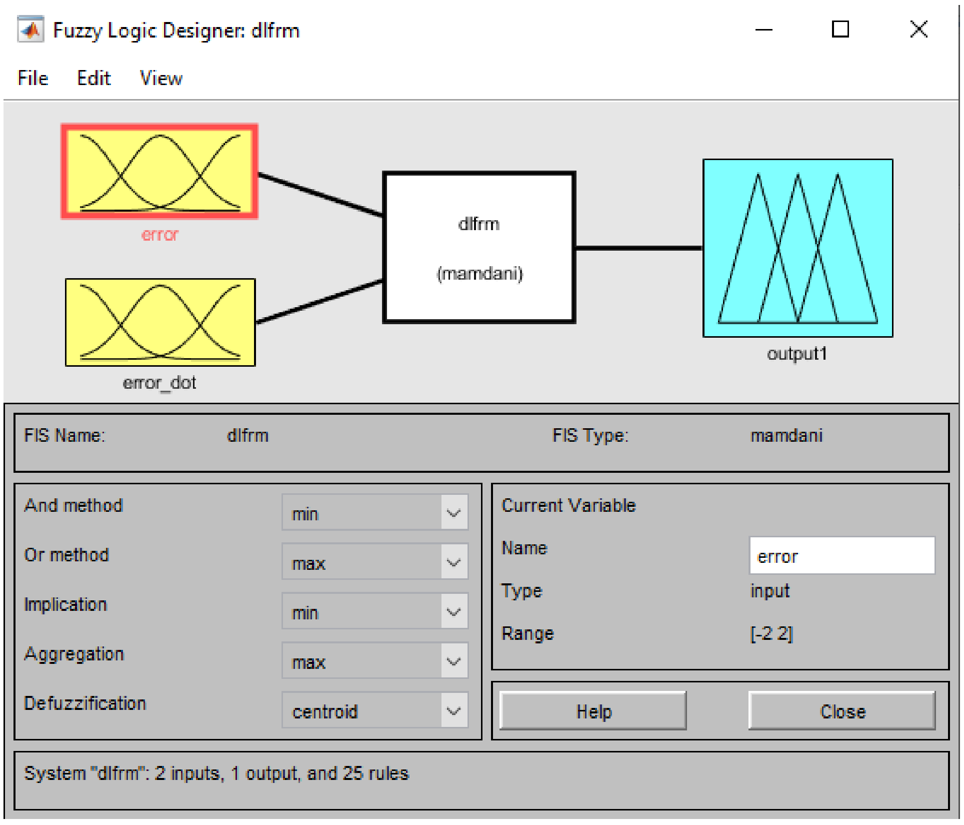


Figure 3-9: Simulink block diagram for FLC

In this system, there are 2 inputs used which is error and error dot. Each input was added into fuzzy logic configuration as Figure 3.10. Each input was given range of – 2 and 2 to fit the range in the system as the system required range large than 1. The FLC system are using Gaussian membership function as in Figure 3.11. The input that has been created will be saved into Fis.file. The configurations can be activated by exporting the Fis.file to workspace and block parameters to execute the system.



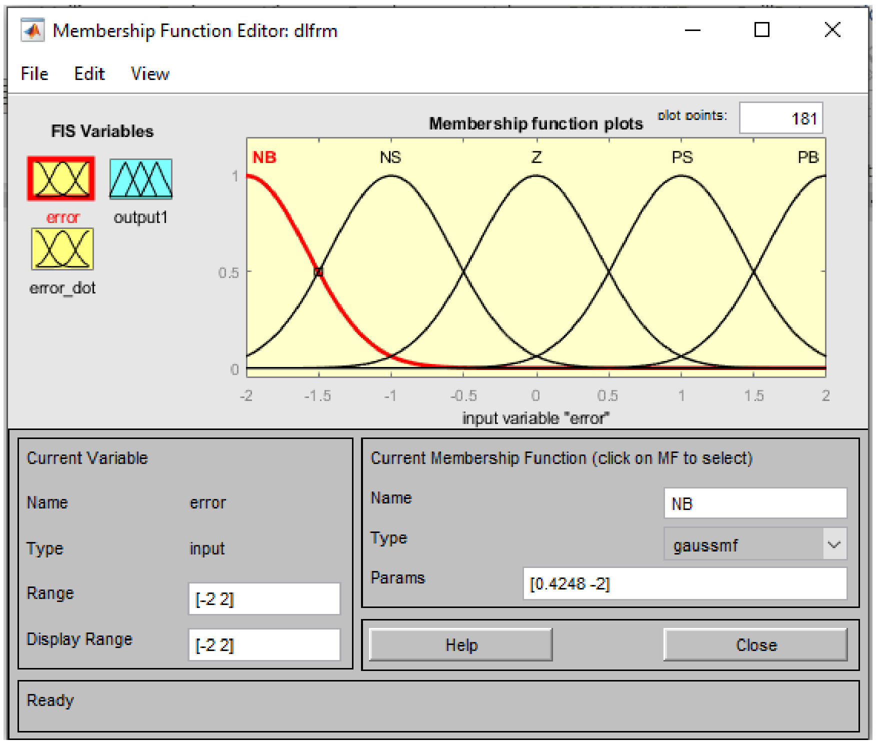
FIS Name: dlfrm
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: error
Type: input
Range: [-2 2]

System "dlfrm": 2 inputs, 1 output, and 25 rules

Figure 3-10: Fuzzy logic configuration



Current Variable

Name	error
Type	input
Range	[-2 2]
Display Range	[-2 2]

Current Membership Function (click on MF to select)

Name	NB
Type	gaussmf
Params	[0.4248 -2]

Ready

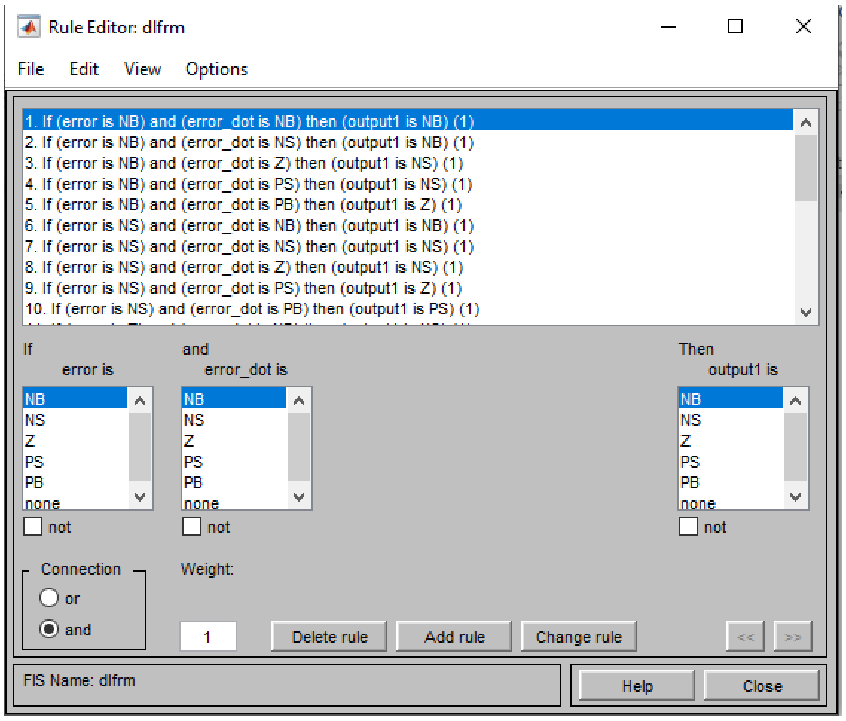
Figure 3-11: Input of the system

Table 3.1 lists the fuzzy controller rule base, and 25 rule bases were developed as shown below where, NB is negative big, NS is negative small, Z is zero, PS is positive small and PB is positive big..

Error / Error Dot	NB	NS	Z	PS	PB
NB	NB	NB	NS	NS	Z
NS	NB	NS	NS	NS	Z
Z	NS	NS	Z	PS	PS
PS	NS	Z	PS	PS	PB
PB	Z	PS	PS	PB	PB

Table 3-1: Rule Notation

IF THEN rule base was used to create the assessments of the algorithm parameters. Table 3.1 shows the options used for linguistic factors which resulted in the combination of 25 rules bases, as shown below.



```
1. If (error is NB) and (error_dot is NB) then (output1 is NB) (1)
2. If (error is NB) and (error_dot is NS) then (output1 is NB) (1)
3. If (error is NB) and (error_dot is Z) then (output1 is NS) (1)
4. If (error is NB) and (error_dot is PS) then (output1 is NS) (1)
5. If (error is NB) and (error_dot is PB) then (output1 is Z) (1)
6. If (error is NS) and (error_dot is NB) then (output1 is NB) (1)
7. If (error is NS) and (error_dot is NS) then (output1 is NS) (1)
8. If (error is NS) and (error_dot is Z) then (output1 is NS) (1)
9. If (error is NS) and (error_dot is PS) then (output1 is Z) (1)
10. If (error is NS) and (error_dot is PB) then (output1 is PS) (1)
```

Connection: or [✓ and]

Weight: 1

FIS Name: dlfrm

Figure 3-12: List of rules

3.5 Hardware Development

3.5.1 Arduino

Arduino is a microcontroller that can be used to write and upload the coding to the physical board. There are many types of Arduino in general, however the most common Arduino is Arduino Uno. Arduino Uno is based on Microchip ATmega328P microcontroller. The board has a number of digital and analogue input or output (I/O) pins that can be used to connect to different expansion boards and other circuits. The board consists of 14 digital I/O pins which six can be used for PWM output and 6 analogue I/O pins, and it can be programmed using the Arduino IDE (Integrated Development Environment) and a USB type B connector[30]. Arduino combined all of the important components of a microcontroller and designed them into a piece of PCB that is incredibly simple to utilize. Arduino is a simple to use platform. When compared to other programming applications, the Arduino is easier to learn to programme because it employs a simplified form of C++.

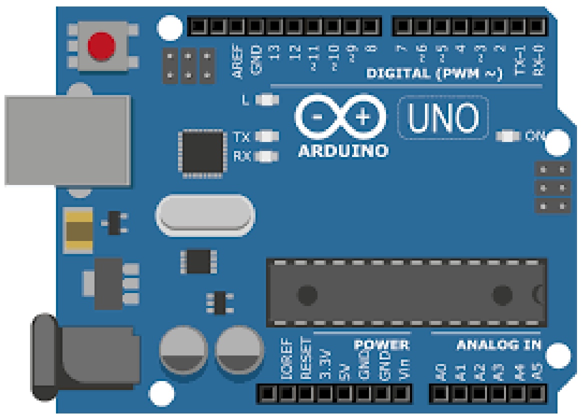


Figure 3-13: Arduino Uno

3.5.2 Motor Driver

Motor drivers serve as a connection between motors and control circuits. The controller circuit operates on low current signals, whereas the motor requires a large amount of current[31]. The purpose of motor drivers is to convert a low-current control signal into a higher-current signal capable of driving a motor. Another typical role of motor drivers is to differentiate low-power electronics from the high-power supply required to run a motor. In this system, L298N was used for hardware implementation. The L298N has a dual H-Bridge motor driver that allows for simultaneous speed and direction control of two DC motors[32]. The module can operate DC motors with voltages ranging from 5 to 35V and peak currents of up to 2A. The advantage of employing an H-bridge is that the motors can have their own power supply. This is especially important while using an Arduino board, since the 5V power supply is simply insufficient for two DC motors.

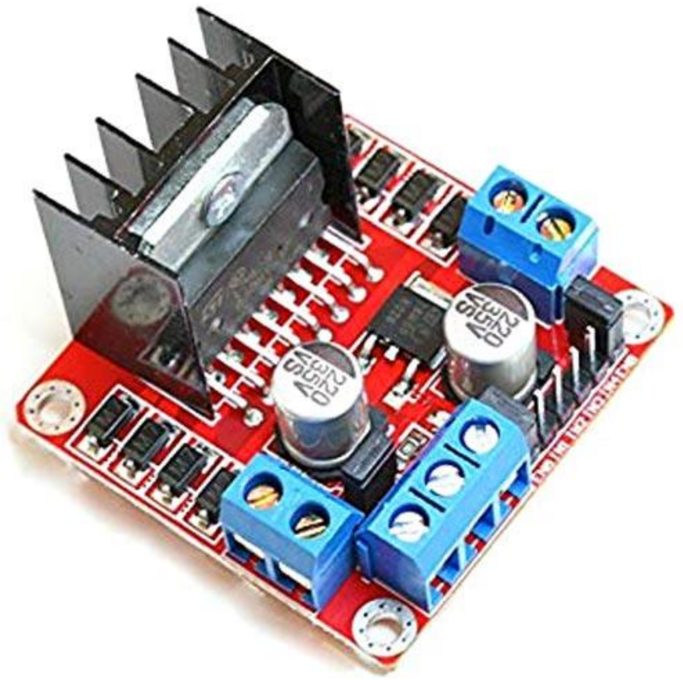


Figure 3-14: L298N Motor Driver

3.5.3 DC Motor with Encoder

An encoder is an electromechanical device that provides an electrical signal that is used for speed or position control[33]. Encoders convert mechanical activity into an electrical impulse that the control system uses to monitor certain application parameters and make adjustments as needed to keep the machine running smoothly In DC motors, where an actuator or rotor with coiled wires revolves inside a magnetic field formed by a stator, DC motor encoders are needed for speed control feedback. The DC motor encoder is a device that measures the rotor's speed and gives closed-loop feedback to the drive, allowing for precise speed control. This study involves double link robot manipulator, hence the use of two DC Motor with encoder due to the different control to different link. Therefore, two DC motor with different specifications was used to support different weight of the link. The DC motor used are 12V 21.92RPM 10kgfcm Planetary DC Geared Motor with Encoder for link 1 and for link 2 is 12V 150RPM 1.8kgfcm Brushed DC Geared Motor with Encoder as shown in Figure 3.15 and Figure 3.16.

