Design, Building and Controlling 3DoF Agile Eye Robot

By

Muhannad Kfafi

Anas Abbas

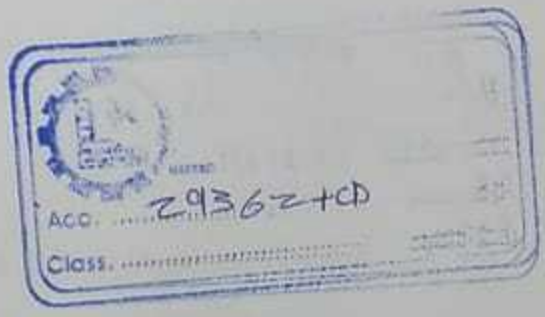
Maryam Tarayrah

Supervisor: Dr. Iyad Hashlamon

Submitted to the College of Engineering

in partial fulfillment of the requirements for the degree of

Bachelor degree in Mechatronics Engineering



Palestine Polytechnic University

DEC 2015

Design, Building and Controlling 3DoF Agile Eye Robot

Project Team

Anas Abbas

Muhannad Kfafi

Maryam Tarayrah

Submitted to the College of Engineering
in partial fulfillment of the requirements for the degree of
Bachelor degree in Mechatronics Engineering

Supervisor Signature

.....

Testing Committee Signature

....

Chair of the Departure Signature

.....

DEC, 2015

Hebron – Palestine

Dedication

"وما توفيقي إلا بالله عليه توكلت وإليه أنيب"

We dedicate this project to our great parents; we could never done this without your faith, support and ceaseless encouragement.

Thank you for all the unconditional love, guidance and support that you have always given us. Thank you for believing in us, thanks for everything.

Acknowledgement

We would like to express our deep gratitude to our supervisor Dr. Iyad F. Hashlamon for his hard work, patience and guidance and for his encouragement throughout the preparation of this project. He inspired us greatly to work in this project. His knowledge and endless support was a great asset from which we learned plenty.

Special thanks to Eng. Ammar Halaiqa and Eng. Ibrahim Hroub for supporting us from the first step of the project.

Finally, we would like to thank the staff of mechanical engineering for helping us in every possible way to complete this project.

Abstract

The Agile Eye robot represents a challenging problem in design and control it to get high speed and acceleration.

The Agile Eye robot is consists of 3 motors, 3 active link, 3 forearm link, 3 passive link, end-effector and base. The motors are fixed on the base with angle 120^{o} between each motor, the motor shaft is connected with active link, the last one is connected with forearm link, and forearm link connected with passive link, finally the passive link is connected with the end-effector.

The controller designed using MATLAB. The aim of this project is educational and it can be used in industry for special uses.

الملخص:

التمثيل الرياضي و التحكم عالي الدقة في الذراع الآلي مغلق الحركه المتوازي "Agile Eye" ثلاثي الابعاد.

اقترحنا في هذه المشروع التمثيل الرياضي الذراع الآلي مغلق الحركه عن طريق التصميم ثلاثي الابعاد باستخدام برنامج "Solidwork".

الروبوت عبارة عن نظام متعدد المداخل ومتعدد المخارج غير خطي ، لذلك فهو يحتاج الى نظام تحكم من اجل تحسين أداء هذا الروبوت . ويتناول ايضا هذا المشروع التحليل الاستاتيكي و الديناميكي . أيضا تم عمل اختبار على التصميم ثلاثي الأبعاد باستخدام برنامج "MATLAB". لضمان الدقة العالية المرجوة من هذا المشروع سيتم تصميم متحكم مناسب لتحقيق هذا الهدف.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

Robots! Robots are everywhere, in hospitals, homes, armies, schools,...etc. Robots are used to discover the deep space, seas and volcanoes. Robots are used in industry making goods and products, saving time and lives. Today robots give us significant effect on many branches of life, from industrial manufacturing to healthcare, transportation, and discover the places that the human can't reach them [1].

Robots are especially desirable for certain work functions because, unlike human, they never get tired, they can endure physical conditions that are uncomfortable or even dangerous, they can operate in airless conditions and they can't be distracted from the task at hand. Robots don't have to look or act like humans but they do need to be flexible so they can perform different tasks [2].

The characteristics that make robots different from regular machinery are that robots usually function by themselves, sensitive to their environment, task oriented and often have the ability to try different methods to accomplish the task [3]. Robots have main components that found in all robots, these component are base, actuators, joints, end-effector and controller.

At 1993 the prototype of agile eye was built, complete dynamic model was then established and high-performance controller based on a DSP was developed. [6]

Solved the direct kinematics, that is to say, finding each possible position and orientation of the mobile platform as a function of the active-joint variables (solved direct kinematics and invers kinematics). [7]

1.1 Serial robot

It also called open-loop manipulator. Serial robot consists of a number of rigid links connected with joints. One end of the robot is attached to the ground and the other end is free to move in space. Every joint need a motor to move it, the fixed link is called base, and the free end where a gripper or a mechanical hand is attached, is called the end effector.

Robots, basically serial robots, are used in applications that require repetitive tasks over long periods of time, operations in hazardous environments (like nuclear radiation, under water, space exploration, etc.), and precision work with high degree of reliability. Some examples of use of industrial robots are following: machine loading and unloading, palletizing, die casting, forging, press work, arc welding and spot welding, heat treatment, spraying (paint, enamel, epoxy resin and other coatings), grinding,

polishing, injection molding, cutting (laser, plasma), inspection, assembly packaging, material handling. Figure 1.1 show the component of serial robot. [4]

