Sudan University of Science and Technology
Collage of Postgraduate

Improving the Accuracy of Robotic Arm position using Fuzzy Logic Controller

تحسين دقة موقع ذراع آلي بإستخدام المتحكم المنطقي الغامض

A thesis Submitted for Partial Fulfillment for the Requirement of M.Sc. Degree in Mechatronics Engineering

Presented by:-

Azza Ali Mohamed Ali

Supervisor:-

Dr. Awadalla Taifour Ali

November- 2019

الآية

قال تعالى:

(وَلَقَدْ آتَيْنَا دَاوُودَ وَسُلَيْمَانَ عِلْمًا وَقَالَا الْحَمْدُ لِلَّهِ الَّذِي فَضَّلَنَا عَلَى كَثِيرٍ مِنْ عِبَادِهِ الْمُؤْمِنِينَ) النمل: [١٥]

صدق الله العظيم

Dedication

To the souls of the martyrs of the glorious December's Revolution, may Allah forgive and accept them and bless our country with peace and stability.

Acknowledgment

All praises to Allah for the strengths and His blessing in completing this thesis. This thesis has been kept on track and has been seen through to completion with the support and encouragement of numerous people.

Threshold I would like to express my deepest appreciation to all those who provided me with the possibility to complete this thesis starting with my advisor **Dr.AwadallaTaifour Ali** and special gratitude I give to**Eng. Ahmed YosifAbd-ALbarri** for all what he has done and the great effort that he exerted and **Eng: Musab** as well.

Last and not the least, I do like to express my sincere gratitude tomy source of inspiration, my source of strength and my strong pillars**my family** and **my friends** for their persistence presence firstly and fortheir love, support, understanding and good wishes whenever I needed throughout my life , so many thanks.

Abstract

Robots are the latest technologies used to facilitate agricultural, industrial and medical tasks for humans and to make their life more stable and to accomplish tasks with utmost accuracy. Therefore, scientific studies continue to improve and develop the structure of robots and control methods, and has always been the mainly goal is to achieve the desired accuracy and the safety of enterprises and workers.

Robots can be guided by an external control device or the control may be embedded within.

The aim of this thesis is to improve the accuracy of a robotic arm position. This robot works with five motors and variable resistors that control the movement of the arm and a position sensor which connected to the microcontroller Arduino and controlled by the Proportional Integral Differential controller (PID).

The control method is changed to fuzzy logic controller with the same robot's hardware components. After sensing the robotic arm current position using MPU5060 sensor, a fuzzy controller compared the current position with the desired value that adjusted with the potentiometer then detected the error and corrected it and gave the desired output and the robotic arm performance was improved.

المستخلص

الأذرع الآلية من أحدث التقنيات الحديثة المستخدمة لتسهيل المهام الزراعية ، الصناعية والطبية علي الإنسان ولجعل حياته أكثر إستقراراً ولإنجاز المهام بدقة متناهية. لذا تتوالي الدراسات العلمية في تحسين وتطوير هيكل الأذرع الآلية وطرق التحكم بها، ولطالما كان الهدف الأساسي الوصول للدقة المرجوة وتحقيق سلامة المنشآت والعاملين.

يمكن يمكن التحكم على الأذرع الآلية بواسطة جهاز تحكم خارجي أو قد يكون عنصر التحكم مضمنًا في الداخل.

يهدف هذا البحث لتحسين دقة ذراع آلي يعمل بخمسة محركات ومقاومات متغيرة تتحكم في حركة الذراع ومتحسس موقع الذراع متصل بالمتحكم الدقيق (أوردوينو) ويتحكم فيه بواسطة المتحكم التناسبي التفاضلي التكاملي. فتم تغيير طريقة التحكم إلي المتحكم المنطقي الغامض بنفس مكونات الروبوت الملموسة، فبعد استشعار الوضع الحالي للذراع بإستخدام المستشعر PMUS060 ، تمت مقارنة الموضع الحالي بالقيمة المطلوبة التي تم ضبطها بإستخدام المقاومات المتغيرة وتم إيجاد نسبة الخطأ وتصحيحها والحصول على النتيجة المطلوبة ووصل الذراع الآلي لدقة أعلي وتحسن أداءه.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

INTRODUCTION

This chapter shows a brief general introduction of the study, define the problem statements, the proposed solutions, the methodology and the research layout.

1.1 Introduction

Robots are mechanical or virtual artificial agents, typically guided by a computer program or electronic circuitry and robotics is an interdisciplinary branch of engineering and science that includes mechanical engineering, electronics engineering, computer science, and others. Robotics deals with the design, construction, operation, and use of robots, as well as computer systems for their control, sensory feedback, and information processing. As more and more robots are designed for specific tasks like Agricultural robots , domestic robots (cleaning and caring for the elderly) , medical robots (performing low-invasive surgery) , household robots (with full use) , nano robots,….etc[1].

Generally for robots, a controller is used to modify the behavior of the physical system according to the input value through computations and actuations. Over the early decade, numerous control techniques and methodologies have been proposed to control the motion of the robot manipulators such as point-to-point, sequencing "continuous path", speed and incremental motions. As an example, the first control method capable for stopping at several different programmed positions, it can be used to pick and place operations. The required control method is chosen depending on the type of the robot manipulator and its possible applications. Varity of robot manipulators and their architectures influence the control methodology, for example, to control the robot manipulator movement between two points x and y (point to point) needs a different controller than the continuous path tracking. On the other hand, the mechanical design of the manipulator affects the controller type.

Although robot manipulators have variety of tasks in all applications, it has limited behavior as compared with human. Therefore, the control technique must apply to achieve the desired behavior [1].

1.2 Problem Statement

Mainly robots are designed to help humans in many of their life aspects (e.g. Automotive industries, medical laboratories, agriculture, nuclear energy... etc) and to apply the required functions in a perfect way we should continue progressing the robot's designs and their controller and caring about the accuracy and the safety.

Here we have a pick and place robotic arm using PID controller which has a good performance in specific limitation for the position but PID controller is not sufficient to obtain the desired tracking control performance because of the nonlinearity of the robot manipulator.

1.3 Objectives

Design a controller which can overcome the PID missing in order to enhancement and increase the accuracy of the robotic arm is needed, so the main objectives of this thesis are to:-

1- Design and implement a fuzzy logic controller for a pick and place robotic arm.

1.4 Methodology

After sensing the robotic arm position using MPU5060 sensor a fuzzy logic controller is designed and applied to improve the accuracy of the pick and place robotic arm which programmed using Arduino then the results are going to set and compared with PID controller results.

1.5 Thesis Layout

This thesis has made up of five chapters as following:

Chapter One shows a brief general introduction of the study, define the problem statements, the proposed solutions, the methodology and the research layout. While Chapter Two shows the theoretical background before applying the practical part, which introduces information and more knowledge about the robot, the fuzzy fundamentals, control system and a view of the microcontroller followed with the previous studies.

Chapter Three gives a view about the used module , the connection and the used fuzzy rules for its programming followed with Chapter Four which shows the system implementation and its testing to figure out whether the goal is achived or no.

Eventually the results which compared with the previous work's results as a conclusion, and the recommendations for the forth coming researchers showed in Chapter Five.

CHAPTER TWO

THEORETICAL BACKGROUND AND LITERATURE REVIEW

CHAPTER TWO

THEORETICAL BACKGROUND AND LITERATURE REVIEW

Chapter Two shows the theoretical background before applying the practical part, which introduces information and more knowledge about the robot, the fuzzy fundamentals, control system and a view of the microcontroller followed with the previous studies.

2.1 Robot

The Robotic Institute of America defined the concept of robotic as a re-programmable multi-functional manipulator designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks, which also acquire information from the environment and move intelligently in response. While ISO 8373 defined it as an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications [1].

robots are designed for specific tasks like Agricultural robots , Domestic robots (cleaning and caring for the elderly) , Medical robots (performing low-invasive surgery) , Household robots (with full use) , Nano-robots,….etc [2]. also welding, painting, assembly, pick and place for printed circuit boards, packaging and labeling, palletizing, product inspection, and testing; all accomplished with high endurance, speed, and precision. They can help in material handling and provide interfaces [3].

2.1.1The robotic arms parameters

The robotic arm builds on the coming parameters:

- Number of axes: Two axes are required to reach any point in a plane; three axes are required to reach any point in space. To fully control the orientation of the end of the arm (i.e. the wrist) three more axes (yaw, pitch, and roll) are required. Some designs (e.g. the SCARA robot) trade limitations in motion possibilities for cost, speed, and accuracy.[4]
- Degrees of freedom : It is usually the same as the number of axes.

- Working envelope: Describes the region of space that a robot can reach.
- Kinematics : Is the actual arrangement of rigid members and joints in the robot, which determines the robot's possible motions. Classes of robot kinematics include articulated, Cartesian, parallel and SCARA.
- Carrying capacity or payload: Describes how much weight a robot can lift.
- Speed: Shows how fast the robot can position the end of its arm. This may be defined in terms of the angular or linear speed of each axis or as a compound speed i.e. the speed of the end of the arm when all axes are moving.[4]
- Acceleration: Describes how quickly an axis can accelerate. Since this is a limiting factor a robot may not be able to reach its specified maximum speed for movements over a short distance or a complex path requiring frequent changes of direction.
- Accuracy: Tells you how closely a robot can reach a commanded position. When the absolute position of the robot is measured and compared to the commanded position the error is a measure of accuracy. Accuracy can be improved with external sensing for example a vision system or Infra-Red. See robot calibration. Accuracy can vary with speed and position within the working envelope and with payload (see compliance).[4]
- Repeatability: Describes how well the robot will return to a programmed position. This is not the same as accuracy. It may be that when told to go to a certain X-Y-Z position that it gets only to within 1 mm of that position. This would be its accuracy which may be improved by calibration. But if that position is taught into controller memory and each time it is sent there it returns to within 0.1mm of the taught position then the repeatability will be within 0.1mm.
- Motion control: For some applications, such as simple pick-and-place assembly, the robot need merely return repeatedly to a limited number of pre-taught positions. For more sophisticated applications, such as welding and finishing (spray painting), motion must be continuously controlled to follow a path in space, with controlled orientation and velocity.[4]
- Power source: Some robots use electric motors, others use hydraulic actuators. The former are faster, the latter are stronger and advantageous in applications such as spray painting, where a spark could set off an explosion; however, low internal air-

pressurization of the arm can prevent ingress of flammable vapors as well as other contaminants.

- Drive: Some robots connect electric motors to the joints via gears; others connect the motor to the joint directly (direct drive). Using gears results in measurable 'backlash' which is free movement in an axis. Smaller robot arms frequently employ high speed, low torque DC motors, which generally require high gearing ratios; this has the disadvantage of backlash. In such cases the harmonic drive is often used.
- Compliance: This is a measure of the amount in angle or distance that a robot axis will move when a force is applied to it. Because of compliance when a robot goes to a position carrying its maximum payload it will be at a position slightly lower than when it is carrying no payload. Compliance can also be responsible for overshoot when carrying high payloads in which case acceleration would need to be reduced [4].

A- The robot's advantages and disadvantages

Any technological equipment has its advantages and dis advantages.

i- The robot's advantages:

- Increase productivity
- Use equipment effectively.
- Reduce working costs .
- Flexibility at work .
- Get the job done in the shortest time.
- Provide good returns on investment .
- Better accuracy in performance .
- Ability to work in risky ways and make it safer.

ii- The robot's disadvantages:

- Cause unemployment for manual workers .
- High initial cost .
- Designed Arm to perform specific tasks and not comparable to the human hand.
- Difficulty programmed to perform accurate tasks.
- Needed a large number of sensors and high accuracy to perform the Complex tasks.

- And other technical problems (especially in the fields of artificial intelligence and machine vision) [5].

2.1.2 Fuzzy fundamental

Fuzzy logic idea is similar to the human being's feeling and inference process. Unlike classical control strategy, which is a point-to-point control, fuzzy logic control is a range-to-point or range-to-range control. The output of a fuzzy controller is derived from fuzzifications of both inputs and outputs using the associated membership functions.

A crisp input will be converted to the different members of the associated membership functions based on its value. From this point of view, the output of a fuzzy logic controller is based on its memberships of the different membership functions, which can be considered as a range of inputs.

Fuzzy ideas and fuzzy logic are so often utilized in our routine life that nobody even pays attention to them. For instance, to answer some questions in certain surveys, most time one could answer with 'Not Very Satisfied' or 'Quite Satisfied', which are also fuzzy or ambiguous answers. Exactly to what degree is one satisfied or dissatisfied with some service or product for those surveys? These vague answers can only be created and implemented by human beings, but not machines. Is it possible for a computer to answer those survey questions directly as a human beings did? It is absolutely impossible. Computers can only understand either '0' or '1', and 'HIGH' or 'LOW'. Those data are called crisp or classic data and can be processed by all machines. Is it possible to allow computers to handle those ambiguous data with the help of a human being? If so, how can computers and machines handle those vague data? The answer to the first question is yes. But to answer the second question, we need some fuzzy logic techniques and knowledge of fuzzy inference system.[6]

The idea of fuzzy logic was invented by Professor L. A. Zadeh of the University of California at Berkeley in 1965. This invention was not well recognized until Dr. E. H. Mamdani, who is a professor at London University, applied the fuzzy logic in a practical application to control an automatic steam engine in 1974, which is almost ten years after the fuzzy theory was invented. Then, in 1976, Blue Circle Cement and

SIRA in Denmark developed an industrial application to control cement kilns . That system began to operation in 1982. More and more fuzzy implementations have been reported since the 1980s, including those applications in industrial manufacturing, automatic control,18 Advanced Fuzzy Logic Technologies in Industrial Applications automobile production, banks, hospitals, libraries and academic education. Fuzzy logic techniques have been widely applied in all aspects in today's society [6].

Fuzzy sets originated in the year 1965 and this concept was proposed by Lofti A.Zadeh. Since then it has grown and is found in several application areas. According to Zadeh, The notion of a fuzzy set provides a convenient point of departure for the construction of a conceptual framework which parallels in many respects of the framework used in the case of ordinary sets, but is more general than the latter and potentially, may prove to have a much wider scope of applicability, specifically in the fields of pattern classification and information processing."

A- Some of the fuzzy terminologies

- **Fuzzy logics** are multi-valued logics that form a suitable basis for logical systems reasoning under uncertainty or vagueness that allows intermediate values to be defined between conventional evaluations like true/false, yes/no, high/low, etc. These evaluations can be formulated mathematically and processed by computers, in order to apply a more human-like way of thinking in the programming of computers.
- **Fuzzy set** is a set containing elements that have varying degrees of membership in the set. Each element is mapped to [0, 1] by membership function

  $\mu A : X \longrightarrow [0,1]$ where X is the universal set.(2.1)

  A fuzzy set is fully characterized by its membership function. There are two kinds of representations of fuzzy set one discrete and the other continuous. A fuzzy set A in X may be represented as a set of ordered pairs of generic element x and its grade of membership function is given below [6].

$$A = \{(x, \mu_A(x)) / x \in X\}$$

When X is discrete, A is commonly written as, $A = \sum_X \mu_A(x)/x$ (2.2)

When X is continuous, $A = \int_X \mu_A(x)/x$ (2.3)

The concept of a fuzzy set is an extension of the concept of a crisp set. Similar to a crisp set a universe set U is defined by its membership function from U to [0,1]. Consider U to be a non-empty set also known as the universal set or universe of discourse or domain. A fuzzy set on U is defined as

$$\mu A(x) : U \rightarrow [0, 1] \quad (2.4)$$

Here:

μA is known as the membership function.