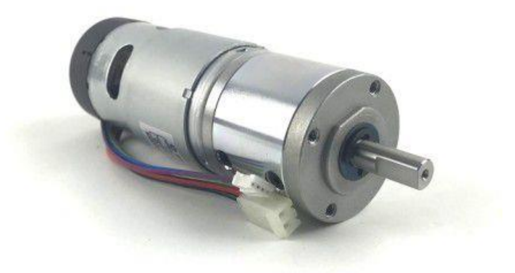


Figure 3-15: 12V 21.92RPM 10kgfcm Planetary DC Geared Motor with Encoder

Figure 3-16: 12V 150RPM 1.8kgfcm Brushed DC Geared Motor with Encoder

Each motor has different value of PPR. Hence, the encoder value needs to be calculated using the equation below.

$$Encoder\ value = \frac{360}{PPR\ \times quadrature\ encoder} \qquad 3.1$$

Motor	Encoder Value
Motor 1	0.75
Motor 2	0.4286

Table 3-2: Encoder Value

3.5.4 Circuit Diagram

In this subtopic, circuit diagram for hardware implementation will be presented. From figure below shows the connection of DC motor, Motor Driver, power supply and Arduino Uno. The use of two Arduino Uno is due to interrupt pin limitation which in each Arduino Uno only have 2 interrupt pins. Given that there are two motor that has been use and each motor use 2 interrupt pin, hence two Arduino was utilize.

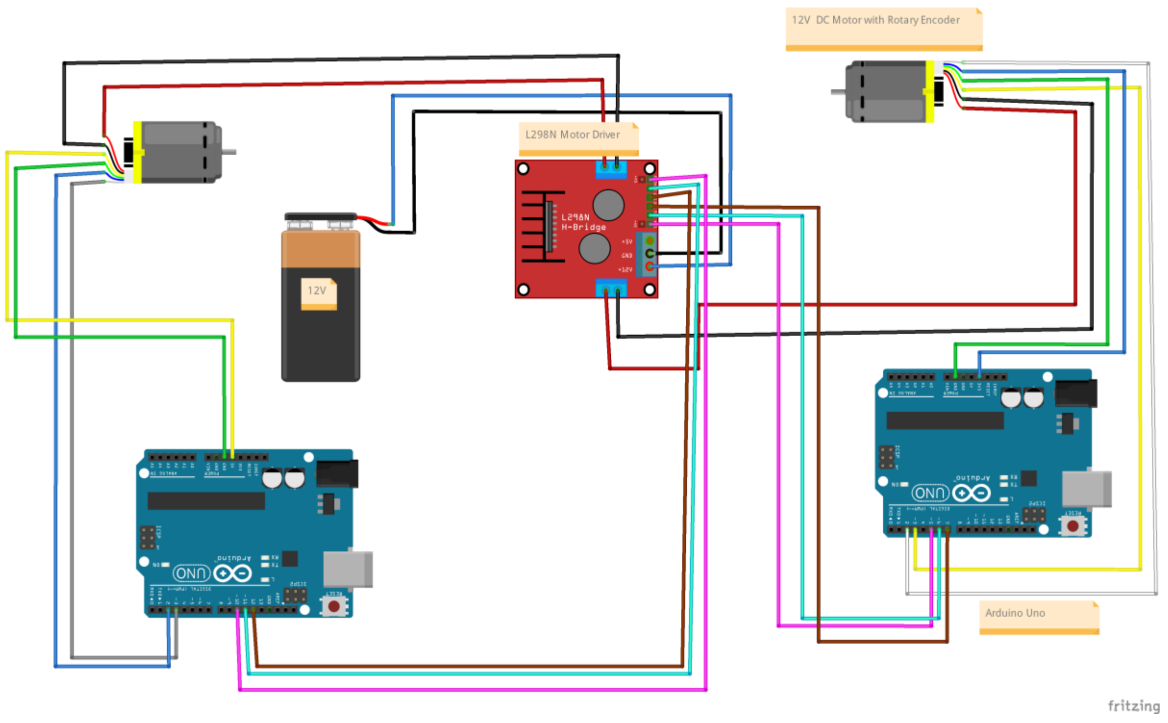


Figure 3-17: Circuit Diagram

3.5.5 Design

This design was developed in Solidwork before implemented in prototype. The joint for link 1 and link 2 was created by using 3D printing. The material used is ABS which is commonly utilised in a range of industrial and commercial applications due to its durability, high heat and wear resistance, and cost - effective. ABS is also regarded to be more durable and stronger than PLA, another popular thermoplastic, despite it is a little more difficult to work with.

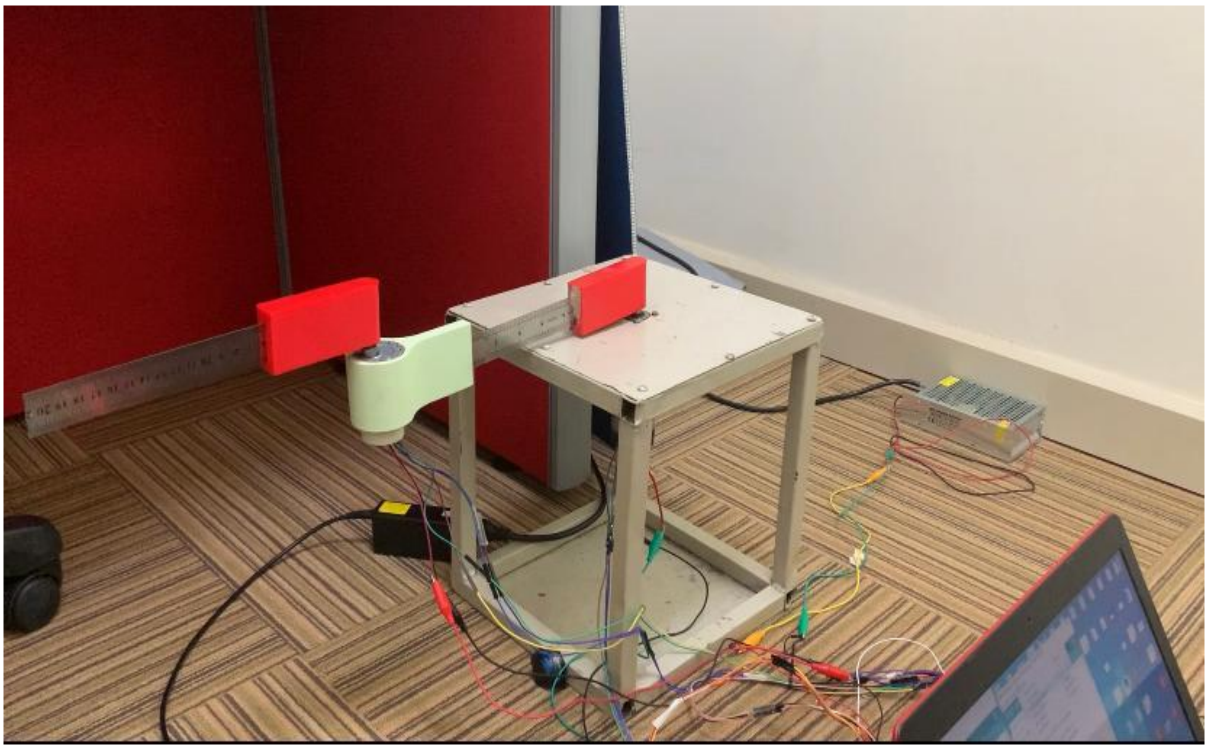
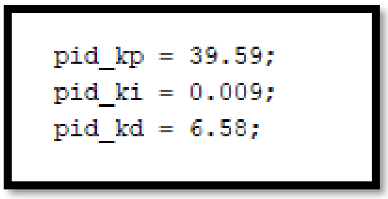


Figure 3-18: Design model

3.5.6 Controller

3.5.6.1 PID Controller

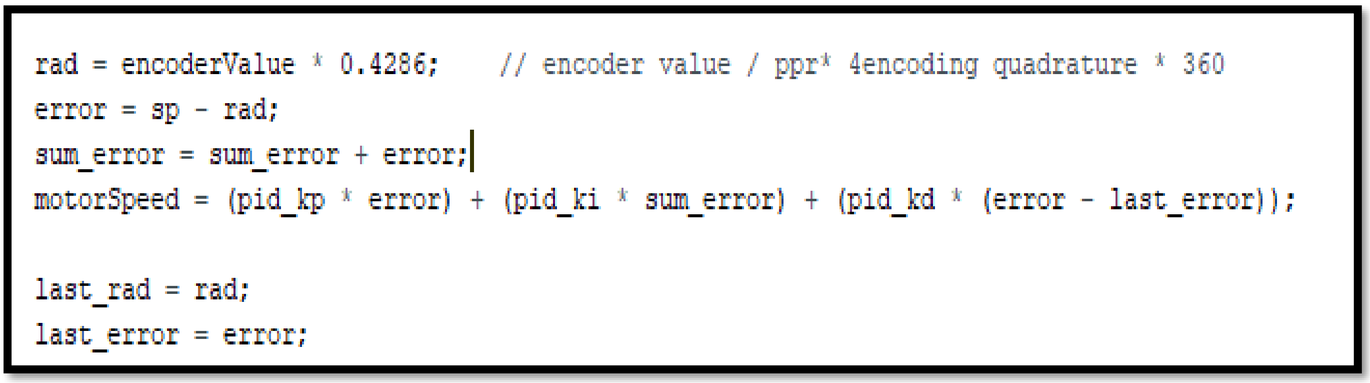
Figure 3.19 shows the value of PID that had been used in the system. The value is extracted from the simulation data. However, the value will not be exactly the same as the simulation due to the filter coefficient in the simulation. Hence, the value that needed to use for the system is based on try and error method until the system is stable to get the desired trajectory.



```
pid_kp = 39.59;
pid_ki = 0.009;
pid_kd = 6.58;
```

Figure 3-19: PID value

Based on Figure 3.20, the theoretical of PID was implemented in the system programming. For motor speed code, Kp, Ki and Kd were implemented. Kp is the error, Ki is sum of error and Kd is change of error.



```
rad = encoderValue * 0.4286;    // encoder value / ppr* 4encoding quadrature * 360
error = sp - rad;
sum_error = sum_error + error;
motorSpeed = (pid_kp * error) + (pid_ki * sum_error) + (pid_kd * (error - last_error));

last_rad = rad;
last_error = error;
```

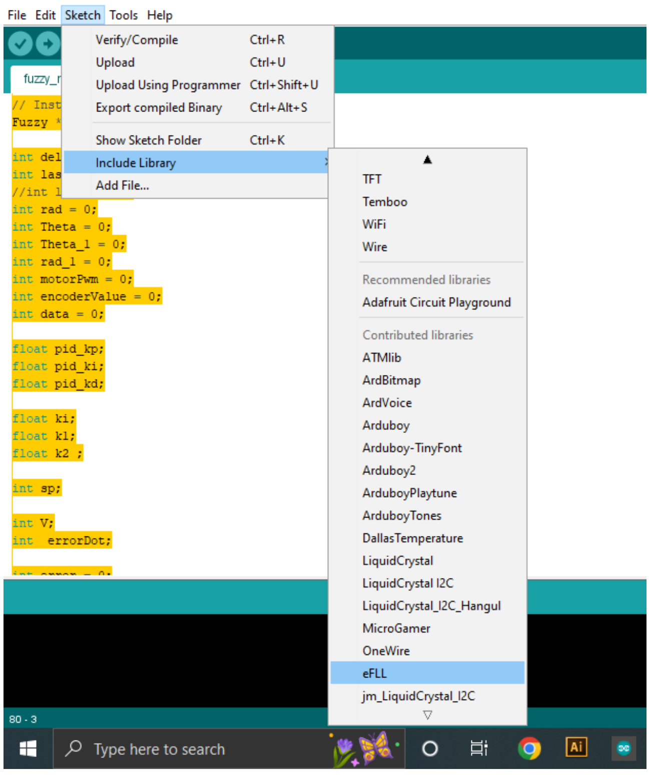
Figure 3-20: PID theoritical

3.5.6.2 Fuzzy Logic Controller

The implementation of fuzzy logic to prototype using Arduino, its it crucial to make sure that the Arduino software has the fuzzy library. Hence, to enable the fuzzy library, the eFLL library need to be included in the Arduino to successfully run the fuzzy.h file as in Figure 3.22. Table 3.21 lists the fuzzy controller rule base where, NB is negative big, NS is negative small, Z is zero, PS is positive small and PB is positive big.