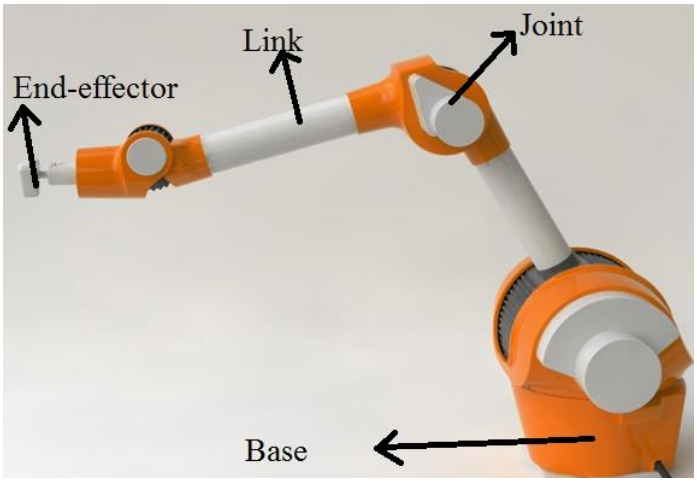


Figure 1.1: Serial robot

1.2 Parallel robot

Parallel manipulators consist of two rigid bodies, one is movable platform and the other is fixed base, connected by a number of kinematic chains. The platform is the end-effector. In each chain, only a few kinematic pairs are actuated while the other pairs are passive joints. The actuation of the chains allows the platform position and orientation to be controlled. All the chains concur to carry the external loads applied to the platform [5].

Parallel structures are more effective than serial ones for industrial automation that require high precision and stiffness, or high load capacity relative to robot weight. Due to their advantages, they have been used in a large number of applications such as astronomy, flight simulators and machine-tool industry.
Parallel manipulators are closed-chain mechanisms with one or more loops where only a certain number of pairs are actively controlled. Fully parallel mechanisms, in particular, feature two rigid bodies, termed base and platform, connected by a set of chains.

Position analysis of a parallel manipulator involves a direct and an inverse kinematic problem.
In general, the inverse problem is trivial, since it asks for the chains angle when the position and orientation of the platform are given with respect to the base. On the contrary, the direct problem, which calls for the position and orientation of the end-effector when the situations of the actively controlled pairs are given, it is a difficult problem for which no general procedure has been found yet and for which closed-form solutions are only available for certain architectures, sometimes satisfying a number of geometric conditions [3].

There are two main cases of parallel manipulator planar manipulator and spatial manipulator, Figure 1.2: spatial manipulator is an example of planar manipulator.

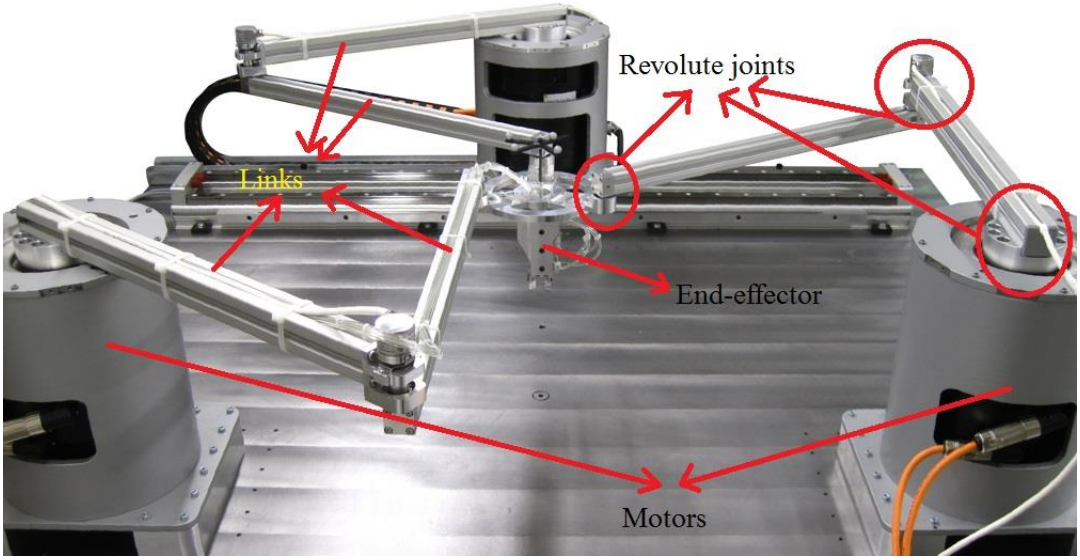


Figure 1.2: spatial manipulator

As shown in Figure 1.3, a 3-DOF spherical Parallel Manipulator is an example of spatial manipulator, it used in many applications, and its end-effector can be attached to laser, camera and can be used for welding and other application [3].