μA(x) is known as the membership grade of x.

- **Membership function:** is the degree of truth or degree of compatibility. The membership function is the crucial component of a fuzzy set. Therefore all the operations on fuzzy sets are defined based on their membership functions [8].
- **Core:** The core of a fuzzy set A is the set of all points with unit membership degree in A and is represented as:

  $$\text{core}(A) = \{x \text{ e } X / \mu A (x) = 1\} \quad (2.5)$$

- **Normal:** A fuzzy set A of X is called **normal** if there exists at least one element x in X such that μA (x) = 1. A fuzzy set A is normal if its core is nonempty. A fuzzy set that is not normal is called **subnormal**.
- **Height:** The height of a fuzzy set A is the largest membership grade of any element in A.

  $$\text{Height}(A) = \text{Max } \mu A(x) \text{ for all } x \text{ in } X \quad (2.6)$$

- **Support:** The support of a fuzzy set **A** is the set of all points with nonzero membership degree in **A.** It is denoted by:

  $$\text{Supp}(A) = \{x \in X / \mu A(x) > 0\} \quad (2.7)$$

- **Crossover point:** The crossover points of a membership function are defined as the elements in the universe for which a particular fuzzy set A has values equal to 0.5.
- **Alpha Cut:** It is one of the most important concepts of fuzzy sets. For a fuzzy set A and for a ϵ [0,1] the **α-cut** $A\alpha = \{x \epsilon X / \mu A(x) > \alpha\}$ (2.8)

  The strong α-cut $A\alpha = \{x \in X / \mu A(x) > \alpha\}$. (2.9)

  The α-cut set is a crisp set. This threshold cut restricts the domain of the fuzzy set.

Two main reasons why α -cuts are important are:

1- The alpha level set describes a power or strength that is used by fuzzy models to decide whether or not a truth value should be considered equivalent to zero. This is a facility that controls the execution of fuzzy rules as well as intersection of multiple fuzzy sets.

2- The strong α -cut at zero defines the support set for a fuzzy set [7].

Fig 2.1 shows the regions in the universe compromising the core, support, crossover points and alpha cut of a typical fuzzy set.

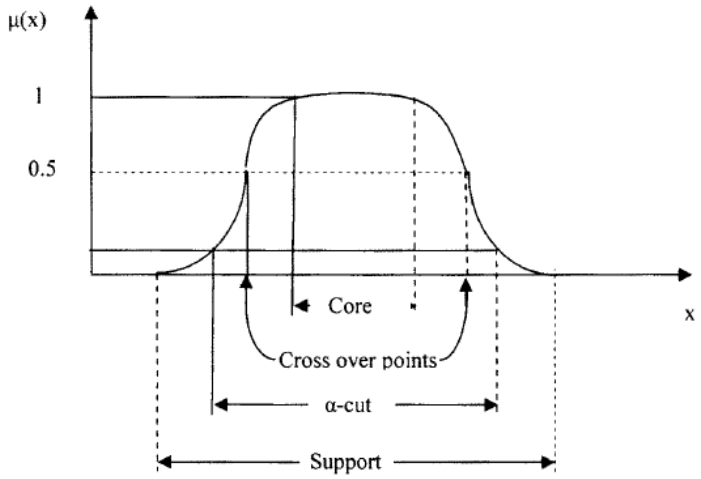


Figure 2-1: The main Fuzzy regions

B- **Fuzzy logic controller**

Fuzzy logic controllers are based on the combination of fuzzy set theory and fuzzy logic. Systems are controlled by fuzzy logic controllers based on rules instead of equations. This collection of rules is known as the rule base usually in the form of IF-THEN-ELSE statements. Here the IF part is known as the antecedent and the THEN part is the consequent. The antecedents are connected with simple Boolean functions like AND,OR, NOT etc.,

As shown in the Figure 2-2, in the fuzzy logic controller sequence outputs from a system are converted into a suitable form by the fuzzification block. Once all the rules have been defined based on the application, the control process starts with the computation of the rule consequences. The computation of the rule consequences takes place

within the computational unit. Finally, the fuzzy set is defuzzified into one crisp control action using the defuzzification module. The decision parameters of the fuzzy logic controller are as follows:

- **Input Unit :** Factors to be considered are the number of input signals and scaling of the input signal.
- **Fuzzification Unit:** The number of membership functions, type of membership functions are to be considered and convert classical data or crisp data into fuzzy data or Membership Functions (MFs)
- **Rule Base:** The total number of rules, type of rule structure (Mamdani or Takagi), rule weights etc. are to be considered.
- **Fuzzy Inference Process:** combine membership functions with the control rules to derive the fuzzy output
- **Defuzzification Unit:** Type of defuzzification procedure is to be considered use different methods to calculate each associated output [6] [8].

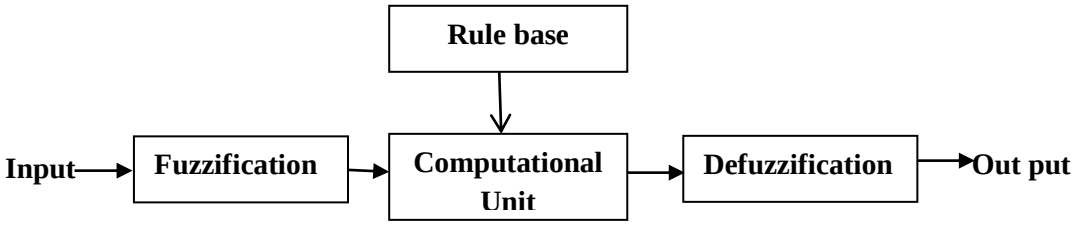


Figure 2-2: shows the Fuzzy Logic Controller sequences

C- Fuzzy logic advantages

- Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems. The theory of fuzzy logic provides a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning.
- Fuzzy systems are suitable for uncertain or approximate reasoning, especially for the system with a mathematical model that is difficult to derive.
- Fuzzy logic allows decision making with estimated values under incomplete or uncertain information
- fuzzy systems are very useful in two general contexts:

1- In situations involving highly complex systems whose behaviours are not well understood,

2- In situations where an approximate, but fast, solution is warranted.

D- The important characteristics of fuzzy logic controllers

1- User defined rules, which can be changed according to the application.

2- Robust.

3- Can be applied to control non-linear systems also.

4- Simple design.

5- Overall cost and complexity is low [9].

E- Representation of Membership Functions

In many practical instances fuzzy sets can be represented explicitly by families of parameterized functions, the most common being the following:

i. Triangular Function

The membership definition for a triangular function is given as:

$$A(x) = \begin{cases} 0, & if\ x \leq a \\ \frac{x-a}{b-a}, & if\ x\ \epsilon[a,b] \\ \frac{c-x}{c-b}, & if\ x\ \epsilon[b,c] \\ 0, & if\ x\ \geq a \end{cases} \quad (2.10)$$

where and a and b denote the lower and upper bounds, respectively, for nonzero values of A(x).

ii. S-Function

The membership definition for a S-function is given as:

$$A(x) = \begin{cases} \mathbf{0}, & if x \leq a \\ \mathbf{2}(\frac{x-a}{b-a})^2, & if\ x\ \epsilon[a,m] \\ \mathbf{1\text{-}2}(\frac{x-a}{b-a})^2, & if\ x\ \epsilon[m,b] \\ \mathbf{1}, & if\ x>b \end{cases} \quad (2.11)$$

iii. Trapezoidal function

The membership definition for a trapezoidal function is given as:

$$A(x) = \begin{cases} \mathbf{0}, & if x \leq a \\ \frac{x-a}{b-a}, & if\ x\ \epsilon[a,b] \\ \mathbf{1}, & if\ x\ \epsilon[b,c] \\ \frac{d-x}{d-c}, & if\ x\ \epsilon[c,d] \\ \mathbf{0}, & if\ x \geq b \end{cases} \quad (2.12)$$

iv. Gaussian Function

The membership definition for a Gaussian function is given as:

$$A(x) = -e^{\frac{(x-c)^2}{2\sigma^2}}, where\ 2\sigma^2 > 0. \quad (2.13)$$

v. **Exponential Function**

The membership definition for a exponential function is given as:

$$A(x) = \frac{1}{1+(k-m)^2}, k > 1 \quad (2.14)$$

Or:

$$A(x) = \frac{(k-m)^2}{1+(k-m)^2}, k > 0 \quad (2.15)$$

F- Fuzzy inference

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves all the topics such as fuzzification, defuzzification, implication, and aggregation. Expert control/modelling knowledge, experience, and linking the input variables of fuzzy controllers/models to output variable (or variables) are mostly based on fuzzy rules. A fuzzy expert system consists of four components namely, the fuzzifier, the inference engine, and the defuzzifier, and a fuzzy rule base [8].

G- Fuzzy implication methods

There are various techniques in which fuzzy implication may be defined. These relationships are mostly derived from multivalued logic theory. The following are some of the common techniques of fuzzy implication:

- **Mamdani system**

  This method is widely accepted for capturing expert knowledge. It allows us to describe the expertise in more intuitive, more human like manner. However, Mamdani type FIS entails a substantial computational burden.

- **TSK model**

  The Takagi-Sugeno-Kang (TSK) model was introduced by T.Takagi and M.Sugeno .This model reduce the number of rules required by Mamdani model, especially for complex and high dimensional problems. To achieve this goal, the TSK model replaces the fuzzy sets in the consequent part (then-part) of the Mamdani rule with a linear equation of the input variables.

- **The standard additive model (SAM)**

  The structure of fuzzy rules in SAM is identical to that of the Mamdani model. It was introduced by B.Kosko.There are four differences between the inference scheme of these two models:

  1- SAM assumes the inputs are crisp, while Mamdani model handles both crisp and fuzzy inputs.

  2- SAM uses the scaling inference method while Mamdani uses the clipping method.

  3- SAM uses addition to combine the conclusion of fuzzy rules, while the mamdani model uses max.

  4- SAM includes the centroid defuzzification technique, while the Mamdani model does not insist on a specific defuzzification method [9].

2.2 Control System

A Control System is a device, or a collection of devices that manage the behaviour of other devices. Some devices are not controllable. A control system is an interconnection of components connected or related in such a manner as to command, direct, or regulate itself or another system.

2.2.1 Classical and modern controllers

Classical and Modern control methodologies are named in a misleading way, because the group of techniques called “Classical” was actually developed later than the techniques labelled "Modern". However, in terms of developing control systems,

Modern methods have been used to great effect more recently, while the Classical methods have been gradually falling out of favor. Most recently, it has been shown that Classical and Modern methods can be combined to highlight their respective strengths and weaknesses.[10]

Some of the classical methods are methods involving the Laplace Transform domain. Physical systems are modelled in the so-called "time domain", where the response of a given system is a function of the various inputs, the previous system values, and time. As time progresses, the state of the system and its response change. However, time-domain models for systems are frequently modelled using high-order differential equations which can become impossibly difficult for humans to solve and some of which can even become impossible for modern computer systems to solve efficiently. To counteract this problem integral transforms, such as the Laplace Transform and the Fourier Transform, can be employed to change an Ordinary Differential Equation (ODE) in the time domain into a regular algebraic polynomial in the transform domain. Once a given system has been converted into the transform domain it can be manipulated with greater ease and analysed quickly by humans and computers alike.

Modern Control Methods, instead of changing domains to avoid the complexities of time domain ODE mathematics, converts the differential equations into a system of lower order time domain equations called State Equations, which can then be manipulated using techniques from linear algebra.

Control Methods were designed to try and incorporate the emerging power of computer systems into previous control methodologies. A special transform, known as the ZTransform, was developed that can adequately describe digital systems, but at the same time can be converted (with some effort) into the Laplace domain. Once in the Laplace domain, the digital system can be manipulated and analyzed in a very similar manner to Classical analog systems [10].

2.3 Microcontroller

A microcontroller is essentially an inexpensive single chip computer . Single chip means the entire computer system lies within the confines of a sliver of silicon encapsulated inside the plastic housing of an integrated circuit.

The microcontroller has features similar to those of a standard personal computer. The microcontroller contains:

- Central Processing Unit (CPU).
- Random Access Memory (RAM).
- Read Only Memory (ROM).
- Input and Output lines (I/O).
- serial and parallel ports.
- Timers.
- sometimes other built in peripherals such as analog to digital (A/D) and digital to analog (D/A) converters.
- The key feature
- The microcontroller's capability of uploading, storing, and running a program[11].

2.3.1 Arduino UNO

Arduino Uno shown in Figure 2.3, is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analogue inputs, a 16 MHz quartz crystal, a USB connection, a power jack and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again[13].

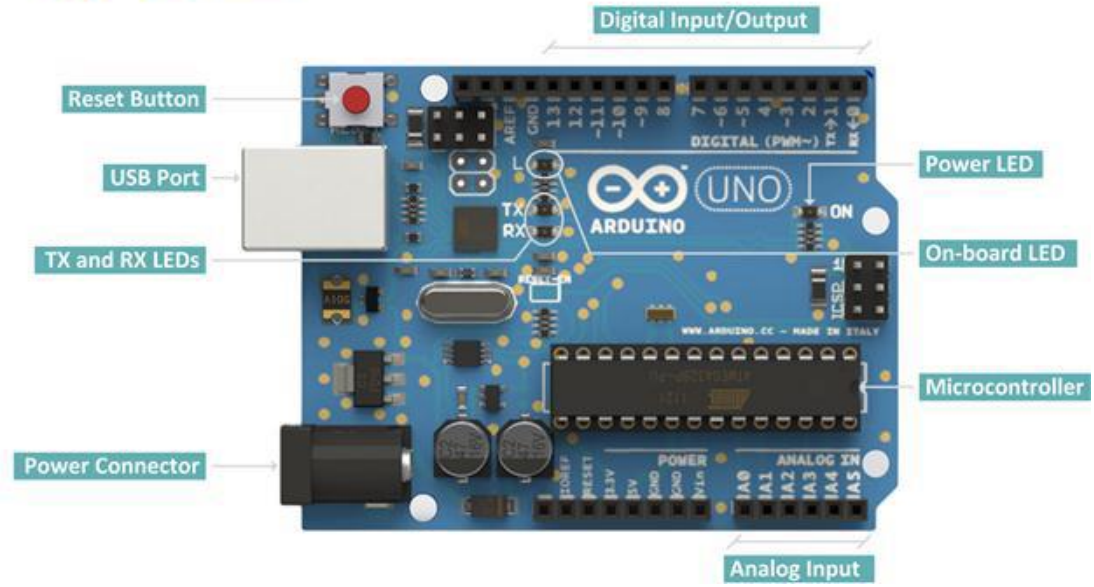


Figure 2-3: Arduino Uno microcontroller board

A- Arduino programming

The Arduino Uno can be programmed with the Arduino Software (IDE).The ATmega328 on the Arduino Uno comes pre-programmed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.

B- Arduino power

The Arduino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power shown in Figure 2.4 can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the power connector [11].