Input	Output
NB	NB
NS	NS
Z	Z
PS	PS
PB	PB

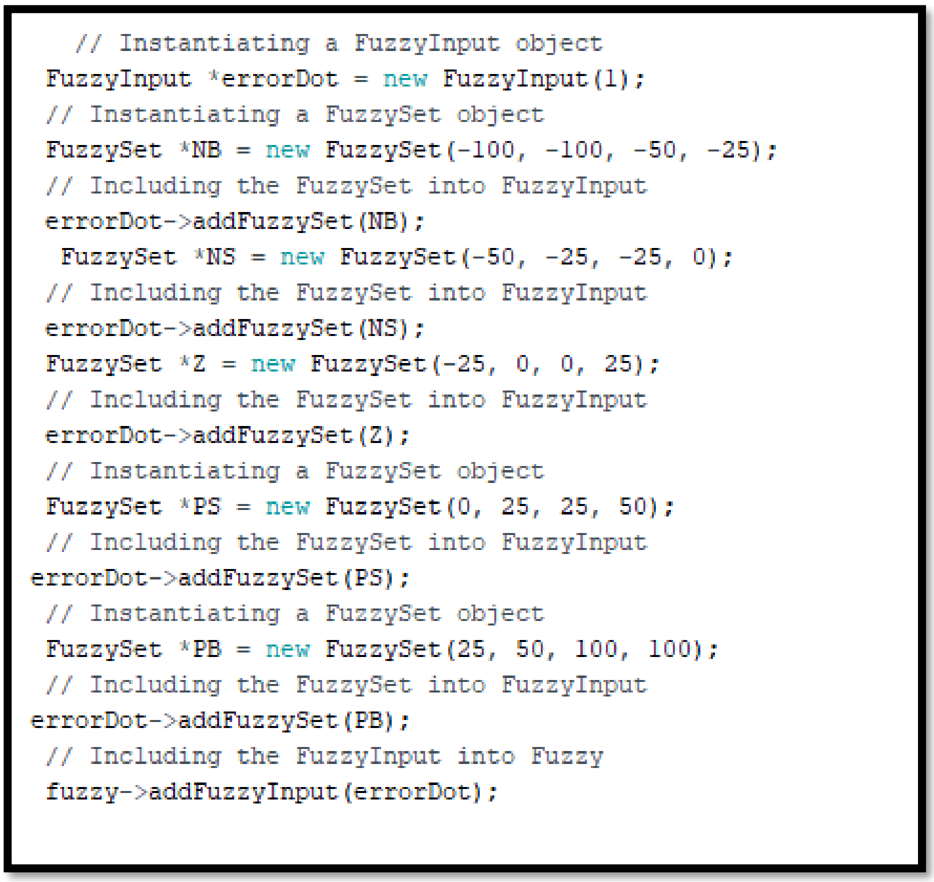
Figure 3-21: Rules notation



```
// Inst
Fuzzy *

int del
int las
//int l
int rad = 0;
int Theta = 0;
int Theta_1 = 0;
int rad_1 = 0;
int motorPwm = 0;
int encoderValue = 0;
int data = 0;

float pid_kp;
float pid_ki;
float pid_kd;

float ki;
float k1;
float k2 ;

int sp;

int V;
int  errorDot;
```

Figure 3-22: Enabling fuzzy library

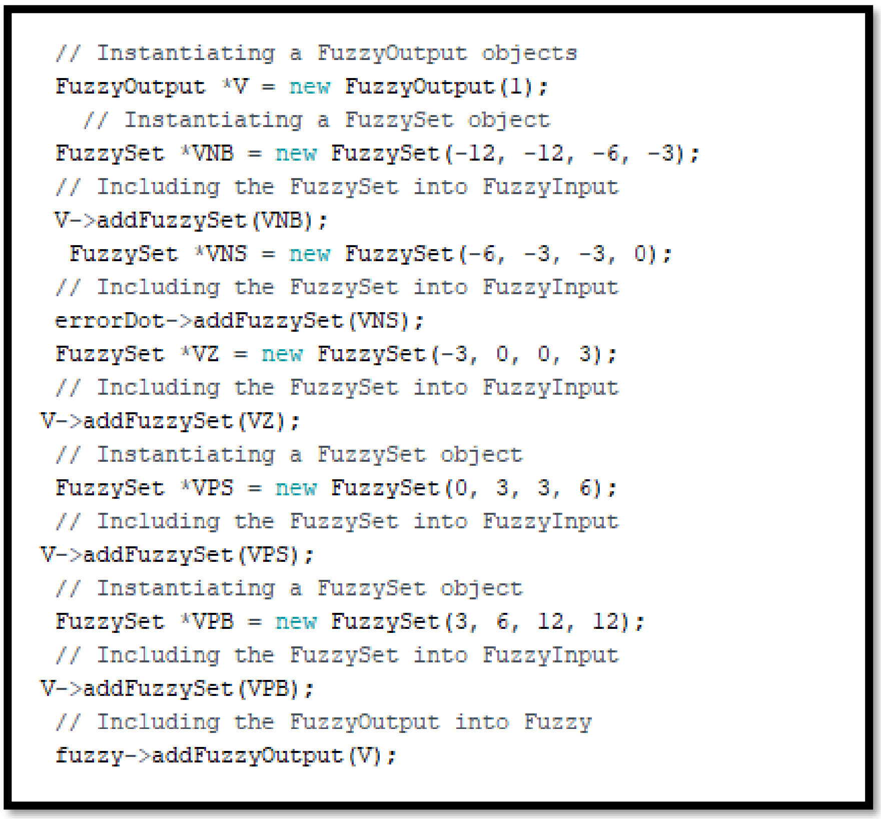
Figure 3.23 shows the input for membership function that has been implemented in Arduino. The range for the membership function is -100 and 100. This is due to error value of the system that expected to reach the range.



```
  // Instantiating a FuzzyInput object
 FuzzyInput *errorDot = new FuzzyInput(1);
 // Instantiating a FuzzySet object
 FuzzySet *NB = new FuzzySet(-100, -100, -50, -25);
 // Including the FuzzySet into FuzzyInput
 errorDot->addFuzzySet(NB);
  FuzzySet *NS = new FuzzySet(-50, -25, -25, 0);
 // Including the FuzzySet into FuzzyInput
 errorDot->addFuzzySet(NS);
 FuzzySet *Z = new FuzzySet(-25, 0, 0, 25);
 // Including the FuzzySet into FuzzyInput
 errorDot->addFuzzySet(Z);
 // Instantiating a FuzzySet object
 FuzzySet *PS = new FuzzySet(0, 25, 25, 50);
 // Including the FuzzySet into FuzzyInput
errorDot->addFuzzySet(PS);
 // Instantiating a FuzzySet object
 FuzzySet *PB = new FuzzySet(25, 50, 100, 100);
 // Including the FuzzySet into FuzzyInput
errorDot->addFuzzySet(PB);
 // Including the FuzzyInput into Fuzzy
 fuzzy->addFuzzyInput(errorDot);
```

Figure 3-23: Input membership function

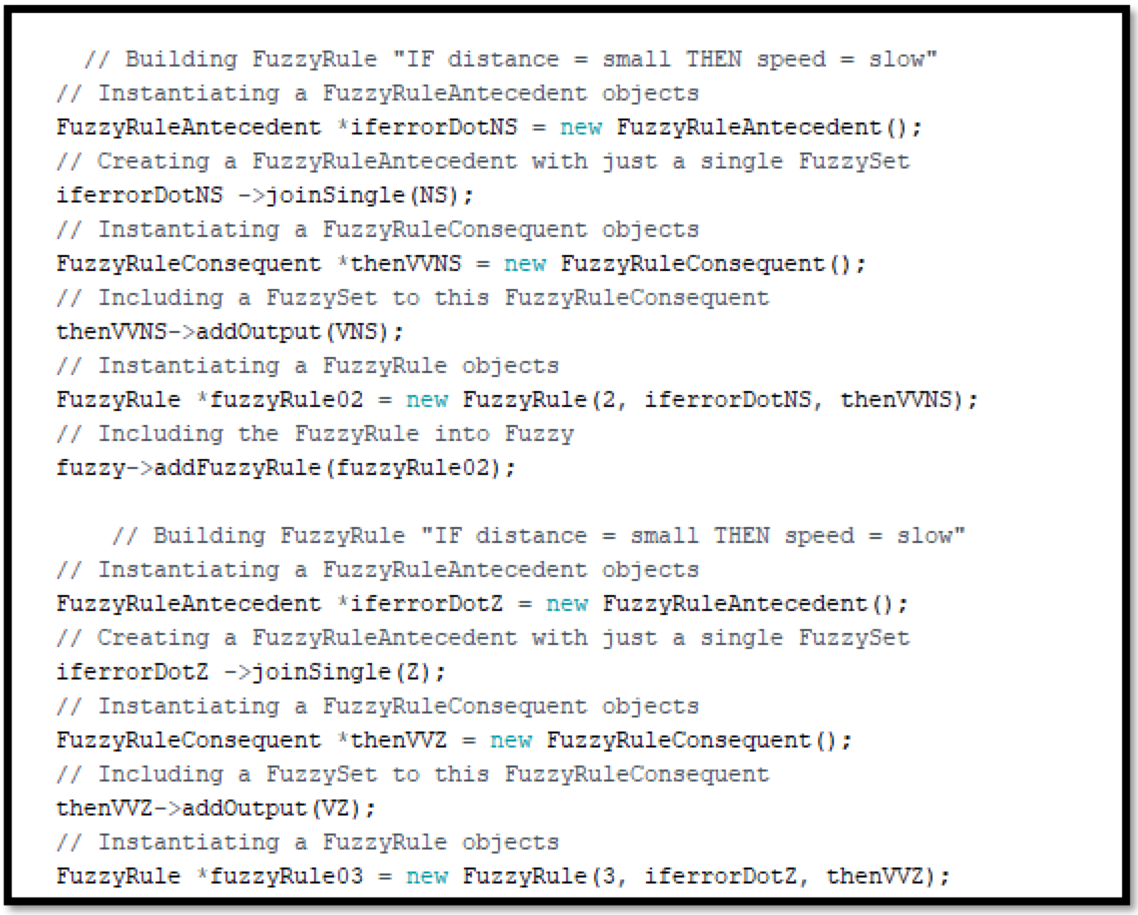
Figure 3.24 shows the input for membership function that has been implemented in Arduino. The range for the membership function is -12 and 12. This is due to output value of voltage in the system.



```
 // Instantiating a FuzzyOutput objects
 FuzzyOutput *V = new FuzzyOutput(1);
   // Instantiating a FuzzySet object
 FuzzySet *VNB = new FuzzySet(-12, -12, -6, -3);
 // Including the FuzzySet into FuzzyInput
 V->addFuzzySet(VNB);
  FuzzySet *VNS = new FuzzySet(-6, -3, -3, 0);
 // Including the FuzzySet into FuzzyInput
 errorDot->addFuzzySet(VNS);
 FuzzySet *VZ = new FuzzySet(-3, 0, 0, 3);
 // Including the FuzzySet into FuzzyInput
V->addFuzzySet(VZ);
 // Instantiating a FuzzySet object
 FuzzySet *VPS = new FuzzySet(0, 3, 3, 6);
 // Including the FuzzySet into FuzzyInput
V->addFuzzySet(VPS);
 // Instantiating a FuzzySet object
 FuzzySet *VPB = new FuzzySet(3, 6, 12, 12);
 // Including the FuzzySet into FuzzyInput
V->addFuzzySet(VPB);
 // Including the FuzzyOutput into Fuzzy
 fuzzy->addFuzzyOutput(V);
```

Figure 3-24: Output membership function

Figure 3.25 below shows the rules for fuzzy logic that are applied to arduino. The rules will act as a decider of the system.



```
  // Building FuzzyRule "IF distance = small THEN speed = slow"
// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *iferrorDotNS = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
iferrorDotNS ->joinSingle(NS);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenVVNS = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenVVNS->addOutput(VNS);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule02 = new FuzzyRule(2, iferrorDotNS, thenVVNS);
// Including the FuzzyRule into Fuzzy
fuzzy->addFuzzyRule(fuzzyRule02);

    // Building FuzzyRule "IF distance = small THEN speed = slow"
// Instantiating a FuzzyRuleAntecedent objects
FuzzyRuleAntecedent *iferrorDotZ = new FuzzyRuleAntecedent();
// Creating a FuzzyRuleAntecedent with just a single FuzzySet
iferrorDotZ ->joinSingle(Z);
// Instantiating a FuzzyRuleConsequent objects
FuzzyRuleConsequent *thenVVZ = new FuzzyRuleConsequent();
// Including a FuzzySet to this FuzzyRuleConsequent
thenVVZ->addOutput(VZ);
// Instantiating a FuzzyRule objects
FuzzyRule *fuzzyRule03 = new FuzzyRule(3, iferrorDotZ, thenVVZ);
```

Figure 3-25: Rules in program

3.6 Flowchart

3.6.1 Simulation Flowchart

Figure 3.26 is the project process in achieving the objectives. The project started with designing the two-link flexible manipulator in the Solidworks software. The model will be evaluated to ensure the model are as desired and can move perfectly. Then, the modelwill be exported to the Simulink and using the Simscape toolbox. The controller was added using the desired controller as in this system is PID controller and FLC. The tuning will takeplace to get the output that follows the trajectory that have been fed as an input. The system must follow the trajectory with minimal vibration.