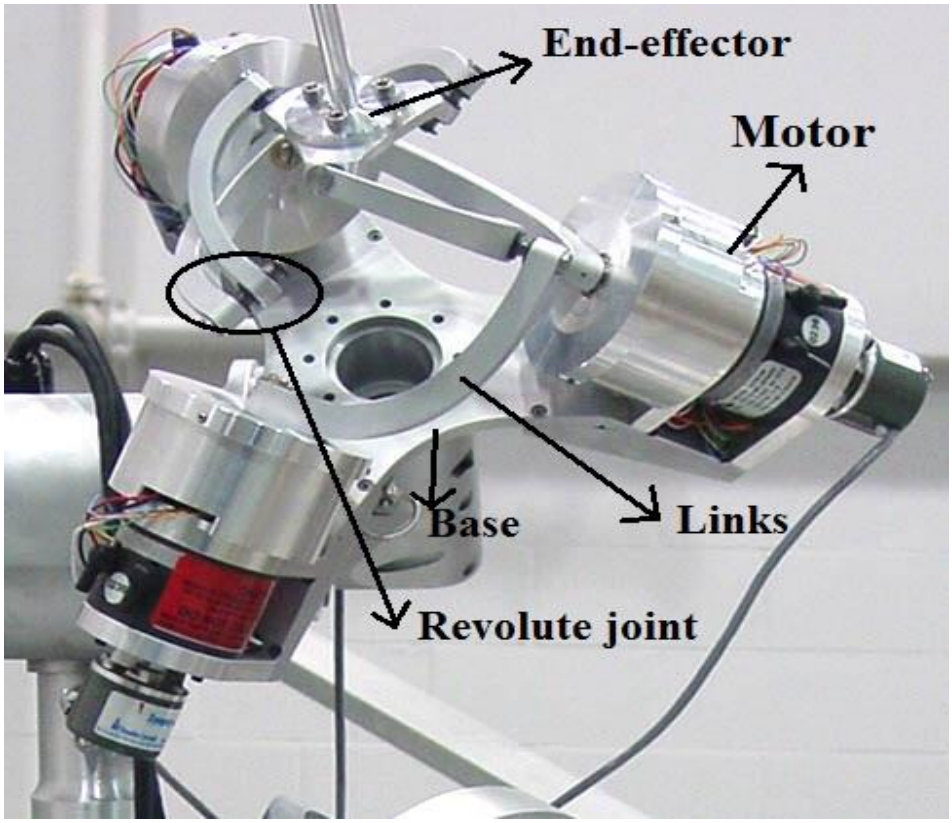


Figure 1.3: 3-DOF spherical parallel manipulator

1.3 Motivation

In this project the design and building of the Agile Eye robot was complete, controller and run the project was done. The project can be used in educational, industry and other application. The project can give high speed, acceleration and accuracy, different tools can be objected on the end-effector.

1.4 Problem definition

There are two problems, the first is that the angle of motor shaft are known and need to determine the orientation of the end-effector, it's known as forward kinematics. The second one the orientation of the end-effector is known and need to determine the angle of motor shaft, this problem is known as inverse kinematics.

1.5 Problem solution

To solve the previous problem we need two functions one for forward kinematics and other one for inverse kinematics.

1.6 Time table

- Stage 1: Select the idea
- Stage 2: Preparing for the project and collecting data
- Stage 3: Project Analysis

  In this step we analyzed the data that we collected then study the possible design options that we have in order to decide the best design.
- Stage 4: Determine the project requirement

  After choosing the best design we determine the detailed mathematical model for the system.
- Stage 5: Documentation the project

  Documenting the project will begin from the first stage to the last stage.

For next semester

- Stage 6: make the hardware available

  In this stage, the needed hardware devices will be brought for the next steps, motors, switches, sensors, belt, shaft rolls, gears, and speed reducers.
- Stage 7: build up the machine and finishing

  All the project equipment's and devices will be bought if there is an available source in the market, then go to lathe to prepare the mechanical parts.
- Stage 8: testing the machine

  Detect if there is an error occurred and making a report about that.
- Stage 9: finishing the graduation final report

  All documentation has made is to be checked and done in this stage. Every change in it is to be added and to be noticed that something is changed.
- Stage 10: Preparing for the final presentation.

  The presentation will be prepared to show our work.

Table1: Time table for first semester

Week / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1															
S2															
S3															
S4															
S5															

Table 2: Time table for second semester

Week / Task	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
S6																	
S7																	
S8																	
S9																	
S10																	

Chapter 2

Agile Eye

This chapter introduce the definition, geometric of Agile Eye robot, and uses of Agile Eye are mentioned here.

2.1 Definition

The Agile Eye is a 3-DOF 3-RRR spherical parallel manipulator developed for the rapid orientation of an end-effector. The work space of the Agile Eye is superior to that the human eye. Moreover, due to its low inertia and its inherent stiffness, it has high velocity and accelerations, which beyond the capabilities of the human eye, Figure 2.1 show the mechanical component of Agile Eye.

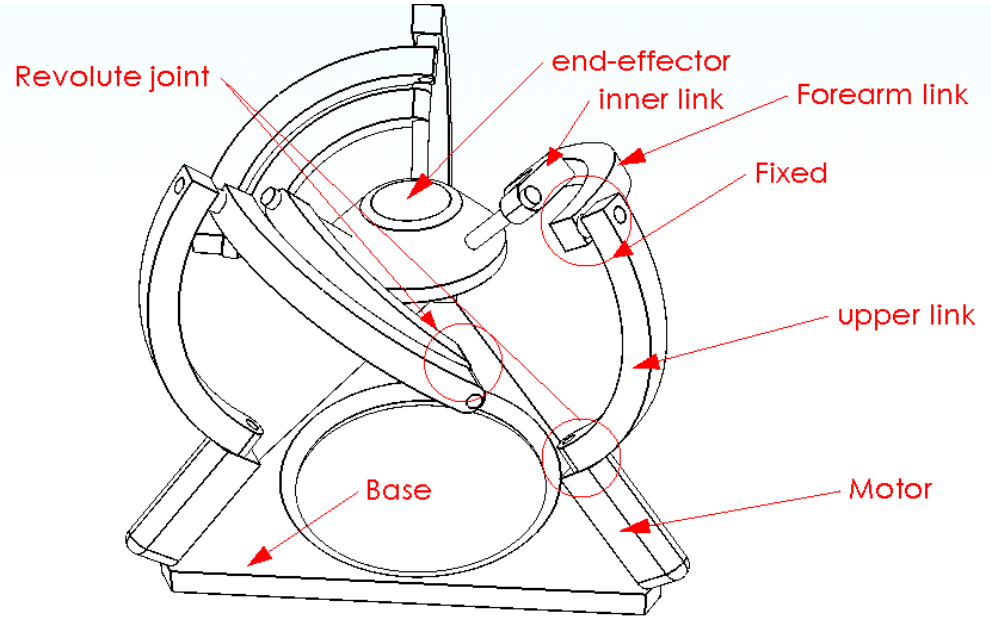


Figure 2.1: CAD model of Agile Eye robot

The input mechanism of the robot is structured into three active revolute joints while the output body is connected through a set of chains with identical geometry, the robot have many advantages that are high speed, high acceleration, high stiffness, low inertia, and high load capacity. The motion of the Agile Eye is a spherical motion, this motion is due to its structure, links, and revolute joints. Each link is a curved link and this property reduces the inertia of the robot.

The Agile Eye have many applications for example the end-effector hold a camera, laser, or other things that need high speed and high acceleration.

As shown Figure 2.2 in the Agile Eye consists of three identically links which is symmetrically ordered around the center of rotation in 120º.