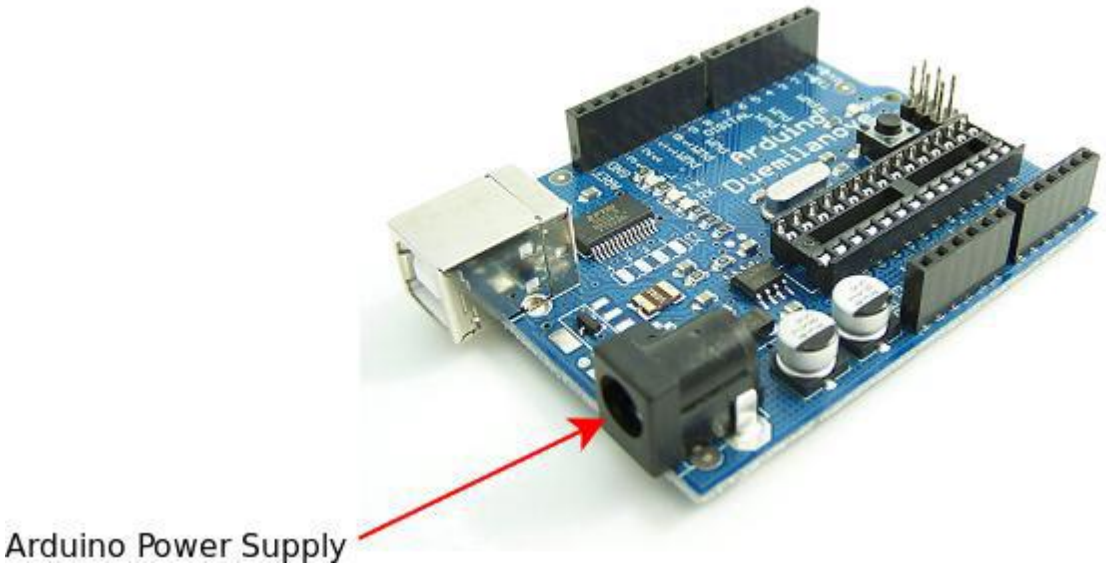


Figure 2-4: shows Arduino power supply entrance

C- Arduino Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM

D- Arduino development

The Arduino Integrated Development Environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface [11].

Arduino programs are written in C or C++ The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output.

Operations much easier. Users only need define two functions.

To make a runnable cyclic executive program:

- **Setup ():** a function run once at the start of a program that can initialize settings.

- **Loop ():** a function called repeatedly until the board powers off.

2.4 Literature Review

An incalculable number of researches were done in the same sphere to robotic arms for several purposes using different controlling programs for their performances and accuracy.

- In January 2013 Journal of Theoretical and Applied Information Technology published a paper titled with (Position Control Of Arm Mechanism Using PID Controller) [12].

  Here The rotary encoder was successfully used as the feedback signal to the PIC microcontroller. PID controller was successfully applied in the lifting system of robotic arm and used to replace the ON and OFF controller.

  which produced a more stable and consistent output compared to the ON and OFF controller system. With the PID controller system, the movement for lifting robotic arm is greatly smooth and reached to the desired position precisely.
- In October 2013 the International Journal of Mechanical & Mechatronics Engineering IJMME-IJENS a paper titled by (Development of a Prototype Robot Manipulator for Industrial Pick-and-Place Operations) [13].

  The construction of a robot arm for simple pick-and-place operations has been considered in this paper. The system hardware and software have been highlighted and discussed, and tests carried out on the complete assembly shows that the implementation is satisfactory. This work uses the independent joint control method for the pick-and-place operations.
- In June 2016 the international journal of scientific and technology research published paper titled by (Position Control Method For Pick And Place Robot Arm For Object Sorting System) [14].

  In this journal forward and reverse kinematic methods has been fully explained and calculated with mathematically .From the result of this, articulated robot arm has been tested by Arduino Software with experimentally. When testing the robot arm for pick up and place operation, there is little error occurs. The occurrence of this error is because of the limitation of selected servo motors and some

mechanism structure of robot arm. Whenever, the maximum rotating angle of the chosen servo motors cannot exactly rotate 180◦.

- In April 2017 IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) published a paper titled with (Pick and Place Robotic Arm Implementation Using Arduino) [15].

Here a robotic arm is implemented using arduino to pick and place objects more safely without incurring much damage. The robotic arm used here contain a soft catching gripper which safely handle the object. The use of soft catching gripper and low power wireless communication technique like Bluetooth makes our system more effective when compared to other systems. The proposed system is capable of lifting only small weights.

- In Sep -2017 the International Research Journal of Engineering and Technology (IRJET) published a paper titled with (Development of Pick and Place Robot for Industrial Applications) [16].

The main task of this project was to program the AVR microcontroller on both the base station and the robot interfaced to the radio packet controller module which would enable to wirelessly control the robot and to program the GUI application which would enable us to serially control the base station. The design and development of pick and place robot has been carried out. A prototype was confirmed functional working of robot system. A robotic arm is implemented using Atmega16 in pick and place objects more safely without incurring much damage.

- In June 2018 Cristina I. Muresan and others published a paper which titled with (Experimental Validation of a Novel Auto-Tuning Method for a Fractional Order PI Controller on an UR10 Robot) [17].

This paper presents an experimental validation of a novel auto-tuning method for robust fractional order controllers.

A case study has been used consisting of a pick and place movement of an UR10 robot. The experimental results, considering two different robot configurations, demonstrate that the designed fractional order PI controller is indeed robust Further work includes comparisons with other auto-tuning methods to demonstrate the benefits of using the proposed algorithm.

CHAPTER THREE

SYSTEM CONTROL DESIGN

CHAPTER THREE

SYSTEM CONTROL DESIGN

Tis chapter gives a view about the used module , the connection and the fuzzy rules that used for module programming**.**

3.1 Robot Arm

In the current module of the robot arm , shown in Figure 3-1 the MPU5060 sensor was used to sense the robotic arm position which will sense the angle of the robotic arm then the signal will be send to the arduino UNO microcontroller.

The robot was programmed by using arduino UNO while the potentiometers were used to control the movement of the motors of the robotic arm and the sensor.

Figure 3-1: The used module

3.1.1 System's components

The system is made up of the following component:

- **Five DC motors**

(DCM1): is a DC motor to open and close the gripper, bidirectional

(DCM2): is a DC motor to band the hand.

(DCM3): is a DC motor to rotate hand (5 RPM).

(DCM4): is a DC motor to move hand (2 RPM).

(DCM5): it is a DC motor to rotate the gripper.

Depending on the size of the motor, we can simply connect an Arduino PWM output to the base of transistor or the gate of a MOSFET and control the speed of the motor by controlling the PWM output. The low power Arduino PWM signal switches on and off the gate at the MOSFET through which the high power motor is driven.

PWM, or pulse width modulation is a technique which allows us to adjust the average value of the voltage that's going to the electronic device by turning on and off the power at a fast rate. The average voltage depends on the duty cycle, or the amount of time.

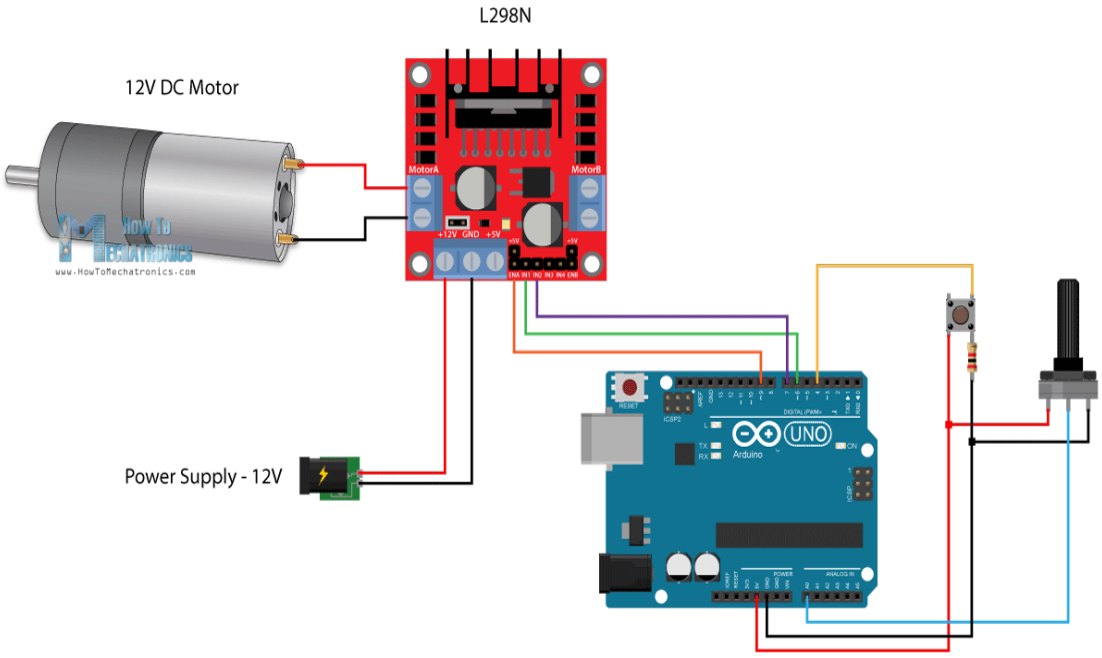


Figure 3-2: Connection of dc motor through driver l298 with Arduino

- **Potentiometers**

  In arduino the three potentiometer's wires were connected to the board. The first goes from one of the outer pins of the potentiometer to ground. The second goes from the other outer pin of the potentiometer to 5 volts. The third goes from the middle pin of the potentiometer to the analog pin A0.

- **Motion Processing Unit (MPU6050)**

  The MPU 6050 communicates with the Arduino through the I2C protocol. The MPU 6050 is connected to Arduino as shown in Figure 3-3

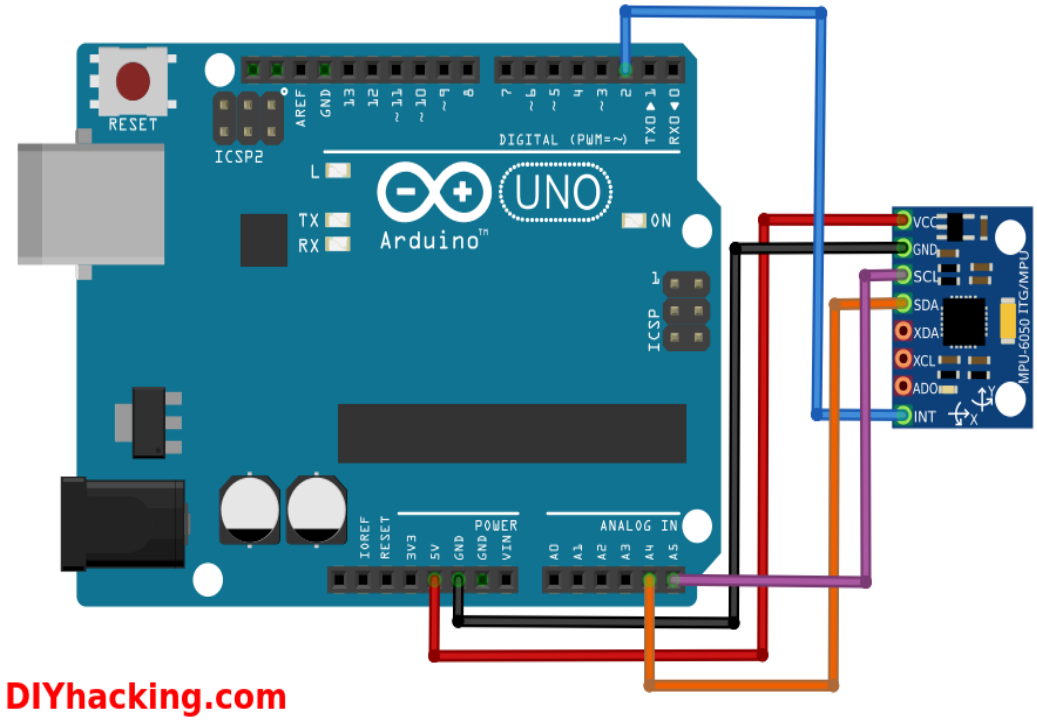


Figure 3-3: PMU connection

- **Arduino UNO**

  The robotic arm movements are controlled by changing the angles through the potentiometers. when the direction of the potentiometer is changed to specific angle then the concerned motor will move accordingly. The first potentiometer is connected to a motion processing unit sensor type MPU6050 first then the reading of the sensor will be processed by the controller and then actuate the first motor.

  Here DCM2, DCM3, DCM4 depend on the DCM1 motion and all of them are connected to L298 driver as shown in figure 3-4 which describes the system components and how they are connected including the input and output devices with the interfacing too

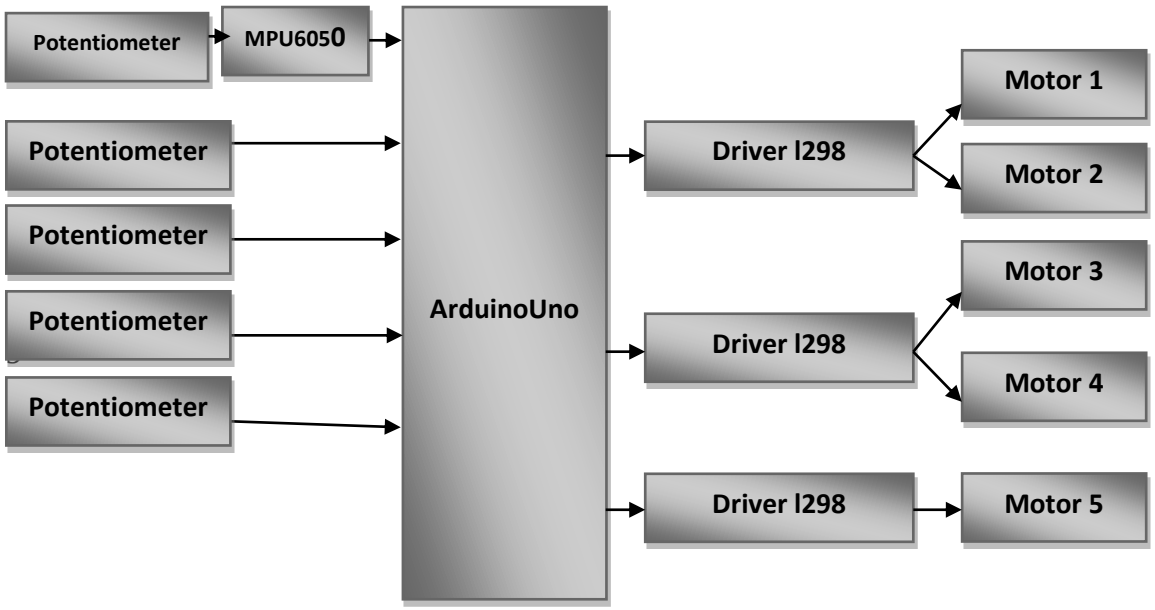


Figure 3-4: Block diagram represent the case study's component

3.2 Fuzzy Controller Design

For fuzzy implication to the case study, Mamdani model technique was used and the membership functions of the inputs, outputs and 49 rules were applied using MATLAB.

The Figuers 3-5, 3-6, 3-7 represent the inputs and the out put with their membership functions.