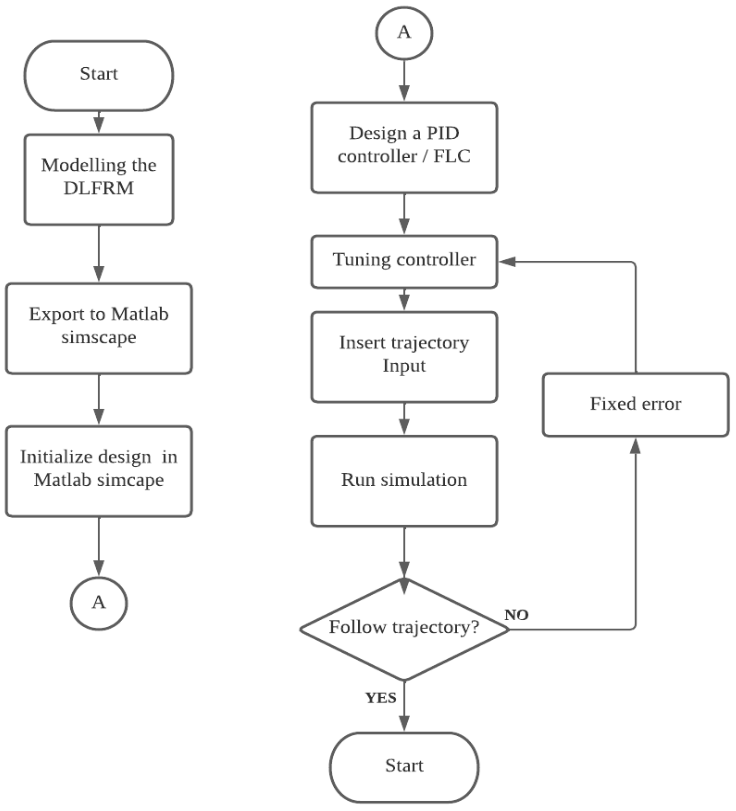


Figure 3-26: Simulation Flowchart

3.6.2 Hardware Flowchart

Figure below shows the workflow for the system for hardware implementation. The system starts with the initialization of variable, timer and the PWM. Next, the setpoint and the direction of DC motor are inserted into the system. The system will read and calculate the error of the set point. Then, the system will go into the controller chosen to apply the PWM duty cycle. The motor will rotate and move the link of robot manipulator. After that, serial plotter will display and measure the position of set point.

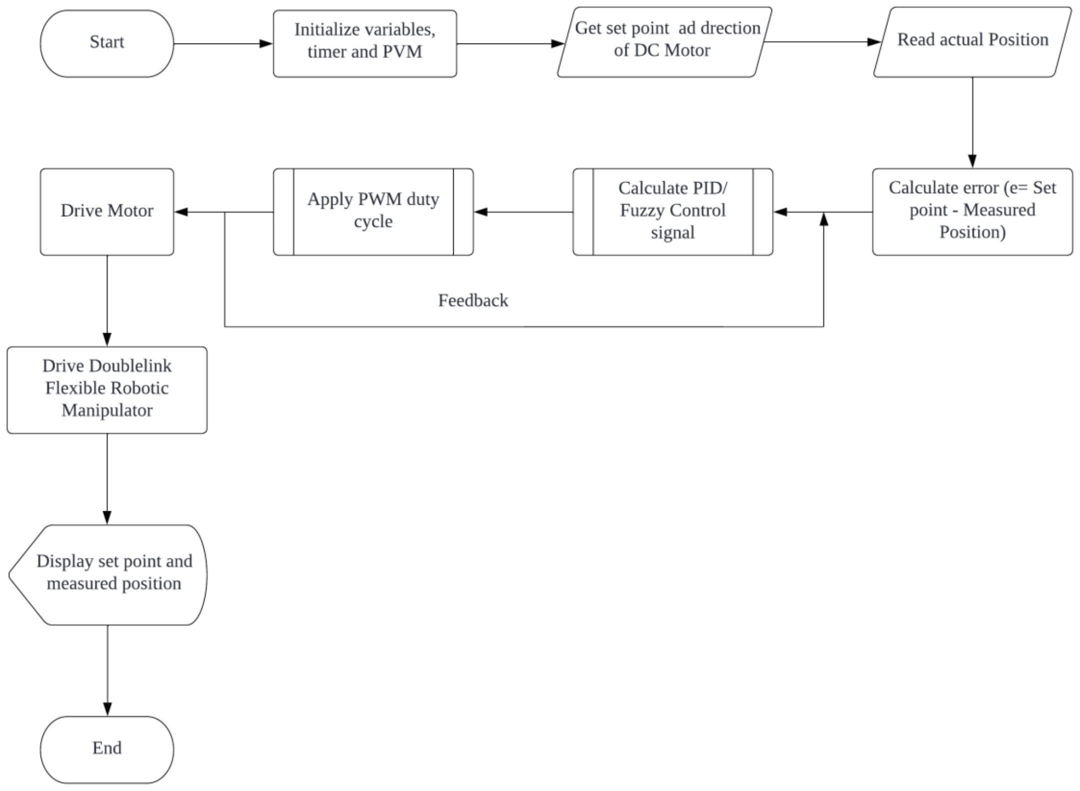

Start
Initialize variables, timer and PVM
Get set point ad drection of DC Motor
Read actual Position
Calculate error (e= Set point - Measured Position)
Calculate PID/ Fuzzy Control signal
Apply PWM duty cycle
Drive Motor
Feedback
Drive Doublelink Flexible Robotic Manipulator
Display set point and measured position
End

Figure 3-27: Hardware flowchart

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

The data, analysis, and individual findings obtained during the development of double link flexible manipulator are presented in this chapter. The simulation is conducted with a variety of controllers. PID and Fuzzy Controllers were developed, and simulations were presented in order to compare the controller's performance. The output for hardware also will be discussed in this chapter. The projected outcome of this investigation is stated in light of the project's goal.

4.2 Simulation Result

4.2.1 PID Controller

4.2.1.1 Without Disturbance

In previous paper research, different controllers can produce different output for improving the robot manipulator by link. In this analysis, the performance of link 1 and link 2 will be presented in term of overshoot, rise time, settle time and steady state error.

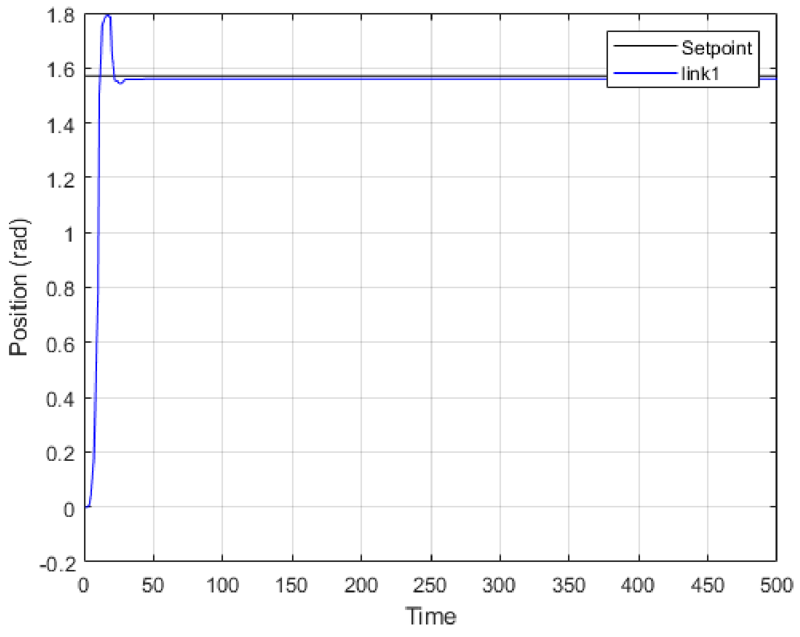


Figure 4-1: Link 1 without disturbance

Based on Figure 4.1, the step response for link 1 was presented. The data for this simulation was exported from block diagram simulation for PID. The graph shows that the system has a rise time and an overshoot before the steady state. After t=21, link 1 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-1: PID Link 1

Response Parameter	**Value**
Overshoot (%)	14.07
Rise Time (s)	0.1655
Settling Time (s)	0.8450
Steady State Error	0.011

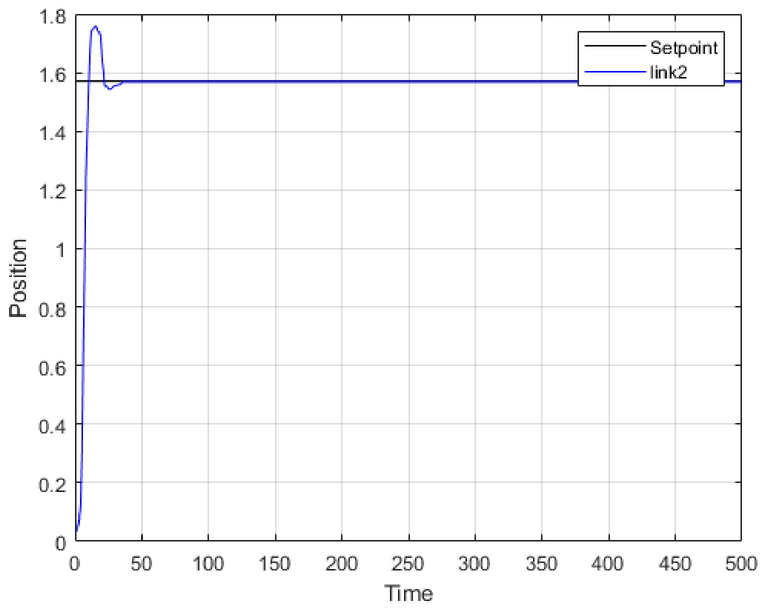


Figure 4-2: Link 2 without disturbance

Based on Figure 4.2, the step response for link 2 was presented. The graph shows that the system has a rise time and an overshoot before the steady state. After t=21, link 2 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-2: PID Link 2

Response Parameter	**Value**
Overshoot (%)	11.9817
Rise Time (s)	0.1562
Settling Time (s)	0.8455
Steady State Error	0.0018

Based on the results, the overshoot for link 1 is higher than link 2 which is 14.0702% and 11.9817% respectively. The steady state error for link 1 is 0.011 and for link 2 is 0.0018. The setting time and rise time for link 1 is 0.8450 s and 0.1655 s. However, for link 2 the settling time and rise time is 0.8455 s and 0.1562 s.

4.2.1.2 Disturbance

In this subchapter, the PID controller was added disturbance signal to see the performance of the system. The output should still follow the trajectory even though with the presence of the disturbance. The result can be referred to the graph below:

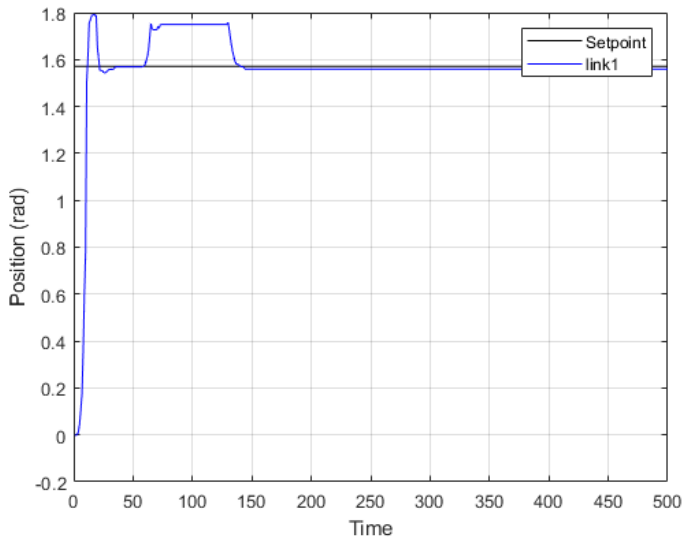


Figure 4-3: Link 1 with disturbance

Based on Figure 4.3, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-3: Response Parameter link 1 with disturbance

Response Parameter	**Value**
Overshoot (%)	14.0702
Rise Time (s)	0.1655
Settling Time (s)	5.4123
Steady State Error	0.0110

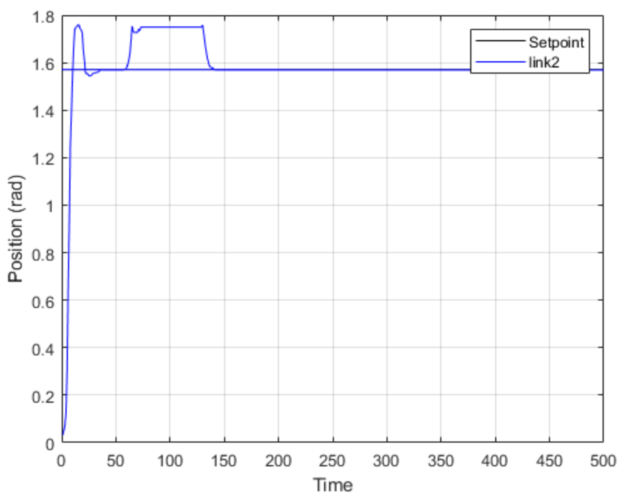


Figure 4-4: Link 2 with disturbance

Based on Figure 4.4, the step response for link 2 was presented. The graph shows that the system has a rise time, overshoot and disturbance before the steady state. After disturbance, link 2 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-4: Response Parameter link 2 with disturbance

Response Parameter	**Value**
Overshoot (%)	11.9817
Rise Time (s)	0.1562
Settling Time (s)	5.4139
Steady State Error	- 0.0005

Based on the results, the overshoot for link 1 is higher than link 2 which is 14.0702% and 11.9817% respectively. The steady state error for link 1 is 0.0110 and for link 2 is -0.0005. The setting time and rise time for link 1 is 5.4123 s and 0.16550 s. However, for link 2 the settling time and rise time is 5.4139 s and 0.1562 s. Settling time and steady state error are different from without disturbance because of the disturbance that have been applied to the system.