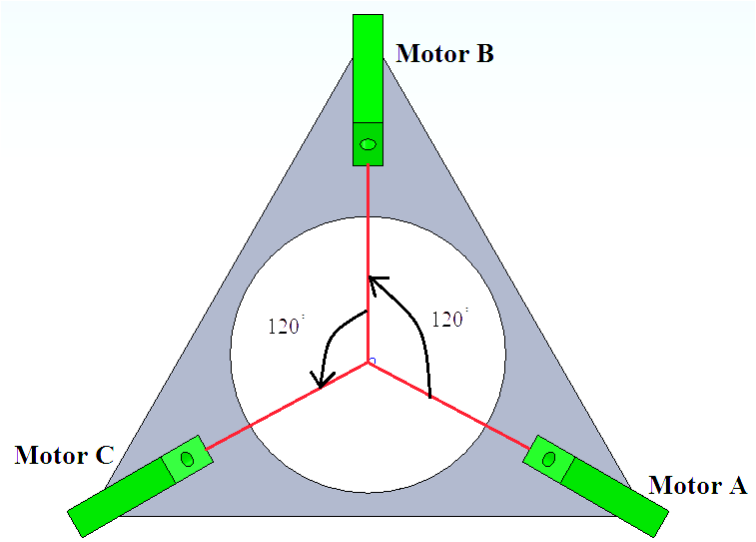


Figure 2.2: The angles between motors

Each link of Agile Eye robot consists of angular bipartite outer link and a passive inner link. The bipartite outer link which comprises an upper and forearm, is coupled at the upper link with actuator rigidly and at the forearm rotatably with the inner link. All inner links are connected with end-effector using revolute joints, all partial links are curved links. The two parts of the outer link are connected rigidly together, as shown in Figure 2.3.

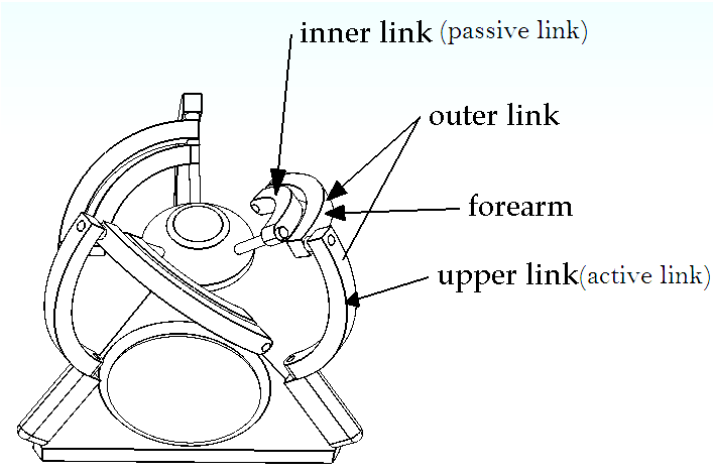


Figure 2.3: Links of Agile Eye

2.2 Geometry

The link angles of the manipulator are assumed to be the same for each of the chains rmaconnecting the end-effector to the base, these angles are β for upper link and fore link. The angle γ is the angle of inner link, R1, R2 and R3 are the radius of upper, .forearm and inner link respectively. These angles and radiuses is shown in Figure 2.4