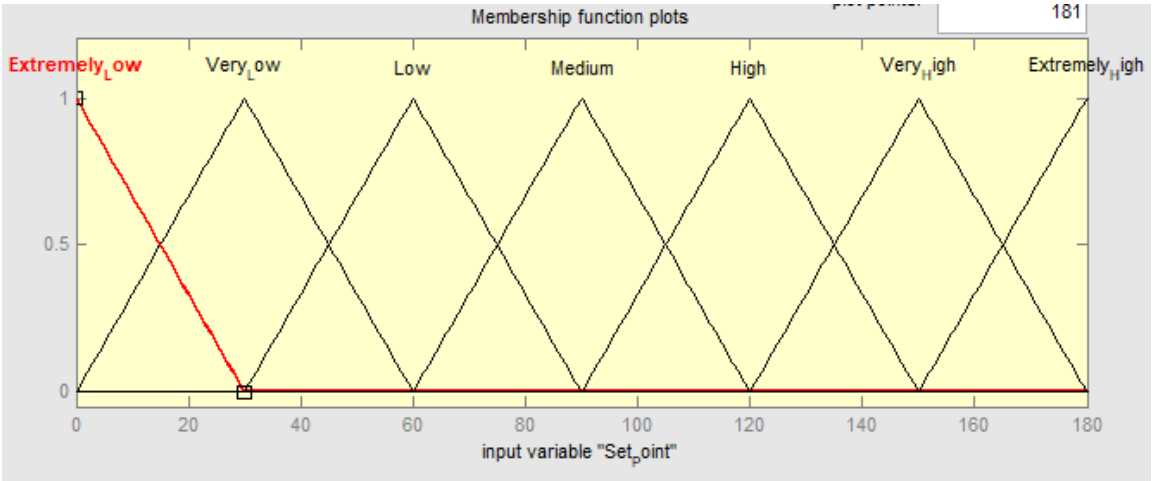


Figure 3-5 : The set point's membership function plot

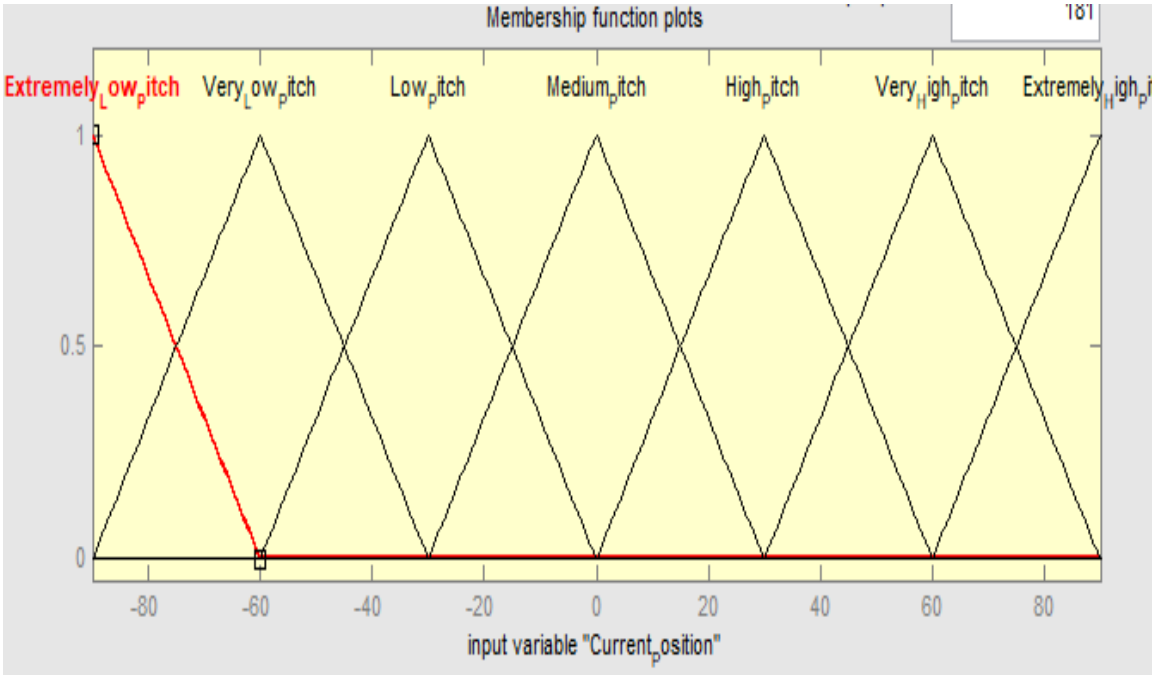


Figure 3-6 : The current position's membership function plot

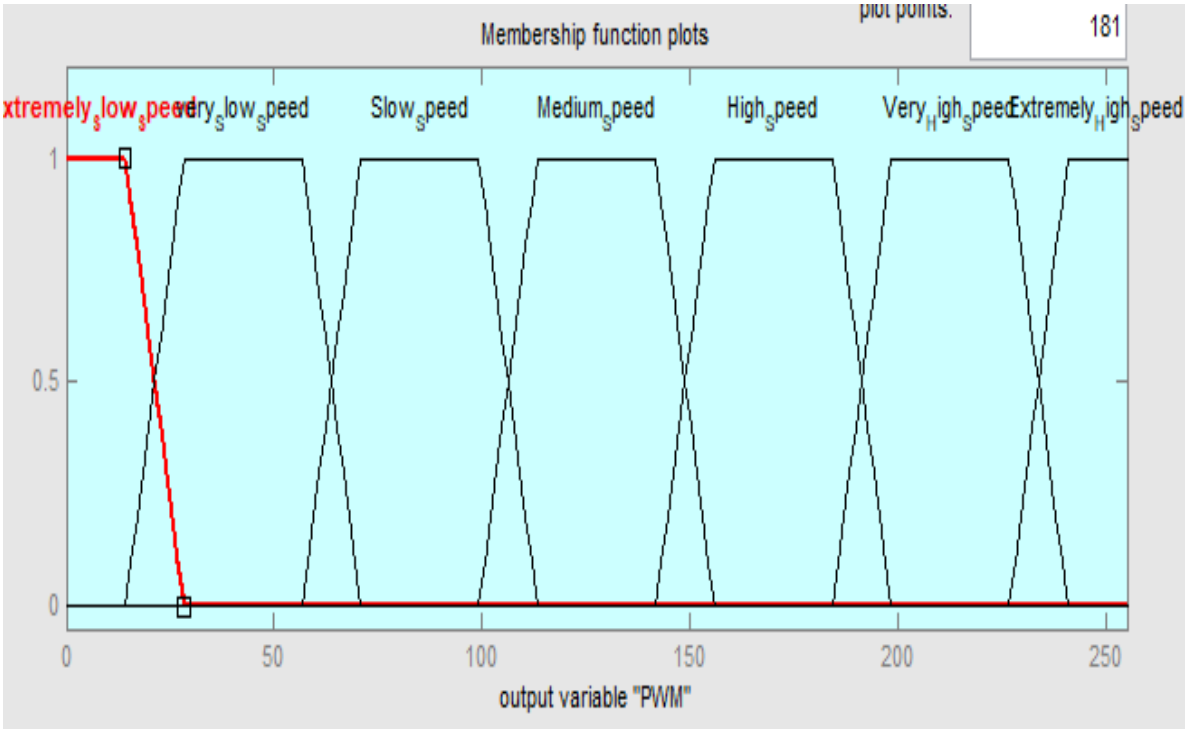


Figure 3-7 : The output plot.

3.2.3 System Flowchart

The overall system is represent as a flowchart that shown all system steps starting from the current position which compared to the desired value till detecting the error and correct it as in Figure 3-8

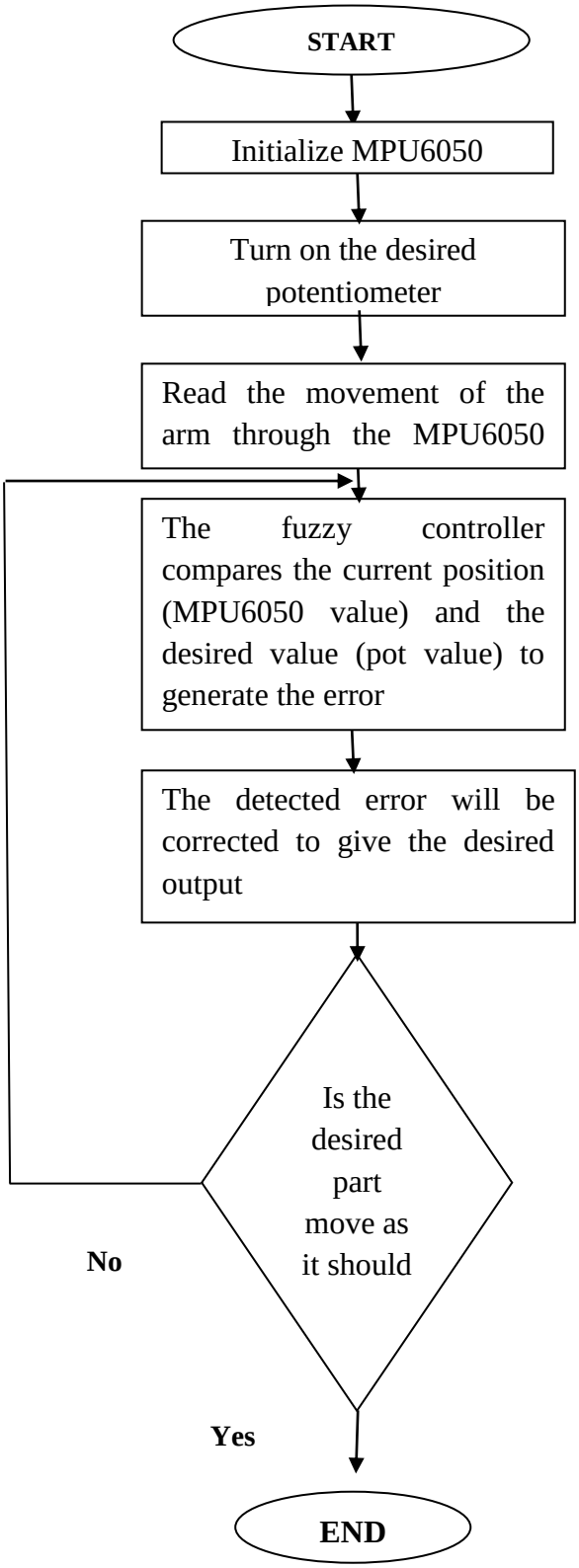


Figure 3-8: The overall system flow chart

CHAPTER FOUR

SYSTEM IMPLEMENTATION , TESTING AND RESULTS

CHAPTER FOUR

SYSTEM IMPLEMENTATION , TESTING AND RUSELTS

Chapter Four shows the system implementation and its testing to figure out whether the goal is achived or no.

4.1 System Implemenntation Steps

Using MATLAB the fuzzy library was opened by writing (fuzzy) on the command window then the window was appered as shown in Figure 4-1

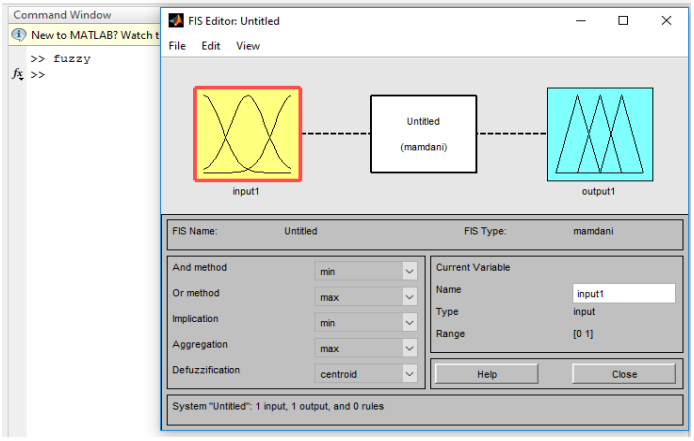


Figure 4-1: Fuzzy library in MATLAB

The input and output functions were adjusted with the suitable function representation where input-1 represents the set point and input-2 for the current-position while the output for the PMW as shown in figures 4-2 to 4-5.

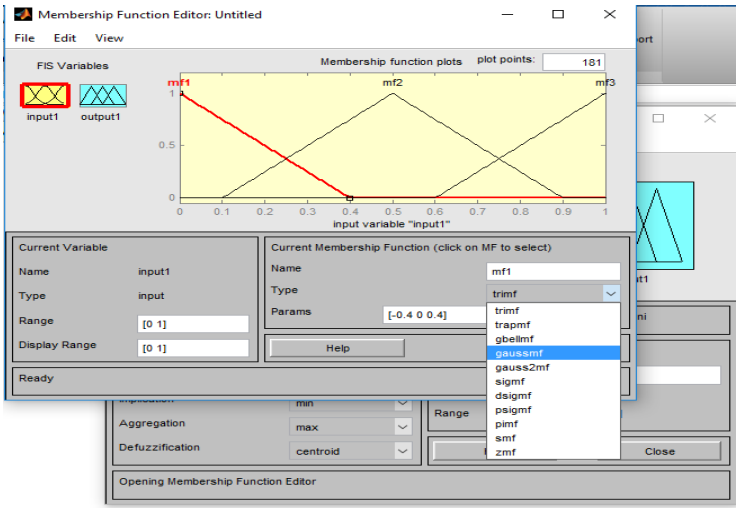


Figure 4-2: The input function adjustment

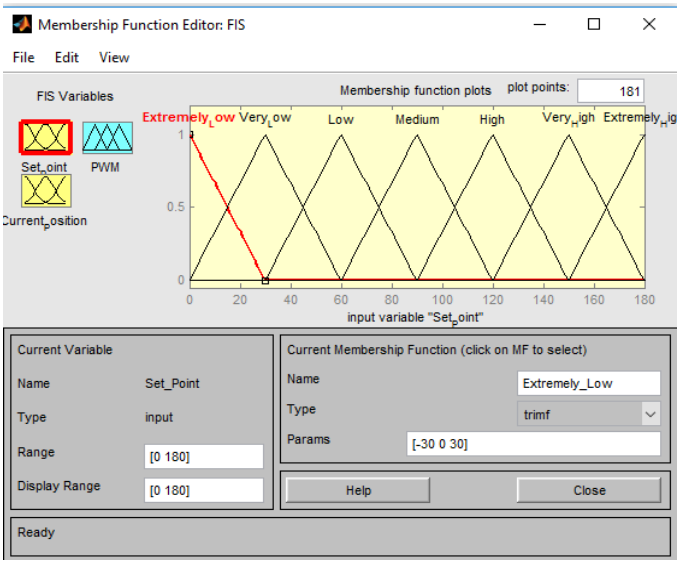


Figure 4-3: The set point representation

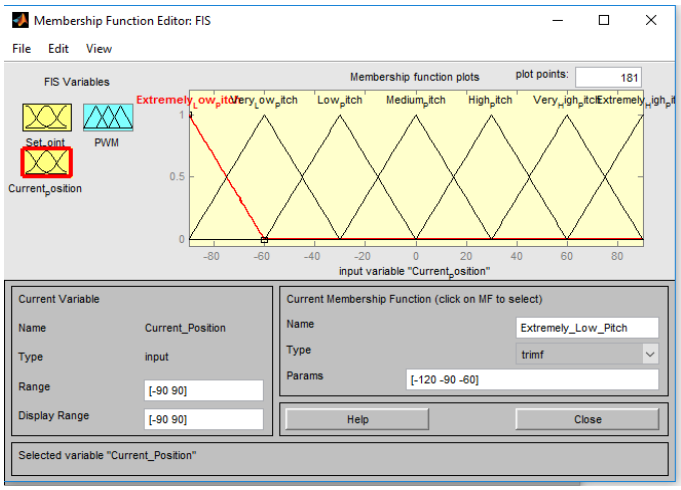


Figure 4-4: The current position representation

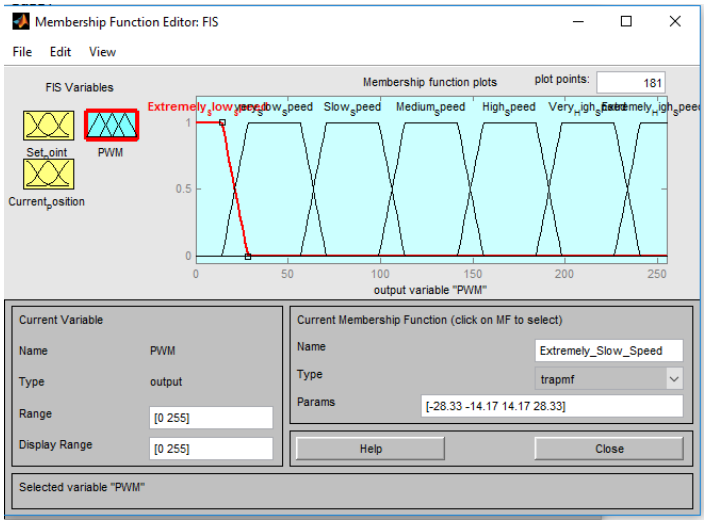


Figure 4-5: The output representation

The rules were added and FIS file were saved as shown in the figuer 4-6 .