4.2.2 Fuzzy Logic Controller

4.2.2.1 Without Disturbance

FLC was implemented to this system to compare to PID controller. This is because FLC has become more commonly used in robot manipulator to overcome the vibration and overshoot. In this analysis, link 1 and link 2 was tested using FLC and the performance will be recorded. The result can be found in the figure below.

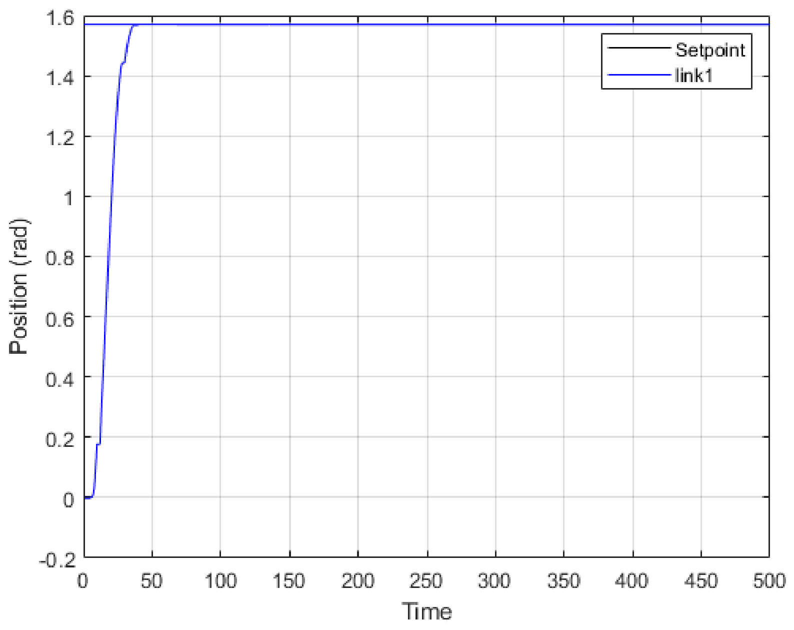


Figure 4-5: FLC without disturbance link 1

Based on figure 4.5, The system follows the trajectory more smoothly with no overshoot. Hence by using FLC the system successfully overcome the overshoot issue on robot manipulator. The detail can be found in table below.

Table 4-5: Response Parameter FLC without disturbance link 1

Response Parameter	**Value**
Overshoot (%)	0
Rise Time (s)	0.6863
Settling Time (s)	1.35034
Steady State Error	- 0.005

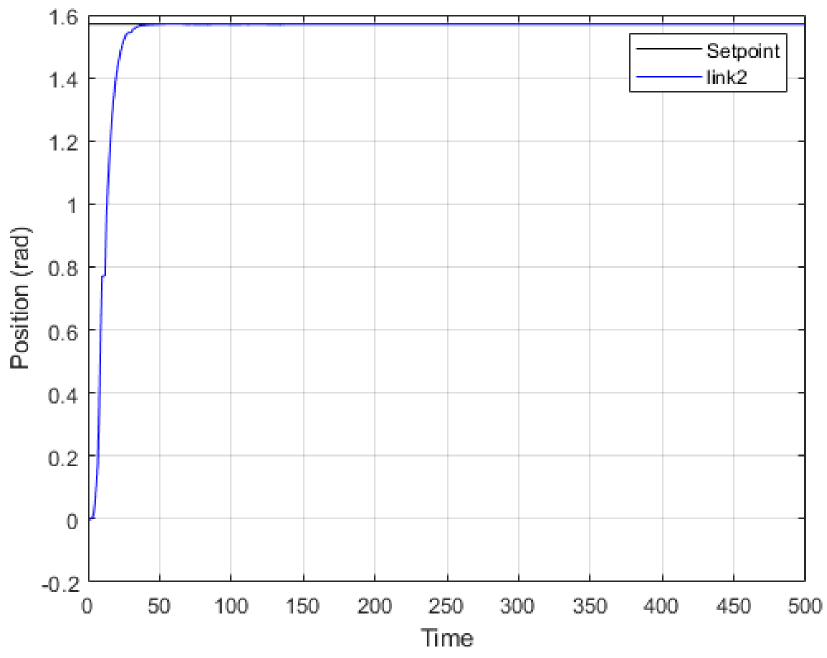


Figure 4-6: FLC without disturbance link 2

Based on figure 4.6, The system follows the trajectory more smoothly with no overshoot. Hence by using FLC the system successfully overcome the overshoot issue on robot manipulator. The detail can be found in table below.

Table 4-6: Response Parameter FLC without disturbance link 2

Response Parameter	**Value**
Overshoot (%)	0
Rise Time (s)	0.53619
Settling Time (s)	1.07994
Steady State Error	-0.0005

Based on the results, the overshoot for link 1 is higher than link 2 which is 0.6863% and 0.53619% respectively. The steady state error for link 1 is -0.005 and for link 2 is -0.0005. The settling time and rise time for link 1 is 1.3503 s and 0.6863 s. However, for link 2 the settling time and rise time is 1.0799 s and 0.5361 s.

4.2.2.2 Disturbance

In this subchapter, the FLC controller was added disturbance signal to see the performance of the system. The output should still follow the trajectory even though with the presence of the disturbance. The result can be referred to the graph below:

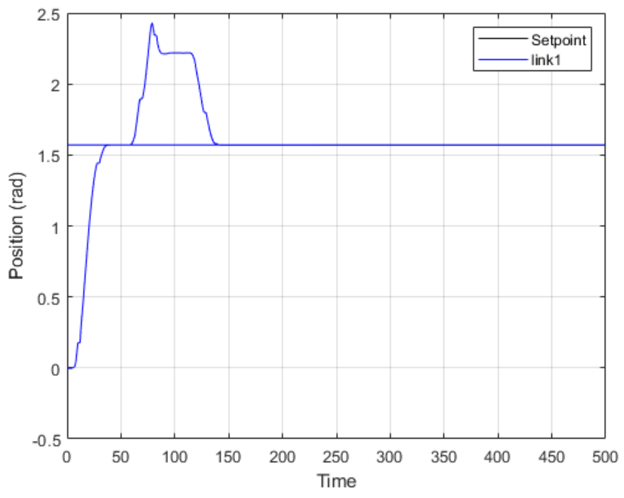


Figure 4-7: Link 1 FLC with disturbance

Based on Figure 4.7, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-7: Response Parameter FLC with disturbance link 1

Response Parameter	Value
Overshoot (%)	0
Rise Time (s)	0.6863
Settling Time (s)	5.4123
Steady State Error	-0.0005

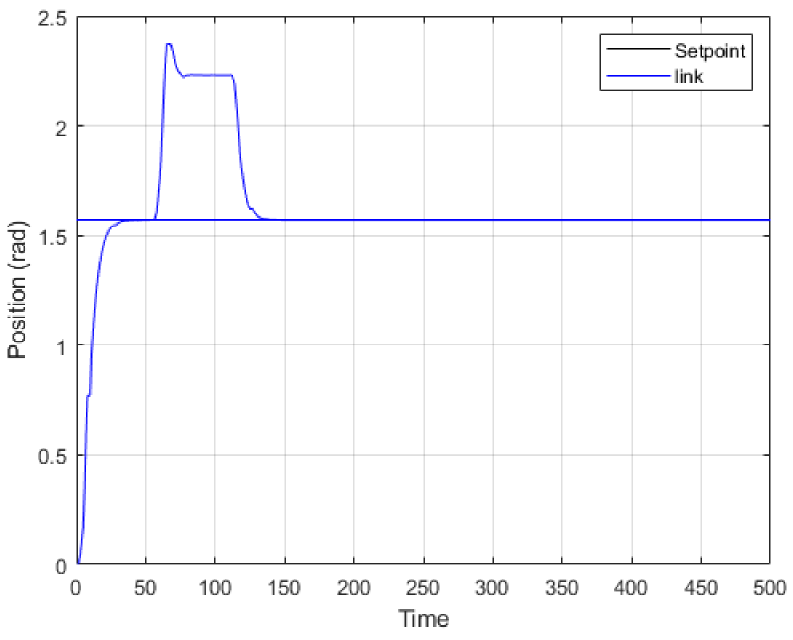


Figure 4-8: Link 2 FLC with disturbance

Based on Figure 4.8, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-8:Response Parameter FLC with disturbance link 2

Response Parameter	Value
Overshoot (%)	0
Rise Time (s)	0.53619
Settling Time (s)	5.1485
Steady State Error	-0.005

Based on the results, the overshoot for link 1 is higher than link 2 which is 0.6863% and 0.53619% respectively. The steady state error for link 1 is -0.005 and for link 2 is -0.0005. The settling time and rise time for link 1 is 5.4123 s and 0.6863 s. However, for link 2 the settling time and rise time is 5.1485 s and 0.53619 s.

4.3 Hardware Prototype Result

4.3.1 PID controller

4.3.1.1 Without Disturbance

PID controller has been implemented to the prototype to evaluate the performance of the system. The performance will be evaluated the same as the simulation which is in term of overshoot, rise time, settling time and steady state error. The result obtained will be presented below.

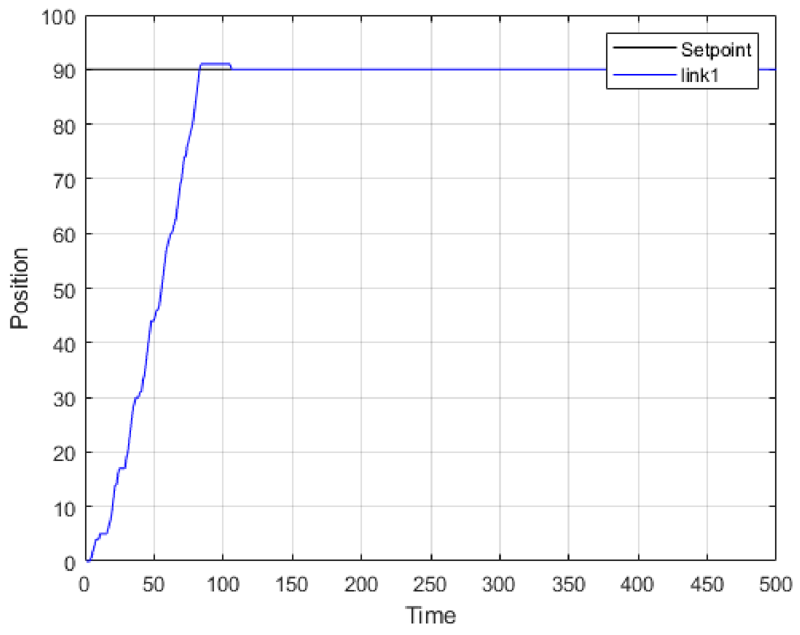


Figure 4-9: link 1 PID without disturbance

Based on Figure 4.9, the step response for link 1 was presented. The graph shows that the system has a rise time and an overshoot before the steady state. After t=3.28, link 1 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-9: Response Parameter PID without disturbance link 1

Response Parameter	Value
Overshoot (%)	1.1111
Rise Time (s)	2.36
Settling Time (s)	3.28
Steady State Error	0

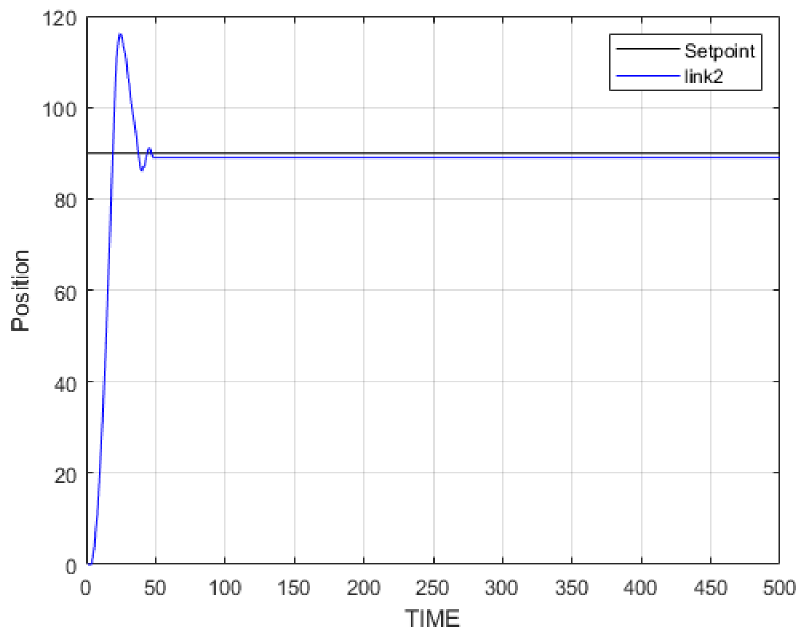


Figure 4-10: Link 2 PID without disturbance

Based on Figure 4.10, the step response for link 2 was presented. The graph shows that the system has a rise time and an overshoot before the steady state. After t=42, link 2 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-10: Response Parameter PID without disturbance link 2

Response Parameter	**Value**
Overshoot (%)	28.8889
Rise Time (s)	0.4311
Settling Time (s)	1.704
Steady State Error	1

Based on the results, the overshoot for link 2 is higher than link 1 which is 28.8889% and 1.1111% respectively. The steady state error for link 1 is 0 and for link 2 is 1. The setting time and rise time for link 1 is 3.28 s and 2.36 s. However, for link 2 the settling time and rise time is 1.704 s and 0.4311 s.