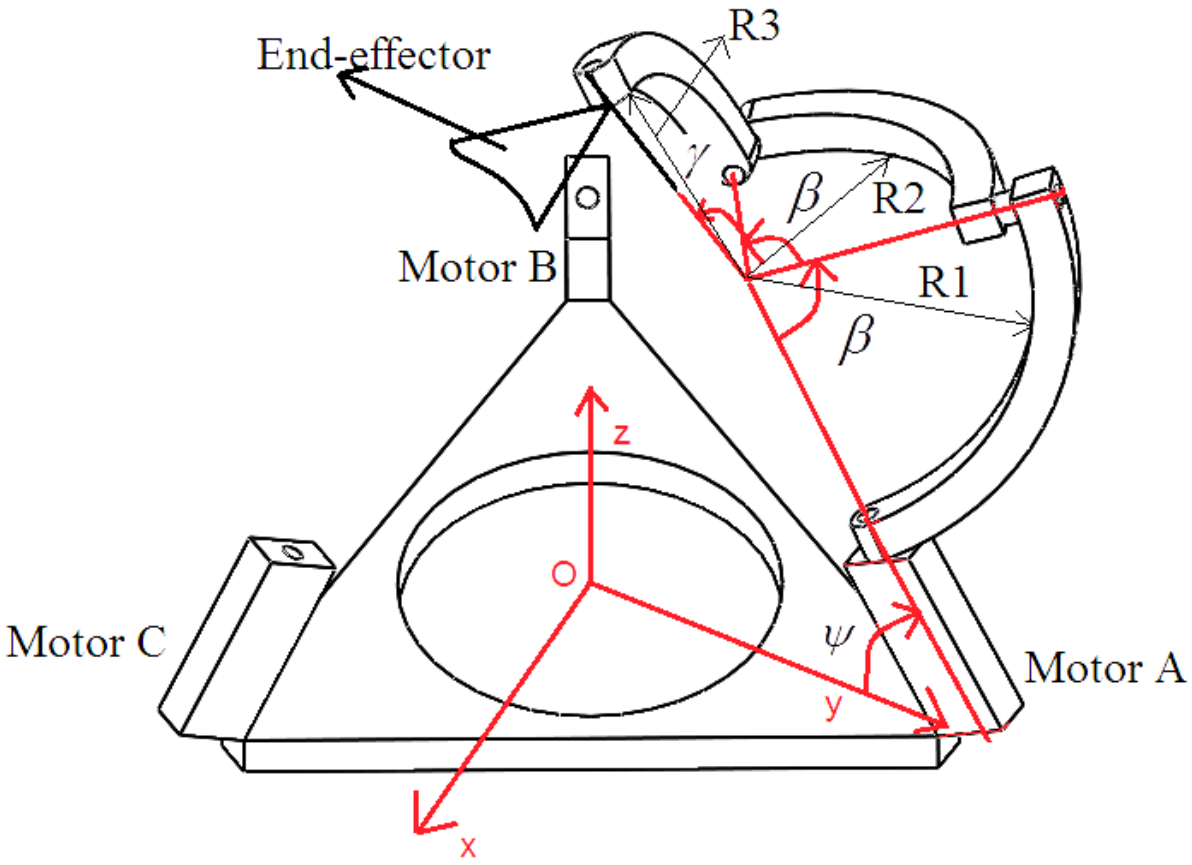


Figure 2.4 Geometric of Agile Eye

Each link of the Agile Eye has 3 chains. Each chain has 3 revolute joints connected serially. The serial chain is called orthogonal, i.e., it's neighboring revolute axes layout at $\beta = \pi / 2$, $\gamma = \pi / 2$, $\psi = \pi / 3$.

Chapter 3

Kinematics Model

3.1 Introduction

In kinematics of manipulators, we study the motion of the links without considering the forces and torques which cause the motion of the links. The robot kinematics can be divided into forward kinematics and inverse kinematics. Forward kinematics is to compute the orientation of the end effector as a function of the joint variables. [9]

Inverse kinematics specifies the end-effector location and computes the associated joint angles or in other word we have orientation position of the end effector so we use it to find angles of three motors.

The relationship between forward and inverse kinematics is illustrated in Figure 3.1.

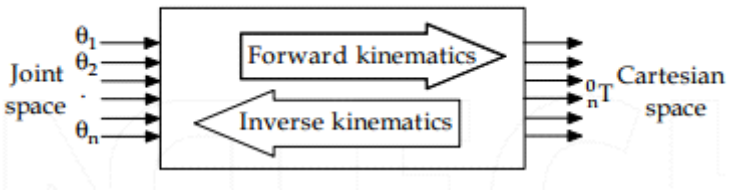


Figure 3.1: relation between inverse and forward kinematics

In general, the forward kinematics (direct) is more complex than inverse kinematics for parallel robot.

3.2 Forward Kinematics

Forward kinematics specifies the joint parameters and computes the configuration of the chain, the aim of direct kinematics is to compute the orientation of the end effector as a function of the joint variables.
The direct kinematics of the *Agile Eye* was solved in [7] and will be reformulated and further analyzed here.
The following constraint equations are written:

$$\sin\psi(\sin\theta_1\sin\phi-\cos\theta\cos\theta_1)+\cos\psi\sin\theta_1\cos\phi=0 \qquad (3.1)$$

$$\cos\psi(\cos\theta_2\sin\theta\cos\phi-\cos\theta\sin\theta_2)+\sin\psi\cos\theta_2\sin\phi=0 \qquad (3.2)$$

$$\sin(\theta_3-\phi)\cos\theta=0 \qquad (3.3)$$

Where θ_i is the motors angle, i=1, 2, 3
And (θ,ϕ,ψ) is the Euler angles.

From equation (3.3) the direct kinematic problem is found to admit two sets of solutions, defined by

$$\cos\theta = 0 \tag{3.4}$$

And

$$\sin(\theta_3 - \phi) = 0 \tag{3.5}$$

A. The first set of solution is trivial solution

Equation (3.4) gives two solution for angle θ:

$$\theta = \pi/2 \quad , \ \theta = -\pi/2$$

From the first solution and after simplification of equation (3.2), the following ,condition is found for arbitrary active joint variables

$$\cos(\phi - \psi) = 0 \tag{3.6}$$

And from the second solution

$$\cos(\phi + \psi) \tag{3.7}$$

These four solution are the trivial solutions to the direct kinematics problem of the Agile Eye robot.

B. The second solution is non-trivial solution

Equation (3.5) gives two solutions for angle ϕ

$$\phi = \theta_3$$

And

$$\phi = \theta_3 \pm \pi$$

Then to find the other angle substitute these two solutions in equation (3.2) and (3.3)

3.3 Inverse Kinematics

To find the set of joint angles that produce a specific end orientation. Inverse kinematics specifies the end-effector orientation and computes the associated joint angles. For parallel manipulators, the specification of the end-effector orientation simplifies the kinematics equations, which yields formulas for the joint parameters.

$$\tan\theta_1 = \frac{\cos\theta\sin\psi}{\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi} \tag{3.8}$$

$$\tan\theta_2 = \frac{\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi}{\cos\theta\cos\psi} \tag{3.9}$$

$$\tan\theta_3 = \tan\phi \tag{3.10}$$

Chapter 4
Simulation

CAD modeling programs can help to explore the shape of components and the interactions of multiple components in assemblies. Another category of software enables us to model dynamics, including applications such as MATLAB and its companion system modeling package Simulink and Working Model.

4.1 Connect Solidwork with MATLAB

In this chapter we discuss how to connect Solidwork with MATLAB through VRML Simulink, which shows that the system works fine.

The first thing that we need is to design the CAD model on Solidwork software as shown in Figure 4.1.

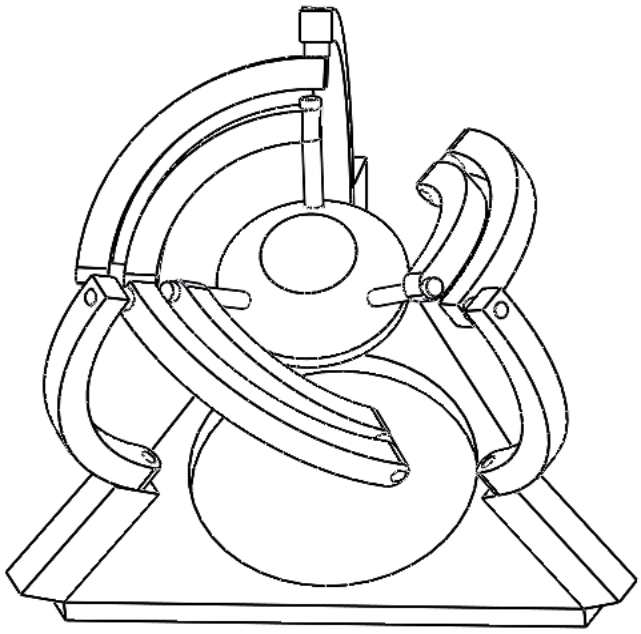


Figure 4.1: CAD model

And save a file in term of (file name. wrl), see Figure 4.2.