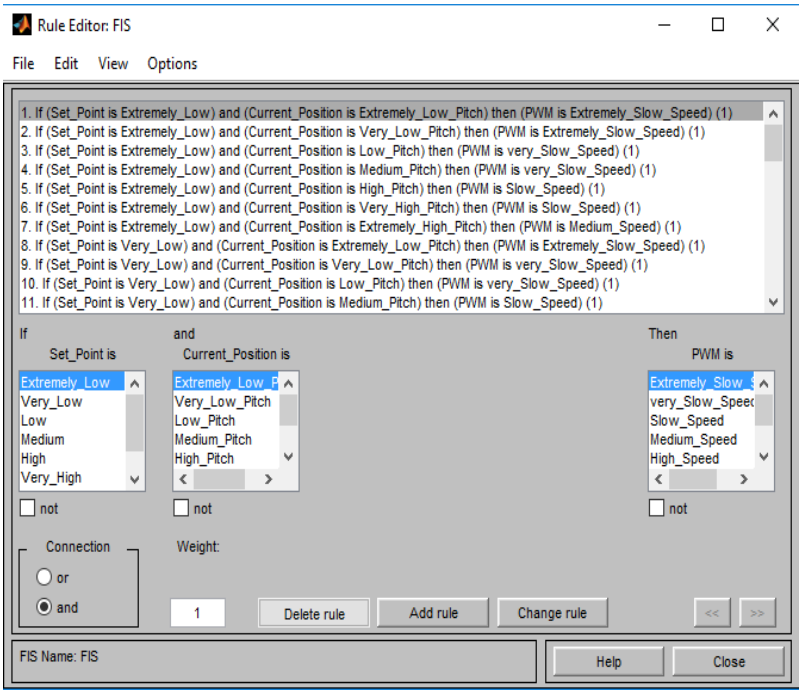


Figure 4-6: Rules representation

The FIS file is converted into Arduino form using Matlab Fuzzy Inference System (FIS) to Arduino converter program as shown in Figure 4-8

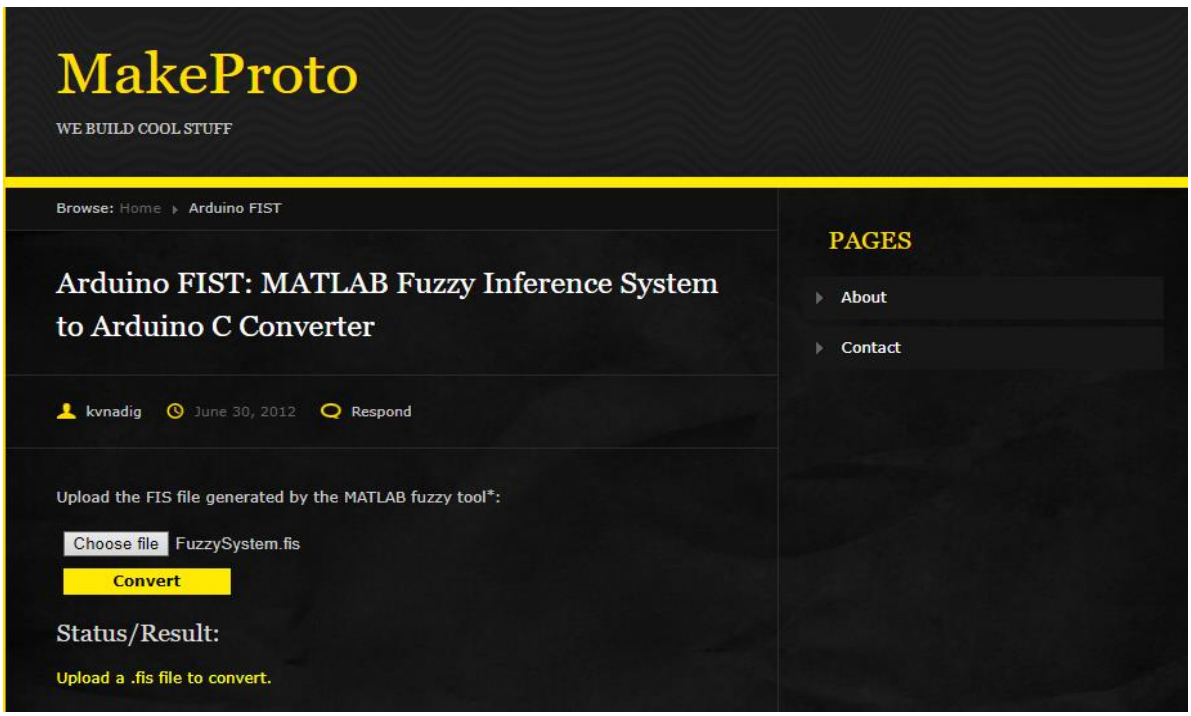


Figure 4-7: Matlab Fuzzy Inference System to Arduino converter program

Enentually the code were opened with Arduino software , compilied, uploaded to the robot arm, tested and the results were showed in Figure 4-8

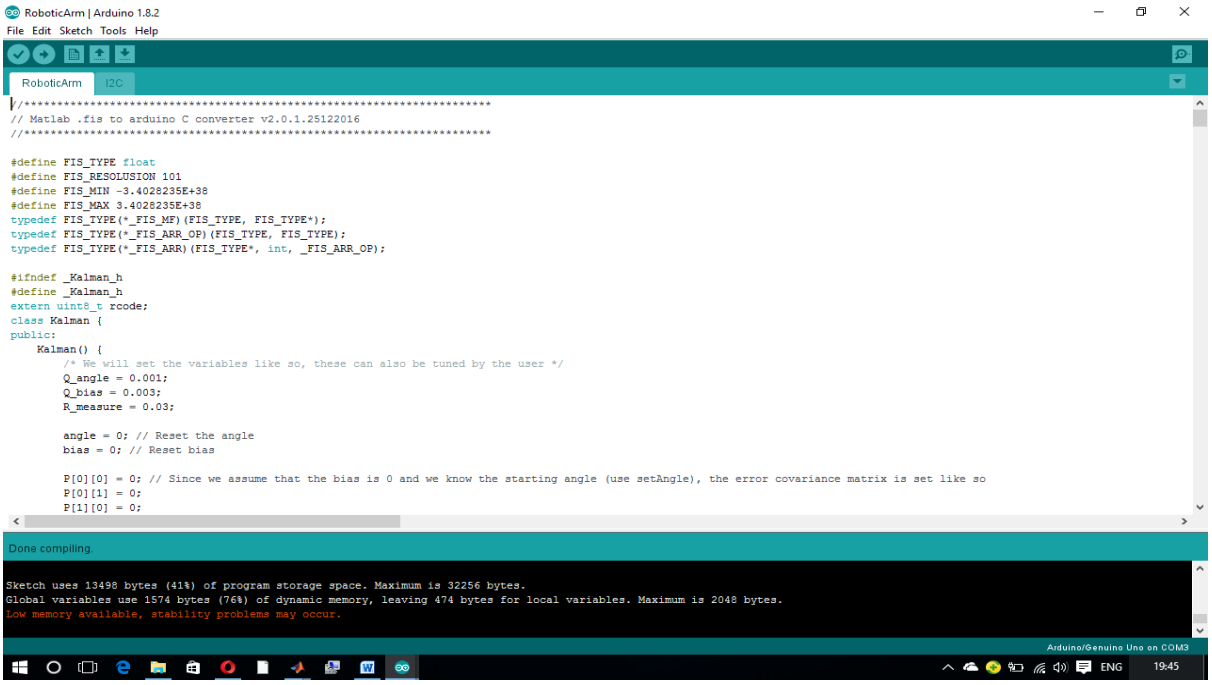


Figure 4-8: Code compiling in Arduino

4.2 System Results

For different positions the accuracy was perfect as shown in figures 4-9 – 4-11 which shows that there is an improvement to the previous model's performance.

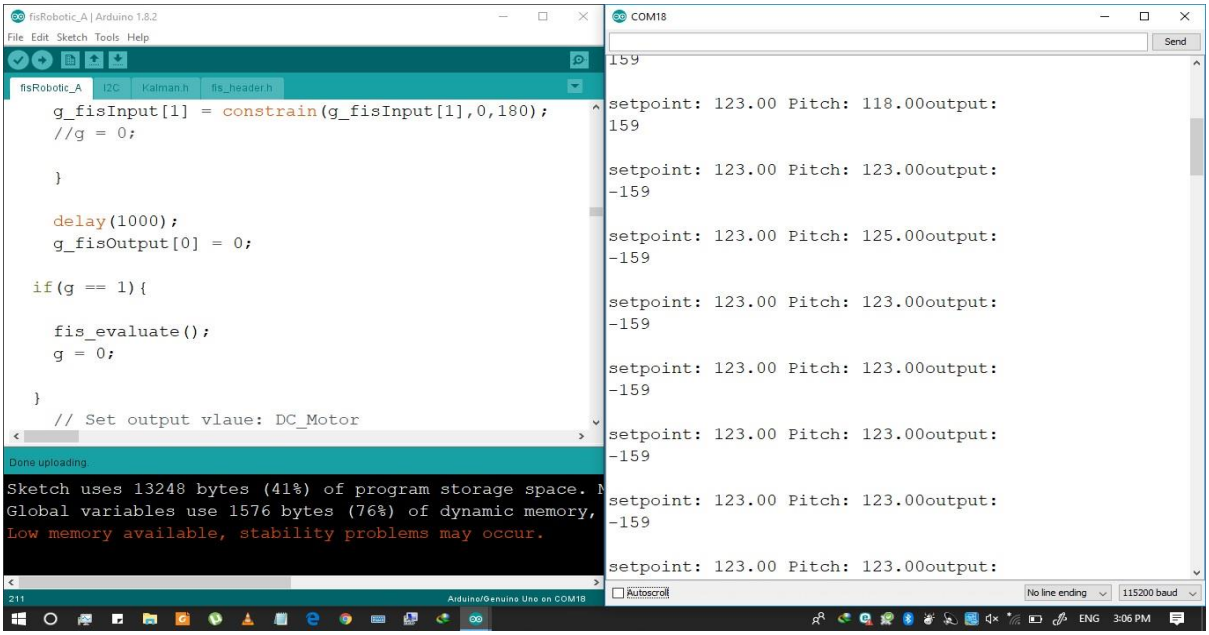


Figure 4-9: The testing results1

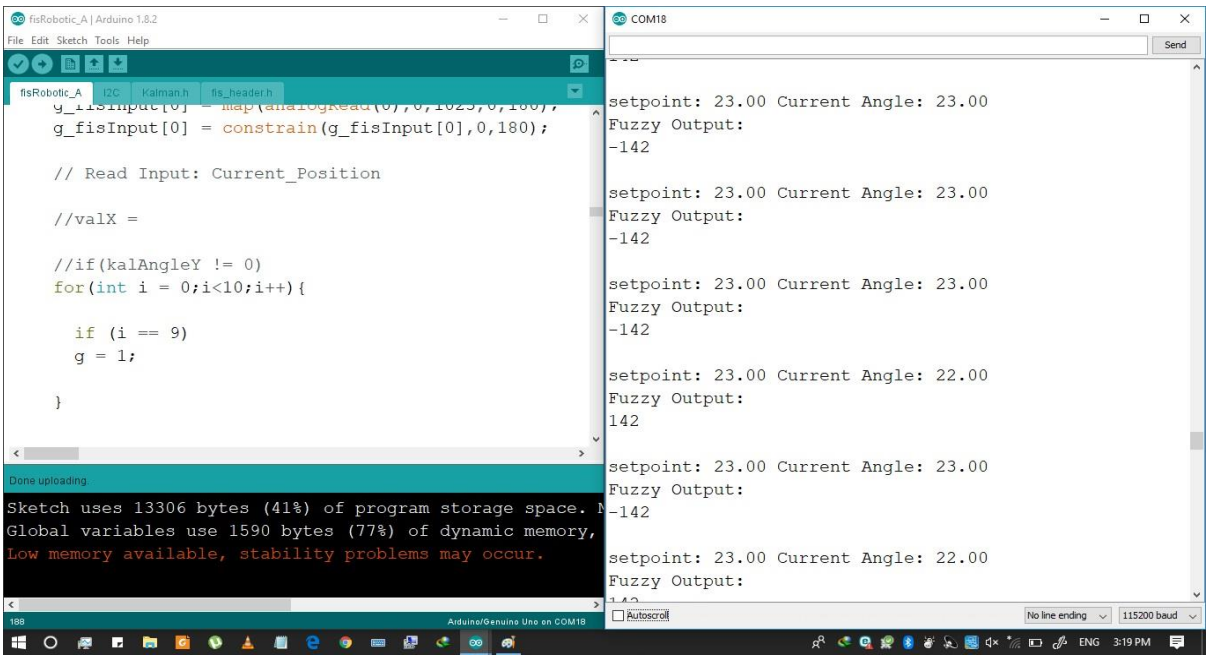


Figure 4-10: The testing results 2

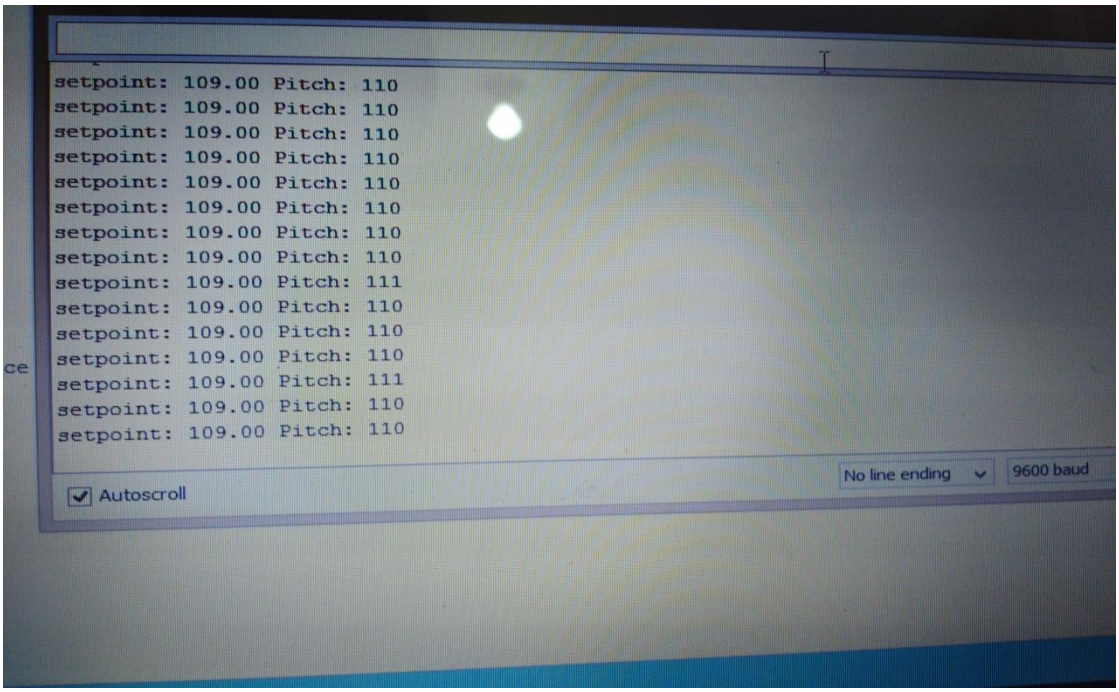


Figure 4-11: The results using PID

- From Figures 4-9 , 4-10 and 4-11 it is obvious that the Fuzzy logic controller has achieved higher accuracy than PID controller.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

Chapter Five shows the results which compared with the previous work's results as a conclusion, and the recommendations for the forth coming researchers.