4.3.1.2 Disturbance

The disturbance signal was added to the PID controller to assess the system's performance. Despite the presence of the disturbance, the output should still follow the trajectory. The following graph represents the outcome:

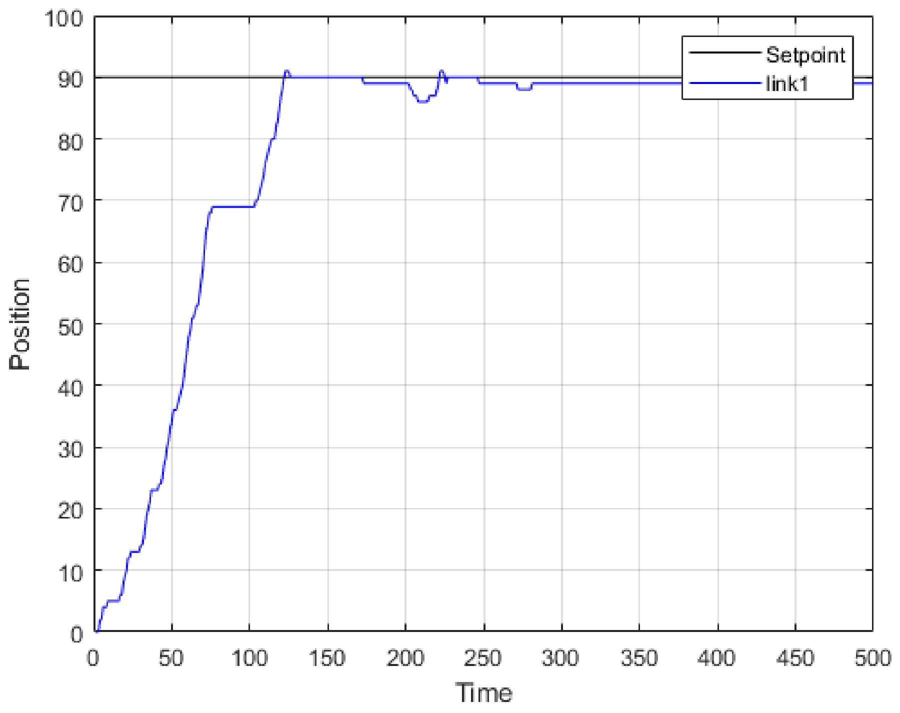


Figure 4-11: Link 1 PID with disturbance

Based on Figure 4.11, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-11: Response Parameter PID with disturbance link 1

Response Parameter	**Value**
Overshoot (%)	1.1111
Rise Time (s)	3.86
Settling Time (s)	11.208
Steady State Error	1

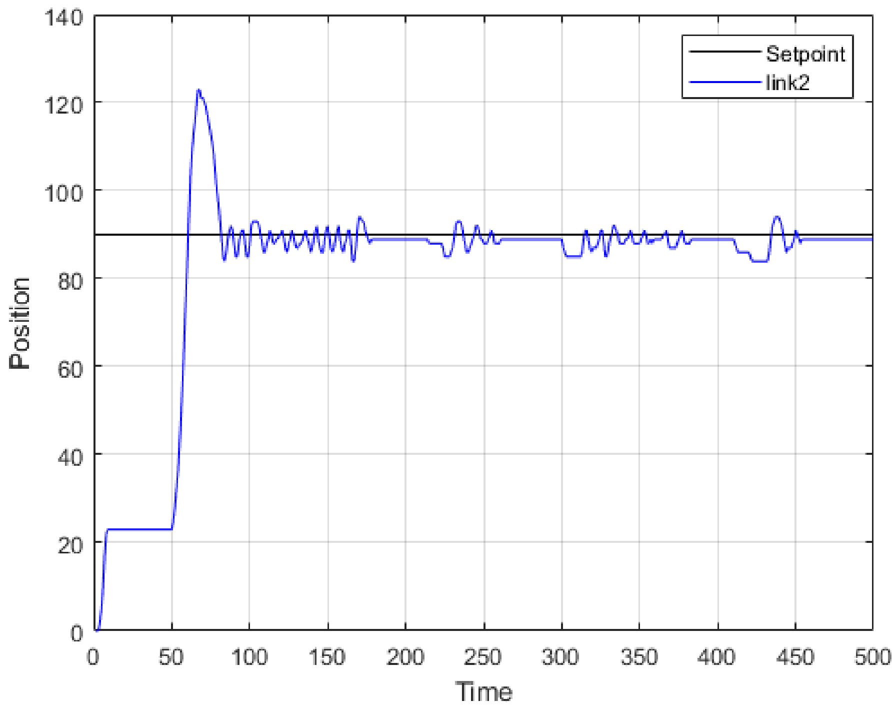


Figure 4-12: Link 2 PID with disturbance

Based on Figure 4.12, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-12: Response Parameter PID with disturbance link 2

Response Parameter	**Value**
Overshoot (%)	36.6667
Rise Time (s)	2.1547
Settling Time (s)	18.128
Steady State Error	1

Based on the results, the overshoot for link 2 is higher than link 1 which is 36.6667% and 1.1111% respectively. The steady state error for link 1 is 1 and for link 2 is 1. The setting time and rise time for link 1 is 11.208 s and 3.86 s. However, for link 2 the settling time and rise time is 18.128 s and 2.1547 s. Settling time and steady state error are different from without disturbance because of the disturbance that have been applied to the system.

4.3.2 Fuzzy Logic Controller

4.3.2.1 Without disturbance

FLC controller has been implemented to the prototype to evaluate the performance of the system. The performance will be evaluated the same as the simulation which is in term of overshoot, rise time, settling time and steady state error. The result obtained will be presented below.

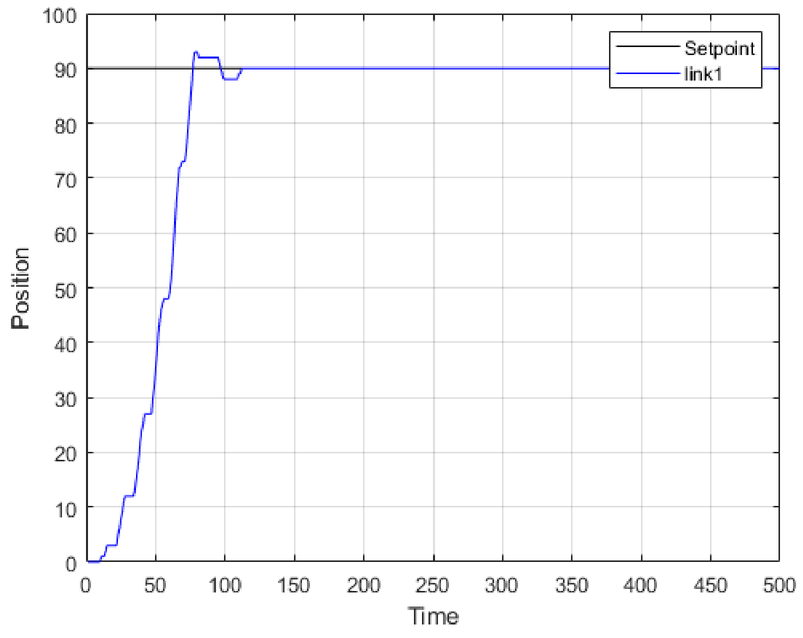


Figure 4-13: Link 1 FLC without disturbance

Based on Figure 4.13, the step response for link 1 was presented. The graph shows that the system has a rise time and an overshoot before the steady state. After t=109, link 1 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-13: Response Parameter FLC without disturbance link 1

Response Parameter	Value
Overshoot (%)	3.3333
Rise Time (s)	1.92
Settling Time (s)	4.368
Steady State Error	0

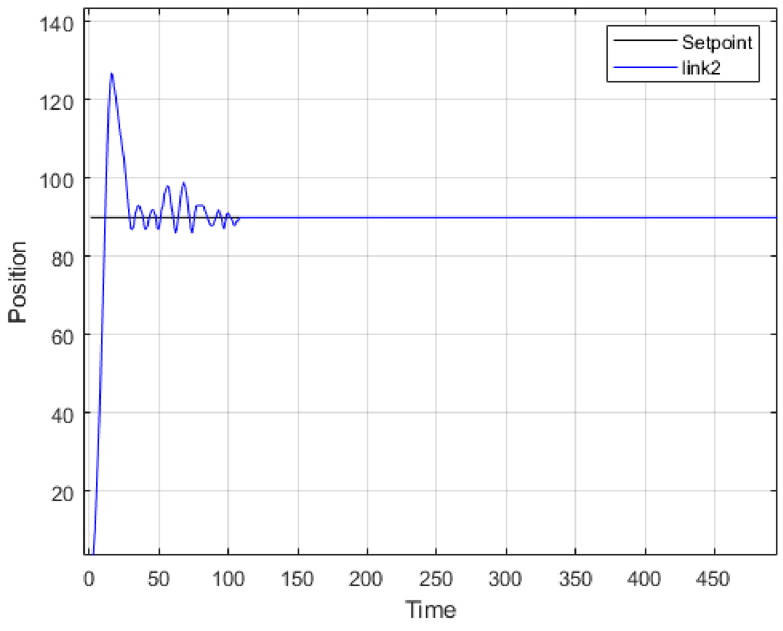


Figure 4-14: Link 2 FLC without disturbance

Based on Figure 4.14, the step response for link 1 was presented. The graph shows that the system has a rise time and an overshoot before the steady state. After t=105, link 1 successfully follows the setpoint that have been configured. The response parameter findings are as table below.

Table 4-14 : Response Parameter FLC without disturbance link 2

Response Parameter	**Value**
Overshoot (%)	41.11
Rise Time (s)	0.278
Settling Time (s)	4.208
Steady State Error	0

Based on the results, the overshoot for link 2 is higher than link 1 which is 41.11% and 3.3333% respectively. The steady state error for link 1 is 0 and for link 2 is 0. The setting time and rise time for link 1 is 4.368 s and 1.92 s. However, for link 2 the settling time and rise time is 4.208 s and 0.278 s.

4.3.2.2 Disturbance

In this subchapter, the FLC controller was added disturbance signal to see the performance of the system. The output should still follow the trajectory even though with the presence of the disturbance. The result can be referred to the graph below:

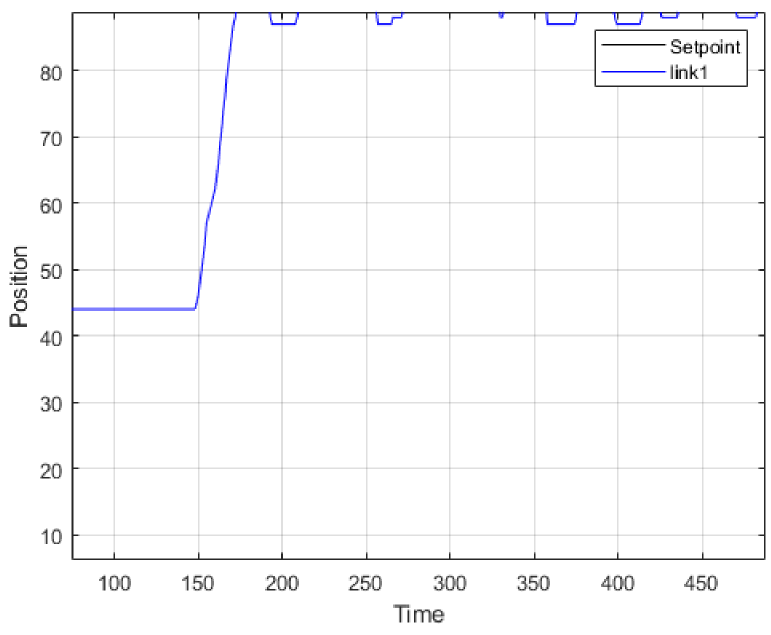


Figure 4-15: Link 1 FLC with disturbance

Based on Figure 4.15, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-15: Response Parameter FLC with disturbance link 1

Response Parameter	Value
Overshoot (%)	5.5556
Rise Time (s)	5.9248
Settling Time (s)	19.2888
Steady State Error	0

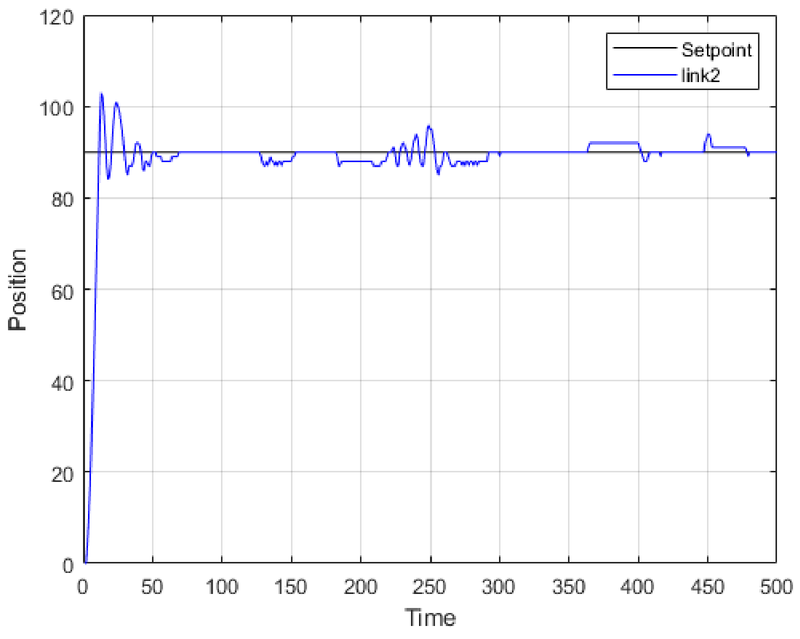


Figure 4-16: Link 2 FLC with disturbance

Based on Figure 4.16, the system follows the trajectory after disturbance was applied. Hence the system is successfully executed. The details of the finding can be found below.