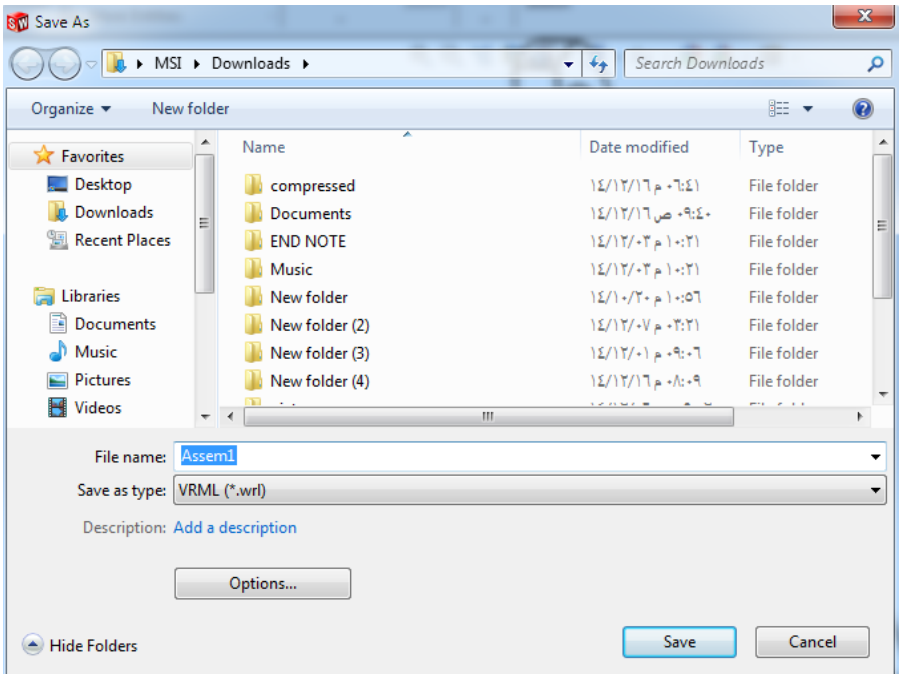


Figure 4.2: form of the file saved from solid work

The second thing is open MATLAB software then the Simulink icon, when we open this icon a libraries of Simulink will open, from this libraries we choose Simulink 3D Animation library. Then choose VR Sink, see Figure 4.3.

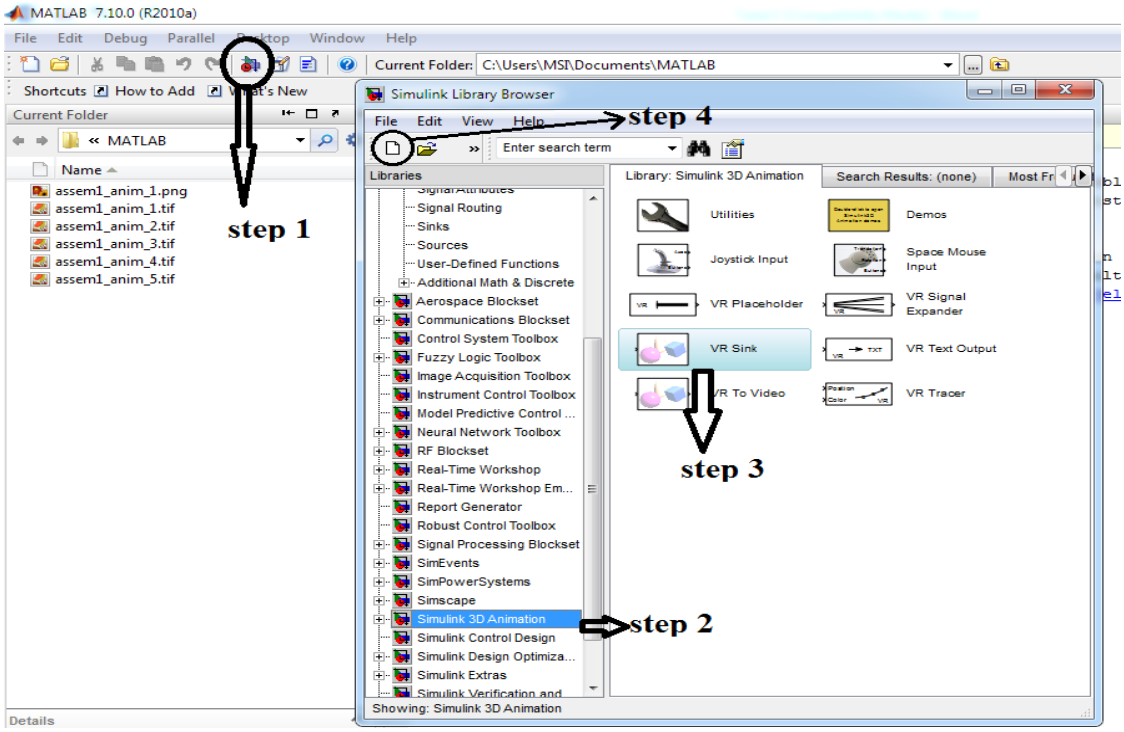


Figure 4.3: steps to open CAD model in matlab

After that we open new model and put the VR Sink to this model as shown in Figure 4.4.