5.1 Conclusion

As we continuously search for implementing the suitable controller of the robotic arms to improve the accuracy of them in order to achieve high response and performance, here by using hardware and software components the degree of the arm is read by the sensor Motion Processing Unit (MPU6050) and the Fuzzy Inference System which can deal with the nonlinearity and overcome the PID missing is used, the accuracy of the robotic arm which is controlled with Proportional Integral Differential controller (PID) is improved and a higher performance of the robotic arm is achieved after comparing the current results with the previous study for the same robotic arm using the PID controller's results.

5.2 Recommendations

For the forth researchers the upcoming points are highly recommended which can help in improving the accuracy and the performance of the robot's arm as well:

- A concerted modeling and control efforts are needed together with the development of good hardware to make arms and hands that can perform anything but the simplest of pick and place operation that are prevalent in industry .The pick and place robot is having the very vast area of application.
- Increasing the number of the motors to increase the degree of freedom
- Connecting the MPU 6050 sensor very close because the length of the wire effect on the reading of the sensor
- Using more developed connection between MPU 6050 sensor and the system to avoid the noise.

References

1- Prof. Alessandro De Luca, " Industrial Robotics" , 2016.

2- Ahmed Zakari Alassar, "Modeling and Control of 5DOF Robot Arm Using Supervisory Control" , March 2010.

3- RK Mittal and IJ Nagarath, "Robotics and Control", BITS Pilani, 2003.

4- LazoRoljić , "History of Industrial Robots" , October 2012.

5- Al teef etal, "Design and the mechanism of controlling a robotic arm", Syrian Private University Faculty of computer & Informatics Enineering , August 2015.

6- Ying Bai and Dali Wang, " Fundamentals of Fuzzy Logic Control – Fuzzy Sets,Fuzzy Rules and Defuzzifications", Springer- Verlag London,2006

7- http://shodhganga.inflibnet.ac.in/bitstream/10603/101833/11/11_chapter%201.pdf 26th.Dec.2018.

8- S. Sumathi, Surekha Paneerselvam, "Computational Intelligence Paradigms: Theory & Applications using MATLAB" ,

9- http://shodhganga.inflibnet.ac.in/bitstream/10603/134769/9/09_chapter%203.pdf 25th.Jan.2019.

10- Patrick Anderson, "Control Systems ClassicalControl", First Edition, Published by:Global Media1819- Bhagirath Palace-Chandni Chowk, 2009 .

11- John Iovine, "PIC Robotics A Beginner's Guide to Robotics Projects Using the PICmicro", 2004.

12- Asnor Juraiza et al, "Position control of Arm Mechanism Using PID Controller", Journal of Theoretical and Applied Information Technology, 20th January 2013.

13- Ayokunle A. Awelewa et al, "Development of a Prototype Robot Manipulator for Industrial Pick and Place Operations", International Journal of Mechanical and Mechatronics Engineering IJMME-IJENS, October 2013.

14- Khin Moe Myint et al, "Position Control Method For Pick And Place Robot Arm For Object Sorting System", International Journal of Scientific and Technology Research volume5- issue6 , June 2016 .

15- Ashly Baby et al, " Pick and Place Robotic Arm Implementation Using Arduino", IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE), April 2017.

16- Vishakha Borkar, Prof G.K.Andurkar, “Development of Pick and Place Robot for Industrial Applications”, International Research Journal of Engineering and Technology (IRJET) - Volume: 04 Issue: 09 , Sep -2017

17- Cristina I. Muresan et al, “Experimental Validation of a Novel Auto-Tuning Method for a Fractional Order PI Controller on an UR10 Robot”, 30 June 2018.

APPENDICES

Appendix (A)

Rules

- If (set point is Extremly_low) and (current position is Extremly_low pitch) Then (PMW is Extremly_slow speed)
- If (set point is Extremly_low) and (current position is Very_low pitch) Then (PMW is Extremly_slow speed)
- If (set point is Extremly_low) and (current position is low pitch) Then (PMW is Very_slow speed)
- If (set point is Extremly_low) and (current position is Mediem pitch) Then (PMW is Very_slow speed)
- If (set point is Extremly_low) and (current position is High pitch) Then (PMW is Slow speed)
- If (set point is Extremly_low) and (current position is Very_High pitch) Then (PMW is Slow speed)
- If (set point is Extremly_low) and (current position is Extremely_High pitch) Then (PMW is Medium speed)
- If (set point is Very_low) and (current position is Extremly_low pitch) Then (PMW is Extremly_slow speed)
- If (set point is Very_low) and (current position is Very_low pitch) Then (PMW is Very_slow speed)
- If (set point is Very_low) and (current position is low pitch) Then (PMW is Very_slow speed)
- If (set point is Very_low) and (current position is Mediem pitch) Then (PMW is Very_slow speed)
- If (set point is Very_low) and (current position is High pitch) Then (PMW is Slow speed)
- If (set point is Very_low) and (current position is Very_High pitch) Then (PMW is Medium speed)
- If (set point is Very_low) and (current position is Extremely_High pitch) Then (PMW is High speed)
- If (set point is Low) and (current position is Extremly_low pitch) Then (PMW is Very_Slow speed)

- If (set point is Low) and (current position is Very_low pitch) Then (PMW is Very_Slow speed)
- If (set point is Low) and (current position is low pitch) Then (PMW is Slow_ speed)
- If (set point is Low) and (current position is Medium pitch) Then (PMW is Slow_speed)
- If (set point is Low) and (current position is High pitch) Then (PMW is Medium_ speed)
- If (set point is Low) and (current position is Very_High pitch) Then (PMW is High_speed)
- If (set point is Low) and (current position is Extremely_High pitch) Then (PMW is High_ speed)
- If (set point is Medium) and (current position is Extremly_low pitch) Then (PMW is Very_slow speed)
- If (set point is Medium) and (current position is Very_low pitch) Then (PMW is Slow_ speed)
- If (set point is Medium) and (current position is Low_ pitch) Then (PMW is Slow_ speed)
- If (set point is Medium) and (current position is Medium_ pitch) Then (PMW is Medium_ speed)
- If (set point is Medium) and (current position is High_ pitch) Then (PMW is High_speed)
- If (set point is Medium) and (current position is Very_High pitch) Then (PMW is High_ speed)
- If (set point is Medium) and (current position is Extremely_High pitch) Then (PMW is Very_High speed)
- If (set point is High) and (current position is Extremly_low pitch) Then (PMW is Slow_ speed)
- If (set point is High) and (current position is Very_low pitch) Then (PMW is Slow_ speed)
- If (set point is High) and (current position is Low_ pitch) Then (PMW is Medium_ speed)

- If (set point is High) and (current position is Medium_ pitch) Then (PMW is High_ speed)
- If (set point is High) and (current position is High_ pitch) Then (PMW is High_speed)
- If (set point is High) and (current position is Very_High pitch) Then (PMW is Very_High_ speed)
- If (set point is High) and (current position is Extremely_High pitch) Then (PMW is Very_High speed)
- If (set point is Very_High) and (current position is Extremly_low pitch) Then (PMW is Slow_ speed)
- If (set point is Very_High) and (current position is Very_low pitch) Then (PMW is Medium_ speed)
- If (set point is Very_High) and (current position is Low_ pitch) Then (PMW is High_ speed)
- If (set point is Very_High) and (current position is Medium_ pitch) Then (PMW is High_ speed)
- If (set point is Very_High) and (current position is High_ pitch) Then (PMW is High_speed)
- If (set point is Very_High) and (current position is Very_High pitch) Then (PMW is Very_High_ speed)
- If (set point is Very_High) and (current position is Extremely_High pitch) Then (PMW is Extremly_High speed)
- If (set point is Extremly_High) and (current position is Extremly_low pitch) Then (PMW is Medium_ speed)
- If (set point is Extremly_High) and (current position is Very_low pitch) Then (PMW is High_ speed)
- If (set point is Extremly_High) and (current position is Low_ pitch) Then (PMW is High_ speed)
- If (set point is Extremly_High) and (current position is Medium_ pitch) Then (PMW is Very_High_ speed)
- If (set point is Extremly_High) and (current position is High_ pitch) Then (PMW is Very_High_speed)

- If (set point is Extremly_High) and (current position is Very_High pitch) Then (PMW is Extremly_High_ speed)
- If (set point is Extremly_High) and (current position is Extremely_High pitch) Then (PMW is Extremly_High speed)

Appendix (B)