Table 4-16: Response Parameter FLC with disturbance link 2

Response Parameter	Value
Overshoot (%)	14.4444
Rise Time (s)	0.28
Settling Time (s)	18.104
Steady State Error	0

Based on the results, the overshoot for link 2 is higher than link 1 which is 14.4444% and 5.5556% respectively. The steady state error for link 1 is 0 and for link 2 is 0. The settling time and rise time for link 1 is 19.2888 s and 5.9248 s. However, for link 2 the settling time and rise time is 18.104 s and 0.28 s.

4.4 Comparison between PID and FLC

This subchapter will be presenting about the comparison between PID and FLC. The comparison will consist of overshoot, settling time, rise time and steady state error.

Table 4-17: Comparison PID and FLC

Response Parameter	**Simulation**				**Hardware**			
	Link1		**Link2**		**Link1**		**Link2**	
	PID	**FLC**	**PID**	**FLC**	**PID**	**FLC**	**PID**	**FLC**
Overshoot (%)	14.07	0	11.9817	0	1.1111	3.3333	28.8889	41.11
Rise Time (s)	0.1655	0.6863	0.1562	0.5361	2.36	1.92	0.4311	0.278
Settling Time (s)	0.8450	1.3503	0.8455	1.0799	3.28	4.368	1.704	4.208
Steady State Error	0.011	-0.005	0.0018	-0.0005	0	0	1	0

Based on Table 4.17, the comparison of PID and FLC was presented for different controller. From the finding for simulation part, in term of overshoot and steady state error, FLC outperform PID. However, in term of rise time and settling time, PID has a better performance. Comparison in link 1 and link 2 in PID, the overshoot is higher on link 1 because of the link 1 need to support the weight of both link 1 and link 2. Consequently, the rise time of link 1 higher than link 2.

In hardware part, in term of overshoot and steady state error, PID outperform FLC. This is due to different rules applied to the system which in hardware using single input and single output. In term of rise time and settling time, PID has a better performance.

CHAPTER 5

CONCLUSION

5.1 Introduction

This chapter will conclude the finding of this project. The limitation on this study will be stated and some recommendations for future study for related projects are also included.

5.2 Conclusion

The advantages on using Solidworks to design the double link flexible manipulator instead of design directly in the Simulink is to design the complex model efficiently and shorten the time to design the model. Hence, the results obtained are matched to the expected results. From the results shows that the implemented of the controller shows that the system followed the trajectory but due to vibration that occurred, the system shows overshoot at the trajectory lines. The result of PID controller has been compared to fuzzy logic as a controller in DLFRM. Next, the benefits of a fuzzy-based controller over a PID controller may be concluded from the data. The fuzzy controller is anticipated to provide better control outcome, robustness, and overall performance. Fuzzy controllers are more stable, have less overshoot, and respond faster. However, each of controller has its own advantages and the disadvantages. The following parameters may be seen in the findings. The standard PID controller could not be utilized to regulate non-linear processes with complex design, according to the findings. From the result that obtained from hardware, the results from the simulation can be a reference to the hardware. The results are almost the same to each other system. As a result, in simulation, PID has better performance for both links in reducing settling time and rise time which is 0.5053s, 0.5308s for link1 and 0.2344s, 0.3799s for link 2 respectively. However, FLC outperforms PID controllers for both links in reducing overshoot and steady state error which is 14.07%, 0.005 for link1 and 11.9817%, 0.0005 for link2 respectively. As for hardware results, PID shows better performance in reducing overshoot and settling time for both links which is 2.2222%, 1.088s for link1 and 12.2211%, 2.504s for link2 respectively. On the other hand, FLC outperform PID controller for both links in reducing rise time and steady state error which is 0.44s, 0 for link1 and 0.1531s,1 for link2 respectively.

5.2.1 Limitations

There are several limitations during the study. The limitations are as below:

i. The simulation results cannot be compared to hardware results due to different resources. The simulations are using Matlab Simulink while hardware using Arduino codes.

ii. In Arduino, for fuzzy logic implementation, there has been an issue to obtain the output as in Arduino accept single input for membership functions.

5.2.2 Future recommendation

This work will feature about the potential of improving the trajectory for control of Double Link Flexible Robotic Manipulator and to be able to compare between simulation and hardware. To overcome the issues, the simulation and hardware must use the same system such as developing the hardware in Matlab and transfer to the Arduino. For tuning PID on simulation, instead of using autotune, it is recommended to use optimization to get the exact value of P, I and D for the system. For FLC, adding more variables into membership functions to make the system more details.

REFERENCES

[1] S. Hussein and M. Saleh, “A Fuzzy Logic Controllerfora Two-Link Functional Manipulator,” *Int. J. Comput. Networks Commun.*, vol. 6, no. 6, pp. 109–118, Nov. 2014, doi: 10.5121/IJCNC.2014.6608.

[2] “What is a Robotic Manipulator? - Robots Done Right.” [Online]. Available: https://robotsdoneright.com/Articles/what-is-a-robotic-manipulator.html. [Accessed: 04-Feb-2022].

[3] “Two-Link Flexible Manipulator Control Using Sliding Mode Control Based Linear Matrix Inequality,” doi: 10.1088/1742-6596/755/1/011001.

[4] J. S. Milind, G. Arunkumar, and T. C. Manjunath, “PID Control of a Double Link (2-link) Flexible Robotic Manipulator (2-DOF) in the 3 DE Space,” *2018 4th Int. Conf. Converg. Technol. I2CT 2018*, pp. 1–7, 2018, doi: 10.1109/I2CT42659.2018.9058215.

[5] I. H. Akyüz, Z. Bingül, and S. Kizir, “Cascade fuzzy logic control of a single-link flexible-joint manipulator,” *Turkish J. Electr. Eng. Comput. Sci.*, vol. 20, no. 5, pp. 713–726, 2012, doi: 10.3906/elk-1101-1056.

[6] D. Shang, X. Li, M. Yin, and F. Li, “Dynamic modeling and fuzzy compensation sliding mode control for flexible manipulator servo system,” *Appl. Math. Model.*, vol. 107, pp. 530–556, Jul. 2022, doi: 10.1016/J.APM.2022.02.035.

[7] J. Annisa, I. Z. M. Darus, M. O. Tokhi, and A. S. Z. Abidin, “Controlling the non-parametric modeling of Double Link Flexible Robotic Manipulator using Hybrid PID tuned by P-Type ILA,” *Int. J. Integr. Eng.*, vol. 10, no. 7, pp. 219–232, 2018, doi: 10.30880/ijie.2018.10.07.020.

[8] B. Karanayil and M. F. Rahman, “Artificial Neural Network Applications in Power Electronics and Electric Drives,” *Power Electron. Handb.*, pp. 1245–1260, 2018, doi: 10.1016/B978-0-12-811407-0.00041-6.

[9] A. Meystel, “Intelligent Control,” *Encycl. Phys. Sci. Technol.*, pp. 1–24, 2003, doi: 10.1016/B0-12-227410-5/00348-3.

[10] S. Bhagwan, J. S. Soni, and A. Kumar, “A Review on : PID Controller,” *Ijrmee*, vol. 3, no. February, p. 17, 2016.

[11] “(PDF) Improved Electrical Discharge Machine (EDM) Servomechanism Controller for Machining Micro Pits.” [Online]. Available: https://www.researchgate.net/publication/344378374_Improved_Electrical_Discharge_Machine_EDM_Servomechanism_Controller_for_Machining_Micro_Pits. [Accessed: 18-Jun-2022].

[12] A. Jamali, I. Z. Mat Darus, M. O. Tokhi, and A. S. Z. Abidin, “Utilizing P-Type ILA in tuning Hybrid PID Controller for Double Link Flexible Robotic Manipulator,” *2018 2nd Int. Conf. Smart Sensors Appl. ICSSA 2018*, pp. 141–146, 2018, doi: 10.1109/ICSSA.2018.8535973.

[13] M. Sazli and B. Saad, “EVOLUTIONARY OPTIMISATION AND REAL-TIME SELF-TUNING ACTIVE VIBRATION CONTROL OF A FLEXIBLE BEAM SYSTEM,” 2014.

[14] M. F. Aranza, J. Kustija, B. Trisno, and D. L. Hakim, “Tunning PID controller using particle swarm optimization algorithm on automatic voltage regulator system,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 128, no. 1, 2016, doi: 10.1088/1757-899X/128/1/012038.

[15] B. D. Argo, Y. Hendrawan, D. F. Al Riza, and A. N. Jaya Laksono, “Optimization of PID controller parameters on flow rate control system using multiple effect evaporator particle swarm optimization,” *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 5, no. 2, pp. 62–68, 2015, doi: 10.18517/IJASEIT.5.2.491.

[16] B. D. Argo, Y. Hendrawan, D. F. Al Riza, and A. N. Jaya Laksono, “Optimization of PID controller parameters on flow rate control system using multiple effect evaporator particle swarm optimization,” *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 5, no. 2, pp. 62–68, 2015, doi: 10.18517/ijaseit.5.2.491.

[17] M. Abachizadeh, M. R. H. Yazdi, and A. Yousefi-Koma, “Optimal tuning of PID controllers using Artificial Bee Colony algorithm,” *IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM*, pp. 379–384, 2010, doi: 10.1109/AIM.2010.5695861.

[18] R. KAYA and M. FURAT, “PID Ayarlama İçin Üç Kanallı Amaç Fonksiyonu Tabanlı Yapay Arı Kolonisi Algoritması,” *Eur. J. Sci. Technol.*, no. April, pp. 382–392, 2020, doi: 10.31590/ejosat.araconf50.

[19] M. Ahmadian, “ACTIVE CONTROL OF VEHICLE VIBRATION,” *Encycl. Vib.*, pp. 37–45, 2001, doi: 10.1006/RWVB.2001.0193.

[20] A. Kharidege, D. Jianbiao, and Y. Zhang, “Performance Study of PID and Fuzzy Controllers for Position Control of 6 DOF arm Manipulator with Various Defuzzification Strategies,” *MATEC Web Conf.*, vol. 77, pp. 3–8, 2016, doi: 10.1051/matecconf/20167701011.

[21] “(PDF) Fuzzy Assessment of Factors Influencing Quality Level of Highway Projects.” [Online]. Available: https://www.researchgate.net/publication/311451967_Fuzzy_Assessment_of_Factors_Influencing_Quality_Level_of_Highway_Projects#pf4. [Accessed: 18-Jun-2022].

[22] U. Kabir, M. F. Hamza, A. Haruna, and G. S. Shehu, “Performance analysis of PID, PD

and fuzzy controllers for position control of 3-DOF robot manipulator," *arXiv*, vol. 8, no. 1, pp. 18–25, 2019.

[23] J. Annisa, I. Z. Mat Darus, M. O. Tokhi, and S. Mohamaddan, "Implementation of PID Based Controller Tuned by Evolutionary Algorithm for Double Link Flexible Robotic Manipulator," *2018 Int. Conf. Comput. Approach Smart Syst. Des. Appl. ICASSDA 2018*, pp. 5–9, 2018, doi: 10.1109/ICASSDA.2018.8477615.

[24] L. Yanan, M. Deshan, L. Houde, W. Xueqian, and L. Bin, "Modeling and control of a two-link flexible space manipulator using the wave-based method," *Proc. 29th Chinese Control Decis. Conf. CCDC 2017*, vol. 0, pp. 512–519, 2017, doi: 10.1109/CCDC.2017.7978148.

[25] S. R. Vaishnav and Z. J. Khan, "Design considerations," *Cutting-Edge Technol. High. Educ.*, vol. 8, pp. 27–44, 2013, doi: 10.1108/S2044-9968(2013)0000008003.

[26] H. Kala, D. Deepakraj, P. Gopalakrishnan, P. Vengadesan, and M. Karumbal Iyyar, "Performance Evaluation of Fuzzy Logic and PID Controller for Liquid Level Process," *Int. J. Innov. Res. Electr.*, vol. 2, no. 3, pp. 2321–5526, 2014.

[27] R. Sam, K. Arrifin, and N. Buniyamin, "Simulation of pick and place robotics system using solidworks softmotion," *Proc. 2012 Int. Conf. Syst. Eng. Technol. ICSET 2012*, 2012, doi: 10.1109/ICSENGT.2012.6339325.

[28] "Control Tutorials for MATLAB and Simulink - Simulink Basics Tutorial." [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?aux=Basics_Simulink. [Accessed: 03-Feb-2022].

[29] "(PDF) Tuning of PID Controllers using Simulink."

[30] "Arduino Uno - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Arduino_Uno. [Accessed: 20-Jun-2022].

[31] "Choosing the right motor-driver | SP Robotic Works." [Online]. Available: https://sproboticworks.com/blog/choosing-the-right-motor-driver. [Accessed: 20-Jun-2022].

[32] "In-Depth: Interface L298N DC Motor Driver Module with Arduino." [Online]. Available: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/. [Accessed: 20-Jun-2022].