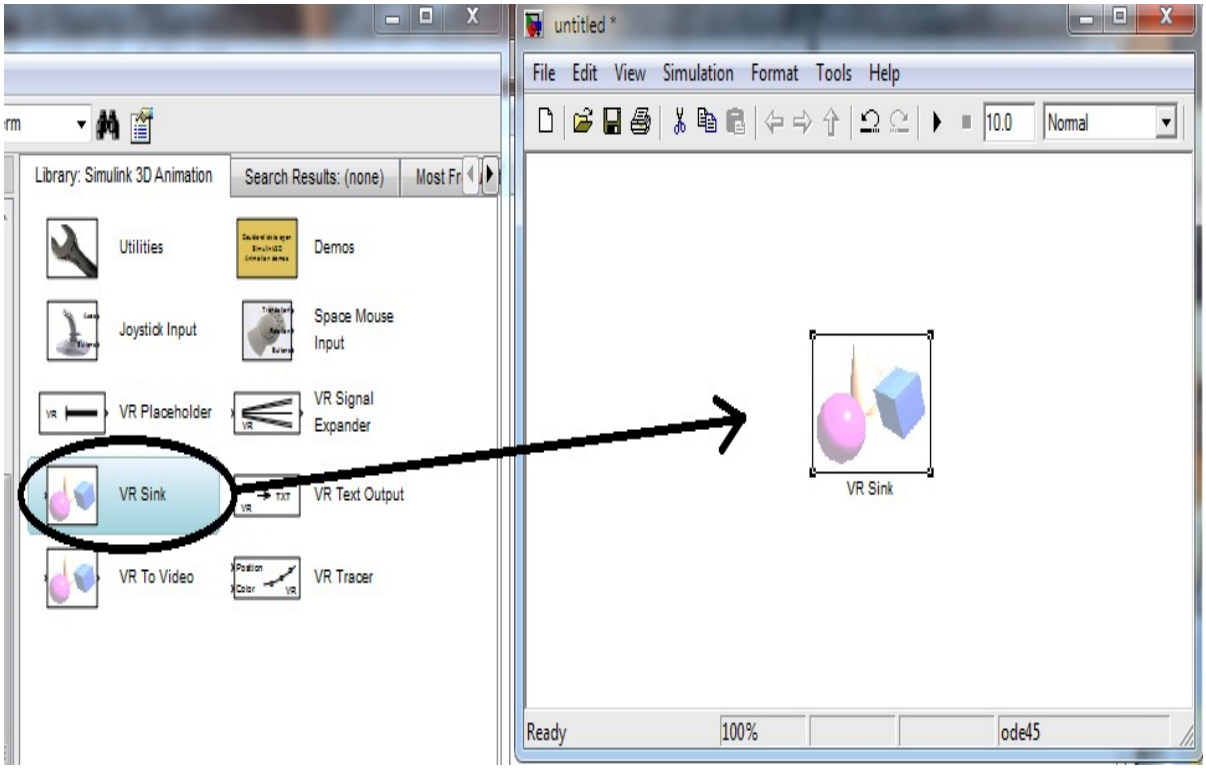


Figure 4.4: How to open VR Sink file

By double click on the block VR Sink a new window will open, from this window, we click on open new file a V-Realm Builder software will open, from this software we open the wrl file that published from Solidwork and save it in wrl form again, then return to the Browser icon and click on it and choose the wrl file that we published from V-Realm Builder, then click ok, see Figure 4.5.

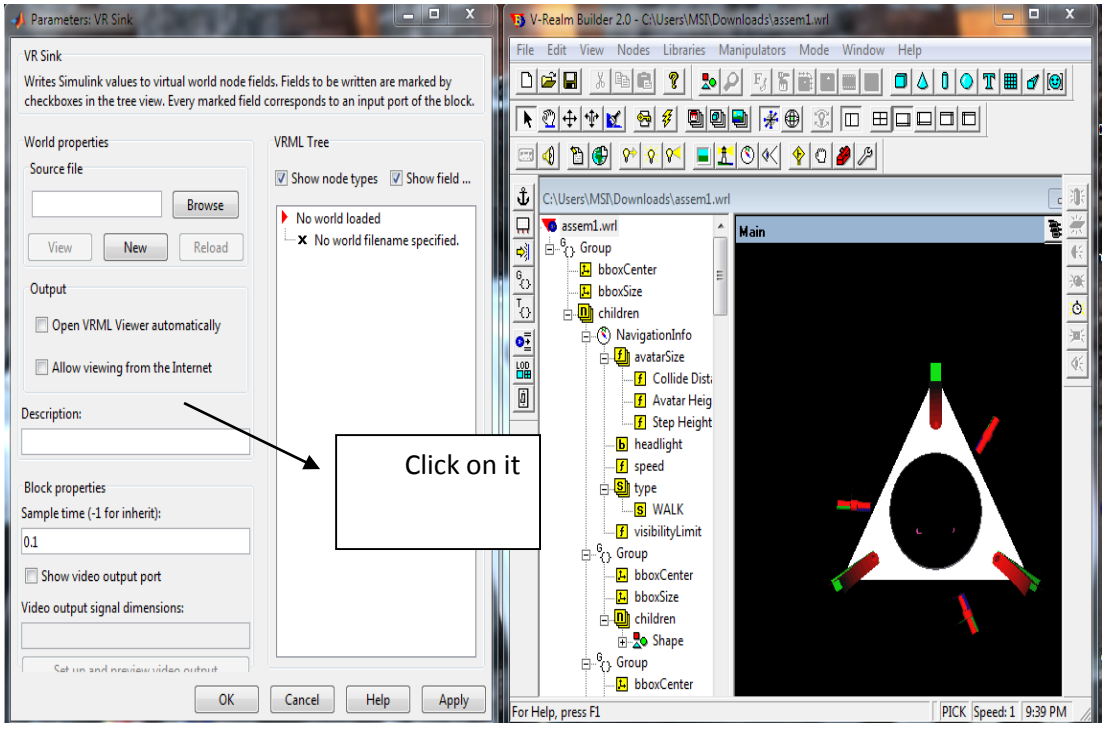


Figure 4.5: How to open V-Realm Builder

After this step return to the VR Sink block and double click on it, a new window will open as shown in Figure 4.6.