Code

```
//*****************************************************************
****

// Matlab .fis to arduino C converter v2.0.1.25122016

// - Karthik Nadig, USA

// Please report bugs to:

// https://github.com/karthiknadig/ArduinoFIS/issues

// If you don't have a GitHub account mail to karthiknadig@gmail.com

//*****************************************************************
****



#include "fis_header.h"



// Number of inputs to the fuzzy inference system

const int fis_gcI = 2;

// Number of outputs to the fuzzy inference system

const int fis_gcO = 1;

// Number of rules to the fuzzy inference system

const int fis_gcR = 49;
```

```
FIS_TYPE g_fisInput[fis_gcI];

FIS_TYPE g_fisOutput[fis_gcO];



// Setup routine runs once when you press reset:

void setup()

{

   // initialize the Analog pins for input.

   // Pin mode for Input: Set_Point

   pinMode(0 , INPUT);

   // Pin mode for Input: Current_Position

   pinMode(1 , INPUT);




   // initialize the Analog pins for output.

   // Pin mode for Output: PWM

   pinMode(2 , OUTPUT);



}



// Loop routine runs over and over again forever:
```

```
void loop()

{

   // Read Input: Set_Point

   g_fisInput[0] = analogRead(0);

   // Read Input: Current_Position

   g_fisInput[1] = analogRead(1);



   g_fisOutput[0] = 0;



   fis_evaluate();



   // Set output vlaue: PWM

   analogWrite(2 , g_fisOutput[0]);



}



//*************************************************************************

****

// Support functions for Fuzzy Inference System

//*************************************************************************

****
```

```
// Triangular Member Function

FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)

{

    FIS_TYPE a = p[0], b = p[1], c = p[2];

    FIS_TYPE t1 = (x - a) / (b - a);

    FIS_TYPE t2 = (c - x) / (c - b);

    if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);

    if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));

    if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));

    t1 = min(t1, t2);

    return (FIS_TYPE) max(t1, 0);

}



// Trapezoidal Member Function

FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)

{

    FIS_TYPE a = p[0], b = p[1], c = p[2], d = p[3];

    FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ? 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));

    FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ? 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));

    return (FIS_TYPE) min(t1, t2);
```

```
}


FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
{
   return min(a, b);
}


FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
{
   return max(a, b);
}


FIS_TYPE fis_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp)
{
   int i;
   FIS_TYPE ret = 0;


   if (size == 0) return ret;
   if (size == 1) return array[0];
```

```
    ret = array[0];

    for (i = 1; i < size; i++)

    {

        ret = (*pfnOp)(ret, array[i]);

    }



    return ret;

}




//*****************************************************************************
****

// Data for Fuzzy Inference System

//*****************************************************************************
****

// Pointers to the implementations of member functions

_FIS_MF fis_gMF[] =

{

    fis_trimf, fis_trapmf

};
```

```
// Count of member function for each Input

int fis_gIMFCount[] = { 7, 7 };



// Count of member function for each Output

int fis_gOMFCount[] = { 7 };



// Coefficients for the Input Member Functions

FIS_TYPE fis_gMFI0Coeff1[] = { -30, 0, 30 };

FIS_TYPE fis_gMFI0Coeff2[] = { 0, 30, 60 };

FIS_TYPE fis_gMFI0Coeff3[] = { 30, 60, 90 };

FIS_TYPE fis_gMFI0Coeff4[] = { 60, 90, 120 };

FIS_TYPE fis_gMFI0Coeff5[] = { 90, 120, 150 };

FIS_TYPE fis_gMFI0Coeff6[] = { 120, 150, 180 };

FIS_TYPE fis_gMFI0Coeff7[] = { 150, 180, 210 };

FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1, fis_gMFI0Coeff2,
fis_gMFI0Coeff3, fis_gMFI0Coeff4, fis_gMFI0Coeff5, fis_gMFI0Coeff6,
fis_gMFI0Coeff7 };

FIS_TYPE fis_gMFI1Coeff1[] = { -120, -90, -60 };

FIS_TYPE fis_gMFI1Coeff2[] = { -90, -60, -30 };

FIS_TYPE fis_gMFI1Coeff3[] = { -59.99, -30.01, 0 };

FIS_TYPE fis_gMFI1Coeff4[] = { -30.01, 0, 30.01 };
```

```
FIS_TYPE fis_gMFI1Coeff5[] = { 0, 30.01, 59.99 };

FIS_TYPE fis_gMFI1Coeff6[] = { 30, 60, 90 };

FIS_TYPE fis_gMFI1Coeff7[] = { 60, 90, 120 };

FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1, fis_gMFI1Coeff2, fis_gMFI1Coeff3, fis_gMFI1Coeff4, fis_gMFI1Coeff5, fis_gMFI1Coeff6, fis_gMFI1Coeff7 };

FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff, fis_gMFI1Coeff };



// Coefficients for the Output Member Functions

FIS_TYPE fis_gMFO0Coeff1[] = { -28.33, -14.17, 14.17, 28.33 };

FIS_TYPE fis_gMFO0Coeff2[] = { 14.17, 28.33, 56.67, 70.83 };

FIS_TYPE fis_gMFO0Coeff3[] = { 56.67, 70.83, 99.17, 113.3 };

FIS_TYPE fis_gMFO0Coeff4[] = { 99.17, 113.3, 141.7, 155.8 };

FIS_TYPE fis_gMFO0Coeff5[] = { 141.7, 155.8, 184.2, 198.3 };

FIS_TYPE fis_gMFO0Coeff6[] = { 184.2, 198.3, 226.7, 240.8 };

FIS_TYPE fis_gMFO0Coeff7[] = { 226.7, 240.8, 269.2, 283.3 };

FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3, fis_gMFO0Coeff4, fis_gMFO0Coeff5, fis_gMFO0Coeff6, fis_gMFO0Coeff7 };

FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff };



// Input membership function set
```

```
int fis_gMFI0[] = { 0, 0, 0, 0, 0, 0, 0 };

int fis_gMFI1[] = { 0, 0, 0, 0, 0, 0, 0 };

int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1};



// Output membership function set

int fis_gMFO0[] = { 1, 1, 1, 1, 1, 1, 1 };

int* fis_gMFO[] = { fis_gMFO0};



// Rule Weights

FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };



// Rule Type

int fis_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };



// Rule Inputs

int fis_gRI0[] = { 1, 1 };

int fis_gRI1[] = { 1, 2 };

int fis_gRI2[] = { 1, 3 };

int fis_gRI3[] = { 1, 4 };
```

```
int fis_gRI4[] = { 1, 5 };

int fis_gRI5[] = { 1, 6 };

int fis_gRI6[] = { 1, 7 };

int fis_gRI7[] = { 2, 1 };

int fis_gRI8[] = { 2, 2 };

int fis_gRI9[] = { 2, 3 };

int fis_gRI10[] = { 2, 4 };

int fis_gRI11[] = { 2, 5 };

int fis_gRI12[] = { 2, 6 };

int fis_gRI13[] = { 2, 7 };

int fis_gRI14[] = { 3, 1 };

int fis_gRI15[] = { 3, 2 };

int fis_gRI16[] = { 3, 3 };

int fis_gRI17[] = { 3, 4 };

int fis_gRI18[] = { 3, 5 };

int fis_gRI19[] = { 3, 6 };

int fis_gRI20[] = { 3, 7 };

int fis_gRI21[] = { 4, 1 };

int fis_gRI22[] = { 4, 2 };

int fis_gRI23[] = { 4, 3 };
```

```
int fis_gRI24[] = { 4, 4 };

int fis_gRI25[] = { 4, 5 };

int fis_gRI26[] = { 4, 6 };

int fis_gRI27[] = { 4, 7 };

int fis_gRI28[] = { 5, 1 };

int fis_gRI29[] = { 5, 2 };

int fis_gRI30[] = { 5, 3 };

int fis_gRI31[] = { 5, 4 };

int fis_gRI32[] = { 5, 5 };

int fis_gRI33[] = { 5, 6 };

int fis_gRI34[] = { 5, 7 };

int fis_gRI35[] = { 6, 1 };

int fis_gRI36[] = { 6, 2 };

int fis_gRI37[] = { 6, 3 };

int fis_gRI38[] = { 6, 4 };

int fis_gRI39[] = { 6, 5 };

int fis_gRI40[] = { 6, 6 };

int fis_gRI41[] = { 6, 7 };

int fis_gRI42[] = { 7, 1 };

int fis_gRI43[] = { 7, 2 };
```

```
int fis_gRI44[] = { 7, 3 };

int fis_gRI45[] = { 7, 4 };

int fis_gRI46[] = { 7, 5 };

int fis_gRI47[] = { 7, 6 };

int fis_gRI48[] = { 7, 7 };

int* fis_gRI[] = { fis_gRI0, fis_gRI1, fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5, fis_gRI6, fis_gRI7, fis_gRI8, fis_gRI9, fis_gRI10, fis_gRI11, fis_gRI12, fis_gRI13, fis_gRI14, fis_gRI15, fis_gRI16, fis_gRI17, fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI21, fis_gRI22, fis_gRI23, fis_gRI24, fis_gRI25, fis_gRI26, fis_gRI27, fis_gRI28, fis_gRI29, fis_gRI30, fis_gRI31, fis_gRI32, fis_gRI33, fis_gRI34, fis_gRI35, fis_gRI36, fis_gRI37, fis_gRI38, fis_gRI39, fis_gRI40, fis_gRI41, fis_gRI42, fis_gRI43, fis_gRI44, fis_gRI45, fis_gRI46, fis_gRI47, fis_gRI48 };



// Rule Outputs

int fis_gRO0[] = { 1 };

int fis_gRO1[] = { 1 };

int fis_gRO2[] = { 2 };

int fis_gRO3[] = { 2 };

int fis_gRO4[] = { 3 };

int fis_gRO5[] = { 3 };

int fis_gRO6[] = { 4 };

int fis_gRO7[] = { 1 };
```

```
int fis_gRO8[] = { 2 };

int fis_gRO9[] = { 2 };

int fis_gRO10[] = { 3 };

int fis_gRO11[] = { 3 };

int fis_gRO12[] = { 4 };

int fis_gRO13[] = { 5 };

int fis_gRO14[] = { 2 };

int fis_gRO15[] = { 2 };

int fis_gRO16[] = { 3 };

int fis_gRO17[] = { 3 };

int fis_gRO18[] = { 4 };

int fis_gRO19[] = { 5 };

int fis_gRO20[] = { 5 };

int fis_gRO21[] = { 2 };

int fis_gRO22[] = { 3 };

int fis_gRO23[] = { 3 };

int fis_gRO24[] = { 4 };

int fis_gRO25[] = { 5 };

int fis_gRO26[] = { 5 };

int fis_gRO27[] = { 6 };
```

```
int fis_gRO28[] = { 3 };

int fis_gRO29[] = { 3 };

int fis_gRO30[] = { 4 };

int fis_gRO31[] = { 5 };

int fis_gRO32[] = { 5 };

int fis_gRO33[] = { 6 };

int fis_gRO34[] = { 6 };

int fis_gRO35[] = { 3 };

int fis_gRO36[] = { 4 };

int fis_gRO37[] = { 5 };

int fis_gRO38[] = { 5 };

int fis_gRO39[] = { 5 };

int fis_gRO40[] = { 6 };

int fis_gRO41[] = { 7 };

int fis_gRO42[] = { 4 };

int fis_gRO43[] = { 5 };

int fis_gRO44[] = { 5 };

int fis_gRO45[] = { 6 };

int fis_gRO46[] = { 6 };

int fis_gRO47[] = { 7 };
```

```
int fis_gRO48[] = { 7 };

int* fis_gRO[] = { fis_gRO0, fis_gRO1, fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5,
fis_gRO6, fis_gRO7, fis_gRO8, fis_gRO9, fis_gRO10, fis_gRO11, fis_gRO12,
fis_gRO13, fis_gRO14, fis_gRO15, fis_gRO16, fis_gRO17, fis_gRO18, fis_gRO19,
fis_gRO20, fis_gRO21, fis_gRO22, fis_gRO23, fis_gRO24, fis_gRO25, fis_gRO26,
fis_gRO27, fis_gRO28, fis_gRO29, fis_gRO30, fis_gRO31, fis_gRO32, fis_gRO33,
fis_gRO34, fis_gRO35, fis_gRO36, fis_gRO37, fis_gRO38, fis_gRO39, fis_gRO40,
fis_gRO41, fis_gRO42, fis_gRO43, fis_gRO44, fis_gRO45, fis_gRO46, fis_gRO47,
fis_gRO48 };



// Input range Min

FIS_TYPE fis_gIMin[] = { 0, -90 };



// Input range Max

FIS_TYPE fis_gIMax[] = { 180, 90 };



// Output range Min

FIS_TYPE fis_gOMin[] = { 0 };



// Output range Max

FIS_TYPE fis_gOMax[] = { 255 };
```

```
//*********************************************************************
****

// Data dependent support functions for Fuzzy Inference System

//*********************************************************************
****

FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o)

{

   FIS_TYPE mfOut;

   int r;



   for (r = 0; r < fis_gcR; ++r)

   {

      int index = fis_gRO[r][o];

      if (index > 0)

      {

         index = index - 1;

         mfOut = (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);

      }

      else if (index < 0)

      {

         index = -index - 1;
```

```
            mfOut = 1 - (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut, fuzzyRuleSet[1][r]);
    }
    return fis_array_operation(fuzzyRuleSet[0], fis_gcR, fis_max);
}

FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] - fis_gOMin[o]) / (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;
```

```
    // calculate the area under the curve formed by the MF outputs
    for (i = 0; i < FIS_RESOLUSION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step * fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] + fis_gOMin[o]) / 2) : (momentum / area));
}

//************************************************************************
****
// Fuzzy Inference System
//************************************************************************
****
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0, 0, 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0, 0, 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fis_gcI] = { fuzzyInput0, fuzzyInput1, };
```

```
FIS_TYPE fuzzyOutput0[] = { 0, 0, 0, 0, 0, 0, 0 };

FIS_TYPE* fuzzyOutput[fis_gcO] = { fuzzyOutput0, };

FIS_TYPE fuzzyRules[fis_gcR] = { 0 };

FIS_TYPE fuzzyFires[fis_gcR] = { 0 };

FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };

FIS_TYPE sW = 0;



// Transforming input to fuzzy Input

int i, j, r, o;

for (i = 0; i < fis_gcI; ++i)

{

   for (j = 0; j < fis_gIMFCount[i]; ++j)

   {

      fuzzyInput[i][j] =

         (fis_gMF[fis_gMFI[i][j]])(g_fisInput[i], fis_gMFICoeff[i][j]);

   }

}



int index = 0;

for (r = 0; r < fis_gcR; ++r)
```

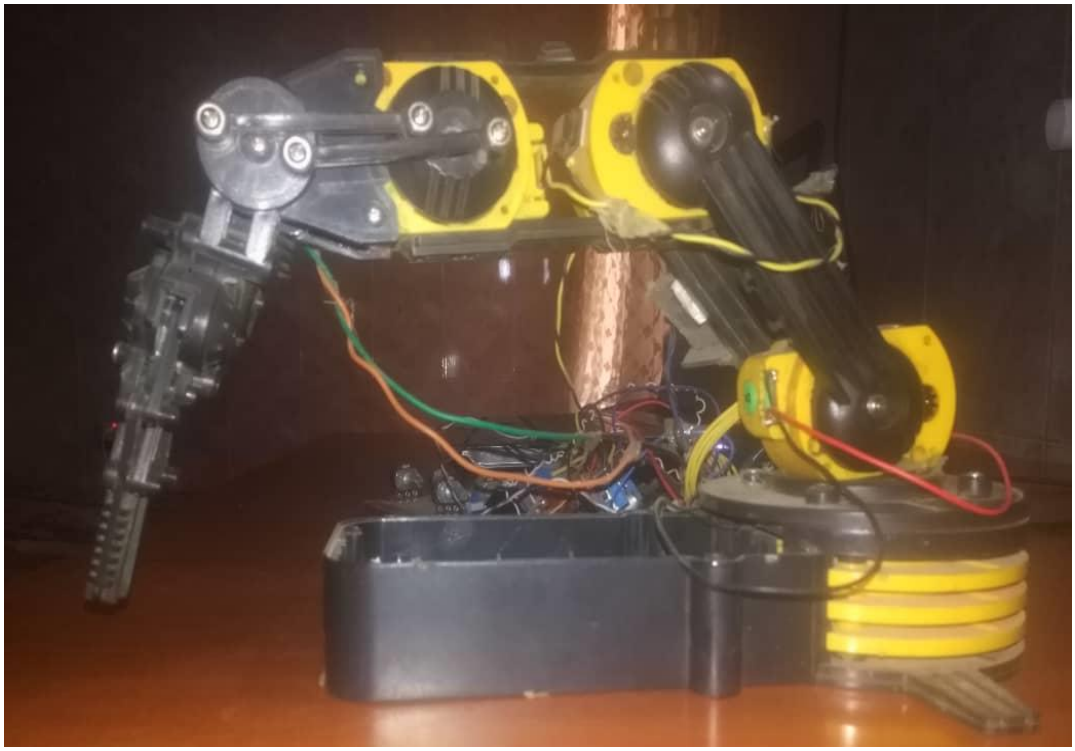
```
{
    if (fis_gRType[r] == 1)
    {
        fuzzyFires[r] = FIS_MAX;
        for (i = 0; i < fis_gcI; ++i)
        {
            index = fis_gRI[r][i];
            if (index > 0)
                fuzzyFires[r] = fis_min(fuzzyFires[r], fuzzyInput[i][index - 1]);
            else if (index < 0)
                fuzzyFires[r] = fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
                fuzzyFires[r] = fis_min(fuzzyFires[r], 1);
        }
    }
    else
    {
        fuzzyFires[r] = FIS_MIN;
        for (i = 0; i < fis_gcI; ++i)
        {
```

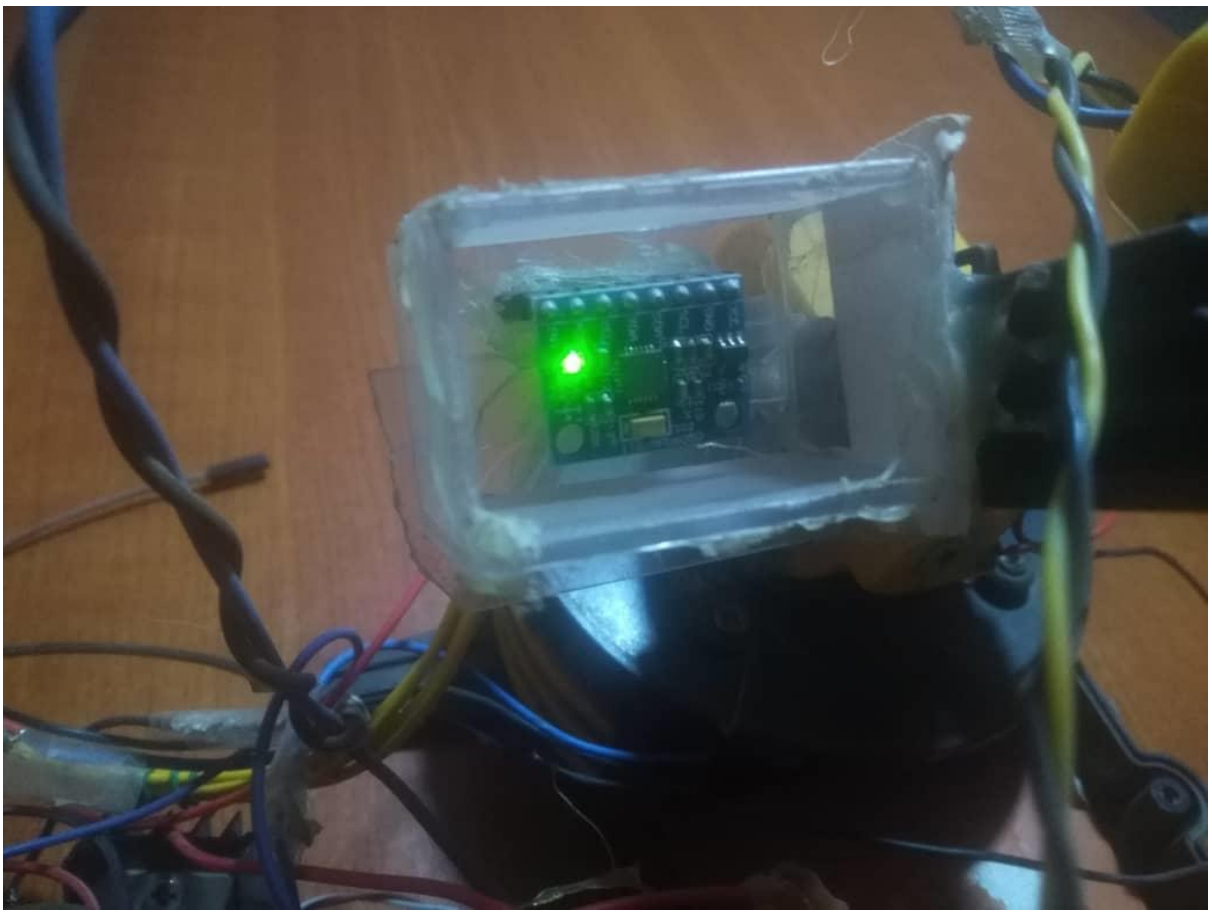
```
            index = fis_gRI[r][i];
            if (index > 0)
                fuzzyFires[r] = fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
            else if (index < 0)
                fuzzyFires[r] = fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
            else
                fuzzyFires[r] = fis_max(fuzzyFires[r], 0);
        }
    }

    fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
    sW += fuzzyFires[r];
}

if (sW == 0)
{
    for (o = 0; o < fis_gcO; ++o)
    {
        g_fisOutput[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
    }
```