[33] "Motor Encoder Speed & Position Overview | Dynapar." [Online]. Available: https://www.dynapar.com/technology/encoder_basics/motor_encoders/. [Accessed: 20-Jun-2022].

APPENDIX A
SAMPLE APPENDIX 1

Arduino codes- PID controller

```
// Driver motor H-bridge Mosfet IRF 9540 & Mosfet IRF-540 > L298
#define PWM 5
#define IN1 6
#define IN2 7

// motor
#define HALLSEN_A 2
#define HALLSEN_B 3

int motorSpeed = 0;

int delta_rad = 0;
int last_rad = 0;
//int last_rad=0;
int rad = 0;
int Theta = 0;
int Theta_1 = 0;
int rad_1 = 0;
int motorPwm = 0;
int encoderValue = 0;
int data = 0;

float pid_kp;
float pid_ki;
float pid_kd;

float ki;
float k1;
float k2 ;

int sp;

int error = 0;
int last_error = 0;

int sum_error = 0;

bool signalA = false;
bool signalB = false;

void setup() {
  Serial.begin(9600);

  pinMode(PWM, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);

  pinMode(HALLSEN_A, INPUT_PULLUP);
  pinMode(HALLSEN_B, INPUT_PULLUP);
  encoderValue = 0;
  attachInterrupt(digitalPinToInterrupt(HALLSEN_A), updateEncoder,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(HALLSEN_B), updateEncoder2,
CHANGE);

  //    Serial.println("CLEARDATA");
  //    Serial.println("LABEL, CLOCK, rad, sp");

  sp = 180;

  //    ki = 0.06;
  //    k1 = 5;
  //    k2 = 5;

  pid_kp = 39.2.36;
  pid_ki = 0.009;
  pid_kd = 6.58;

  //  pid_kp = 5;
  //  pid_ki = 0.009;
  //  pid_kd = 0.2391;

  //    pid_kp = 149.95;
  //    pid_ki = 0.009;
  //    pid_kd = 1.506;

  //    pid_kp= 300;
  //    pid_ki= 0.0015;
  //    pid_kd= 0.0001;
}

void loop() {
  while (data < 1000)
  {

    //   rad = encoderValue * 1.46938;       // encoder value / ppr*
1encoding quadrature * 360
    //    rad = encoderValue * 0.73469;      // encoder value / ppr*
2encoding quadrature * 360
    //  rad = encoderValue * 0.36735;     // encoder value / ppr*
4encoding quadrature * 360
    //   rad = encoderValue * 0.18367;   // encoder value / ppr*
8encoding quadrature * 360

    //   rad = encoderValue * 1.46938;       // encoder value / ppr*
1encoding quadrature * 360
    //    rad = encoderValue * 0.73469;      // encoder value / ppr*
2encoding quadrature * 360
    rad = encoderValue * 0.4286;     // encoder value / ppr* 4encoding
quadrature * 360
    //   rad = encoderValue * 0.18367;   // encoder value / ppr*
8encoding quadrature * 360

    error = sp - rad;

    sum_error = sum_error + error;
    // delta_rad = rad-last_rad;
    // delta_error = error - last_error;
    // motorSpeed = (ki*sum_error) - ( (k1 * rad) + (k2 * delta_rad));
    motorSpeed = (pid_kp * error) + (pid_ki * sum_error) + (pid_kd *
(error - last_error));

    last_rad = rad;
    last_error = error;

    if (motorSpeed > 250) motorSpeed = 250;
    if (motorSpeed > 250) motorSpeed = -250;

    if (motorSpeed < 0) digitalWrite(IN1, LOW);
    else digitalWrite(IN1, HIGH);

    if (motorSpeed < 0) digitalWrite(IN2, HIGH);
    else digitalWrite(IN2, LOW);

    analogWrite(PWM, motorSpeed);

    //    Serial.print("DATA,TIME,");
    Serial.print(rad);
    Serial.print(",");
    Serial.print(sp);
    Serial.print(",");
    Serial.print(PWM);
    //    Serial.print(",");
    //    Serial.print(error);
    Serial.println();

    data++;
  }
  //  digitalWrite(IN1, LOW);
  //  digitalWrite(IN2, HIGH);
  analogWrite(PWM, 0);
}

void updateEncoder()
{
  signalA = digitalRead(HALLSEN_A);
  if (signalA != signalB) encoderValue++;
  else encoderValue--;
}

void updateEncoder2()
{
  signalB = digitalRead(HALLSEN_B);
  if (signalA == signalB) encoderValue++;
  else encoderValue--;
}
```

Arduino codes- fuzzy logic controller

```
// Driver motor H-bridge Mosfet IRF 9540 & Mosfet IRF-540 > L298
#define PWM 5
#define IN1 6
#define IN2 7
#include <Fuzzy.h>

// motor
#define HALLSEN_A 2
#define HALLSEN_B 3

int motorSpeed = 0;

// Instantiating a Fuzzy object
Fuzzy *fuzzy = new Fuzzy();

int delta_rad = 0;
int last_rad = 0;
//int last_rad=0;
int rad = 0;
int Theta = 0;
int Theta_1 = 0;
int rad_1 = 0;
int motorPwm = 0;
int encoderValue = 0;
int data = 0;

float pid_kp;
float pid_ki;
float pid_kd;

float ki;
float k1;
float k2 ;

int sp;

int V;
int  errorDot;

int error = 0;
int last_error = 0;

int sum_error = 0;

bool signalA = false;
bool signalB = false;

void setup() {
  Serial.begin(9600);
  Serial.println("CLEARSHEET");
  Serial.println("LABEL,CLOCK,Setpoint,Output");

  pinMode(PWM, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  pinMode(HALLSEN_A, INPUT_PULLUP);
  pinMode(HALLSEN_B, INPUT_PULLUP);

  encoderValue = 0;
  attachInterrupt(digitalPinToInterrupt(HALLSEN_A), updateEncoder,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(HALLSEN_B), updateEncoder2,
CHANGE);

     Serial.println("CLEARDATA");
    Serial.println("LABEL, CLOCK, rad, sp");

    // Instantiating a FuzzyInput object
  FuzzyInput *errorDot = new FuzzyInput(1);
  // Instantiating a FuzzySet object
  FuzzySet *NB = new FuzzySet(-100, -100, -50, -25);
  // Including the FuzzySet into FuzzyInput
  errorDot->addFuzzySet(NB);
   FuzzySet *NS = new FuzzySet(-50, -25, -25, 0);
  // Including the FuzzySet into FuzzyInput
  errorDot->addFuzzySet(NS);
  FuzzySet *Z = new FuzzySet(-25, 0, 0, 25);
  // Including the FuzzySet into FuzzyInput
  errorDot->addFuzzySet(Z);
  // Instantiating a FuzzySet object
  FuzzySet *PS = new FuzzySet(0, 25, 25, 50);
  // Including the FuzzySet into FuzzyInput
 errorDot->addFuzzySet(PS);
  // Instantiating a FuzzySet object
  FuzzySet *PB = new FuzzySet(25, 50, 100, 100);
  // Including the FuzzySet into FuzzyInput
 errorDot->addFuzzySet(PB);
  // Including the FuzzyInput into Fuzzy
  fuzzy->addFuzzyInput(errorDot);


  // Instantiating a FuzzyOutput objects
  FuzzyOutput *V = new FuzzyOutput(1);
    // Instantiating a FuzzySet object
  FuzzySet *VNB = new FuzzySet(-12, -12, -6, -3);
  // Including the FuzzySet into FuzzyInput
  V->addFuzzySet(VNB);
   FuzzySet *VNS = new FuzzySet(-6, -3, -3, 0);
  // Including the FuzzySet into FuzzyInput
  errorDot->addFuzzySet(VNS);
  FuzzySet *VZ = new FuzzySet(-3, 0, 0, 3);
  // Including the FuzzySet into FuzzyInput
 V->addFuzzySet(VZ);
  // Instantiating a FuzzySet object
  FuzzySet *VPS = new FuzzySet(0, 3, 3, 6);
  // Including the FuzzySet into FuzzyInput
 V->addFuzzySet(VPS);
  // Instantiating a FuzzySet object
  FuzzySet *VPB = new FuzzySet(3, 6, 12, 12);
  // Including the FuzzySet into FuzzyInput
 V->addFuzzySet(VPB);
  // Including the FuzzyOutput into Fuzzy
  fuzzy->addFuzzyOutput(V);

  // Building FuzzyRule "IF distance = small THEN speed = slow"
  // Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *iferrorDotNB = new FuzzyRuleAntecedent();
  // Creating a FuzzyRuleAntecedent with just a single FuzzySet
  iferrorDotNB ->joinSingle(NB);
  // Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenVVNB = new FuzzyRuleConsequent();
  // Including a FuzzySet to this FuzzyRuleConsequent
  thenVVNB->addOutput(VNB);
  // Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule01 = new FuzzyRule(1, iferrorDotNB, thenVVNB);
  // Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule01);


    // Building FuzzyRule "IF distance = small THEN speed = slow"
  // Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *iferrorDotNS = new FuzzyRuleAntecedent();
  // Creating a FuzzyRuleAntecedent with just a single FuzzySet
  iferrorDotNS ->joinSingle(NS);
  // Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenVVNS = new FuzzyRuleConsequent();
  // Including a FuzzySet to this FuzzyRuleConsequent
  thenVVNS->addOutput(VNS);
  // Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule02 = new FuzzyRule(2, iferrorDotNS, thenVVNS);
  // Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule02);

      // Building FuzzyRule "IF distance = small THEN speed = slow"
  // Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *iferrorDotZ = new FuzzyRuleAntecedent();
  // Creating a FuzzyRuleAntecedent with just a single FuzzySet
  iferrorDotZ ->joinSingle(Z);
  // Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenVVZ = new FuzzyRuleConsequent();
  // Including a FuzzySet to this FuzzyRuleConsequent
  thenVVZ->addOutput(VZ);
  // Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule03 = new FuzzyRule(3, iferrorDotZ, thenVVZ);
  // Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule03);

    // Building FuzzyRule "IF distance = small THEN speed = slow"
  // Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *iferrorDotPS = new FuzzyRuleAntecedent();
  // Creating a FuzzyRuleAntecedent with just a single FuzzySet
  iferrorDotPS ->joinSingle(PS);
  // Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenVVPS = new FuzzyRuleConsequent();
  // Including a FuzzySet to this FuzzyRuleConsequent
  thenVVPS->addOutput(VPS);
  // Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule04 = new FuzzyRule(4, iferrorDotPS, thenVVPS);
  // Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule04);


    // Building FuzzyRule "IF distance = small THEN speed = slow"
  // Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *iferrorDotPB = new FuzzyRuleAntecedent();
  // Creating a FuzzyRuleAntecedent with just a single FuzzySet
```

```
  iferrorDotPB ->joinSingle(PB);
  // Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenVVPB= new FuzzyRuleConsequent();
  // Including a FuzzySet to this FuzzyRuleConsequent
  thenVVPB->addOutput(VPB);
  // Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule05 = new FuzzyRule(5, iferrorDotPB, thenVVPB);
  // Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule05);

  sp = 90;

  //    ki = 0.06;
  //    k1 = 5;
  //    k2 = 5;

  pid_kp = 69.242.36;

  pid_kd = 0.2391;

}

void loop() {
  while (data < 500)
  {
  fuzzy -> setInput(1, errorDot);
  fuzzy->fuzzify();
  V = fuzzy -> defuzzify(1);

    //   rad = encoderValue * 1.46938;       // encoder value / ppr*
1encoding quadrature * 360
    //    rad = encoderValue * 0.73469;     // encoder value / ppr*
2encoding quadrature * 360
    //  rad = encoderValue * 0.36735;    // encoder value / ppr*
4encoding quadrature * 360
    //   rad = encoderValue * 0.18367;  // encoder value / ppr*
8encoding quadrature * 360

    //   rad = encoderValue * 1.46938;       // encoder value / ppr*
1encoding quadrature * 360
    //    rad = encoderValue * 0.73469;     // encoder value / ppr*
2encoding quadrature * 360
    rad = encoderValue * 0.423;    // encoder value / ppr* 4encoding
quadrature * 360
    //   rad = encoderValue * 0.18367;  // encoder value / ppr*
8encoding quadrature * 360

    errorDot = sp - rad;

    sum_error = sum_error +  errorDot;


  V = (pid_kp *  errorDot) + (pid_kd * ( errorDot - last_error));

    last_rad = rad;
    last_error =  errorDot;

    if (V > 255) V = 255;
    if (V > 255) V = -255;

    if (V < 0) digitalWrite(IN1, LOW);
    else digitalWrite(IN1, HIGH);

    if (V < 0) digitalWrite(IN2, HIGH);
    else digitalWrite(IN2, LOW);

    analogWrite(PWM,V);
//    //
    Serial.print("DATA,TIME,");
    Serial.print(sp);
    Serial.print(",");
    Serial.print(rad);
//    Serial.print(",");
//    Serial.print( V);
    Serial.println();

    data++;
  }
  //  digitalWrite(IN1, LOW);
  //  digitalWrite(IN2, HIGH);
  analogWrite(PWM, 0);
}

void updateEncoder()
{
  signalA = digitalRead(HALLSEN_A);
  if (signalA != signalB) encoderValue++;
  else encoderValue--;
}

void updateEncoder2()
{
  signalB = digitalRead(HALLSEN_B);
  if (signalA == signalB) encoderValue++;
  else encoderValue--;
}
```