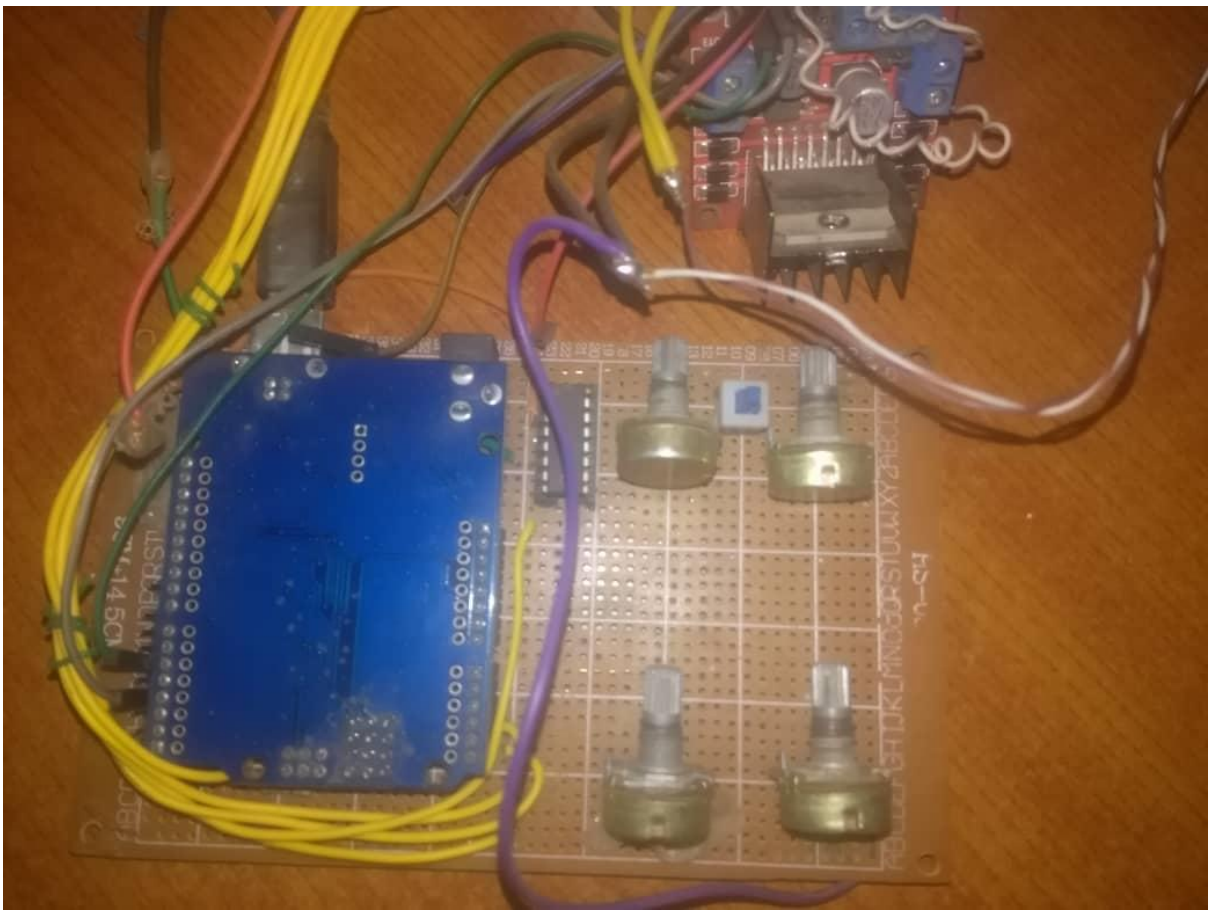
```
    }

    else

    {

        for (o = 0; o < fis_gcO; ++o)

        {

            g_fisOutput[o] = fis_defuzz_centroid(fuzzyRuleSet, o);

        }

    }

}
```

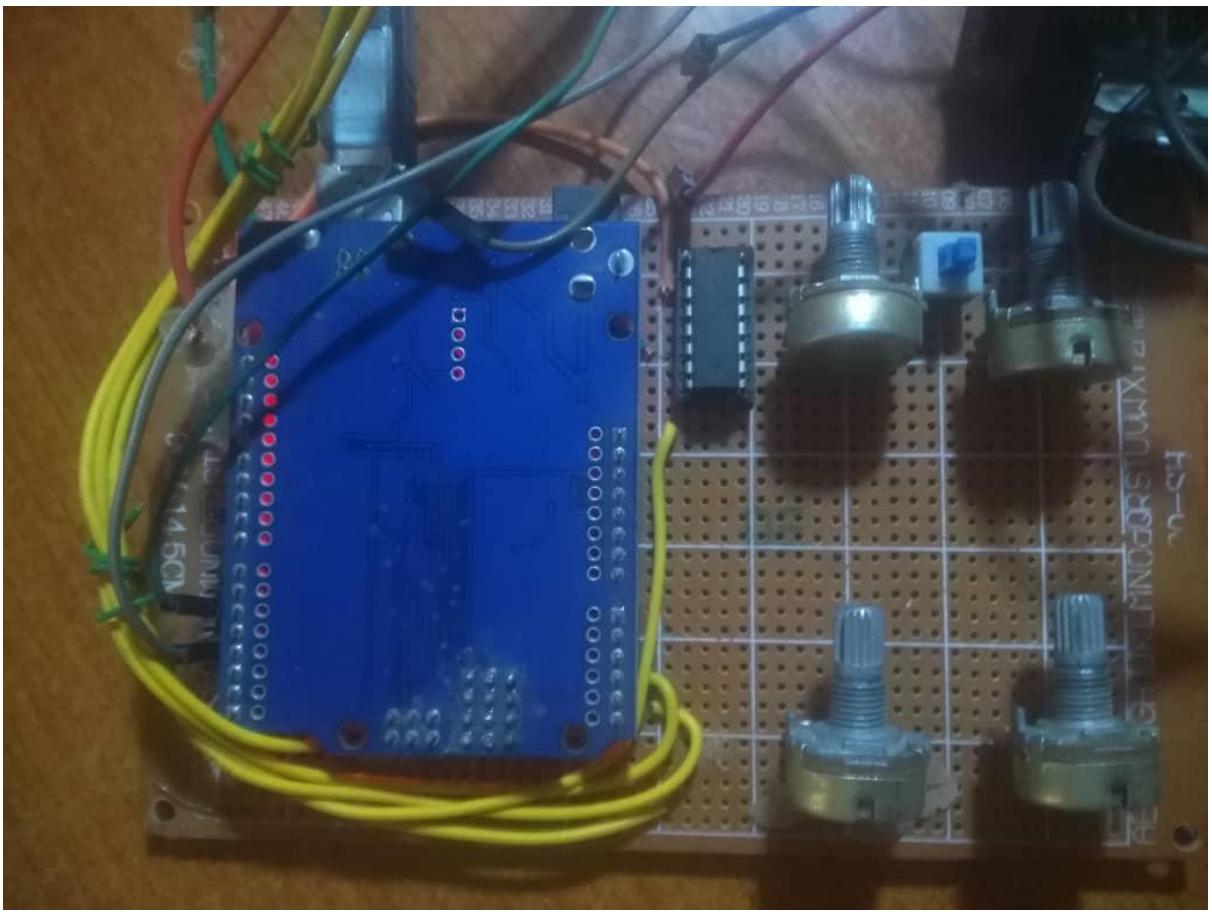
Appendix (C)

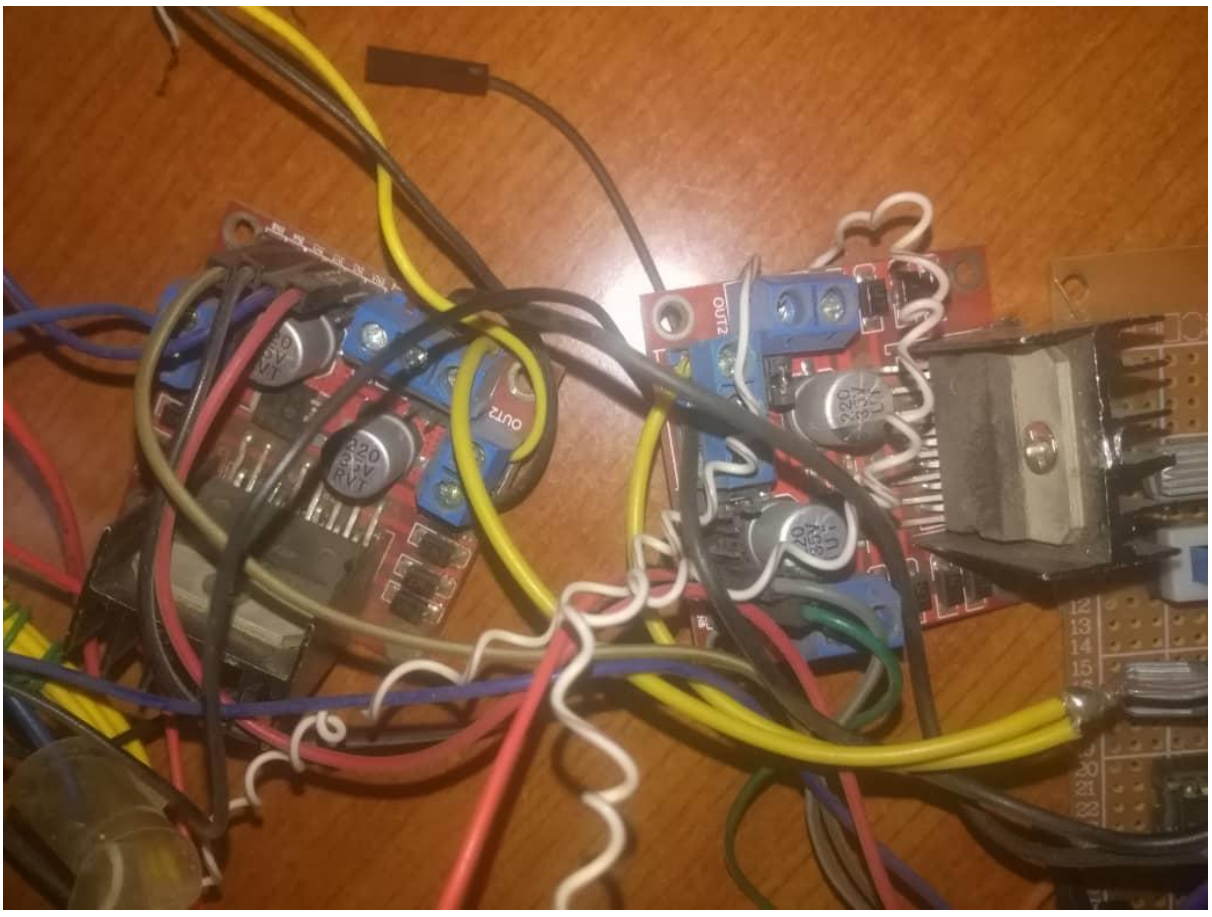
Robot Arm pictures

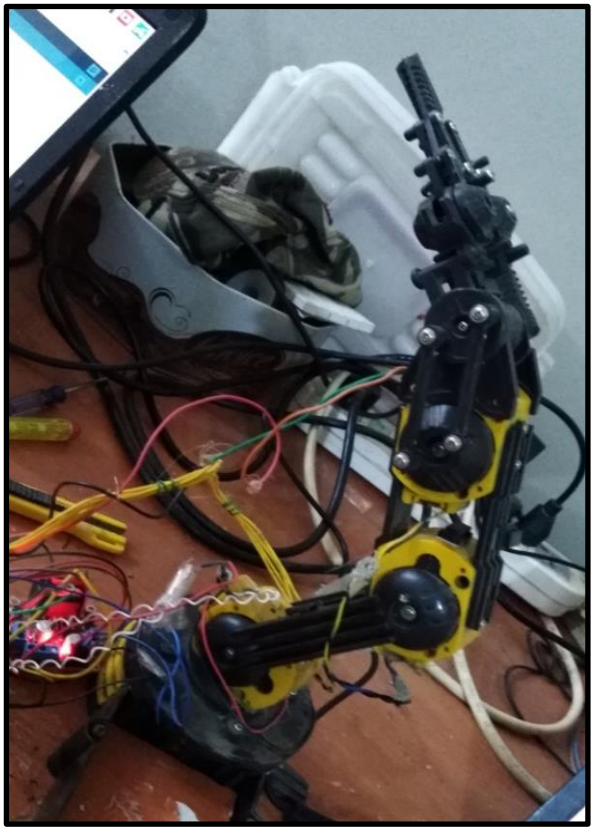


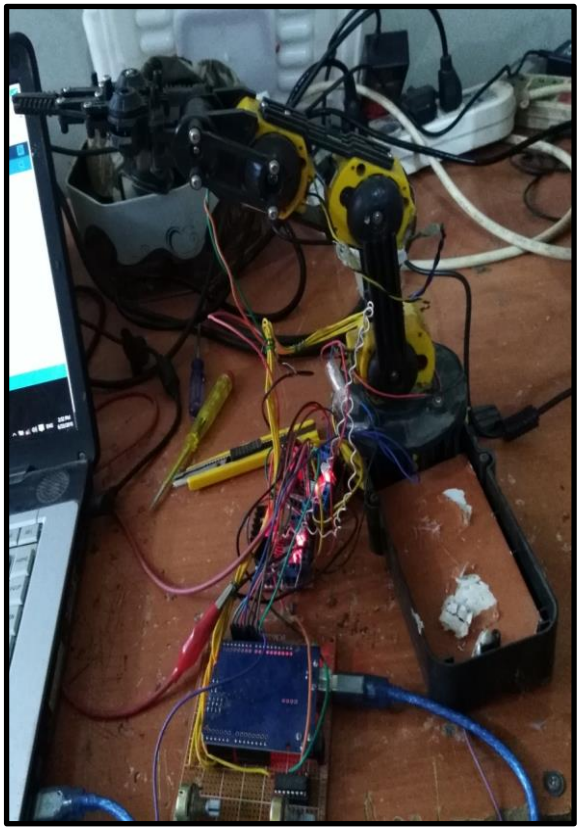







OUT2
OUT2





Appendix (D)

MPU60X0 data sheet

Motion Interface™ is becoming a "must-have" function being adopted by smart phone and tablet manufacturers due to the enormous value it adds to the end user experience. In smart phones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, Motion Tracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for Motion Interface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-60X0 is the world's first integrated 6-axis Motion Tracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis Motion Fusion™ output. The MPU-60X0 Motion Tracking devices, with its 6-axis integration, on-board Motion Fusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I2C port. The MPU-60X0 is footprint compatible with the MPU-30X0 families.

Appendix (E)

Driver L298 data sheet

The L298 is an integrated monolithic circuit in a 15-lead Multi watt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.