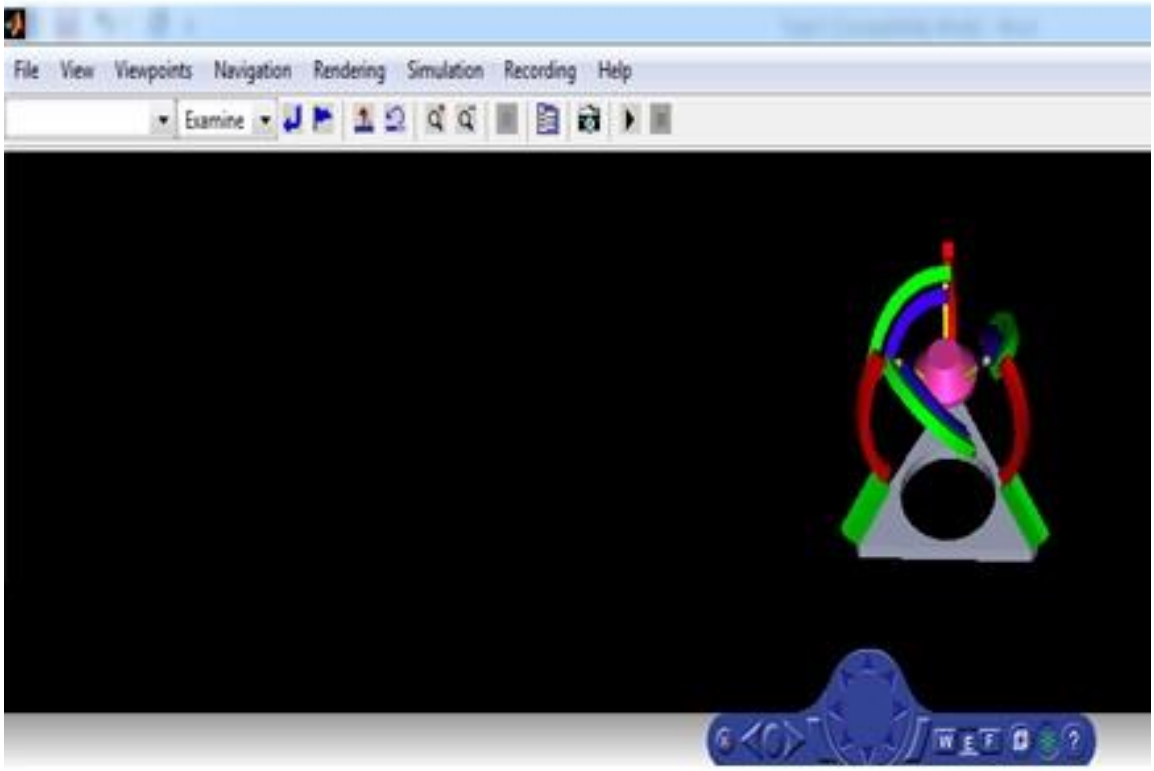


Figure 4.6: The CAD model in MATLAB Simulink

Chapter 5

Component Selection

5.1 Introduction

After doing the necessary calculations and analysis on the project requirements and outputs, we select the project components.

This chapter describes the selected components which are used in this project the three DC servo actuators, Arduino, motor drivers, position sensor , encoder, 3-axis accelerometer with gyro , bearing, and power supply.

5.2 Electrical component

5.2.1 DC-servo actuator:

Its rotary actuator that allows for precise control of angular position, velocity and acceleration it consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system, see Figure 5.1.

Servomotors are used in applications such as Robots, CNC machinery or automated manufacturing.

Specification of servo motors:

1- The relation between speed and voltage must be linear, in order to simplify the control system, and improve the efficiency.

2- Fast response.

Agile Eye robot need three actuators to make the links and end effector move to the given position , we choose this type of motor because its characteristics and properties that give the best results than other type and it's properties are zero backlash, high positional accuracy and high stiffness, the encoder and controller of a servomotor are an additional cost, but they optimize the performance of the overall system (for all of speed, power and accuracy) relative to the capacity of the basic motor. With larger systems, where a powerful motor represents an increasing proportion of the system cost, servomotors have the advantage. Table 5.1 show the specification of the servo motor.

Figure 5.1 DC servo motor

Specification of servo motor:

Required motors:

Table 5.1 specifications of required motor

Rated Output Power [W]	18.5
Rated Voltage [V]	24
Rated Current [A]	1.8
Rated Output Torque [Nm]	5.9
Rated Output Speed [rpm]	30
Peak Current [A]	4.1
Maximum Output Torque [Nm]	20
Maximum Output Speed [rpm]	50
Torque Constant [Nm/A]	5.76
Voltage Constant (B.E.M.F.) [V/rpm]	0.60
Inertia at Output Shaft [Kgm^2]	$81.6*10^{-3}$
Mechanical Time Constant [ms]	7.0
Viscous Damping Constant [Nm/rpm]	$1.5 \cdot 10^{-1}$
Gear Ratio	100

Motor Rated Output [W]	30
Motor Rated Speed [rpm]	3000
Armature Resistance [Ω]	2.7
Armature Inductance [mH]	1.1
Electrical Time Constant [ms]	0.41
Starting Current[A]	0.43
No-Load Running Current[A]	0.91

Selection Procedure:

Requirements for Preliminary Selection:
Load Torque T_L [Nm] < Rated Torque T_N [Nm]
Load Speed n_L [rpm] < Rated Output Speed n_N [rpm]
Load Inertia J_L [kgm^2] < 3 J_A (Actuator Inertia) acceptable
Load Inertia J_L [kgm^2] < J_A (Actuator Inertia) for best Dynamic response.

Determination of the acceleration torque T_1 [Nm] :

$$T_1 = T_L + \frac{2\pi}{60} * \frac{(J_A + J_L) * n_L}{t_1}$$

- Acceleration Torque T_1 < Maximum Output Torque T_m.

Determination of the average torque T_A [Nm] :

$$T_A = \sqrt{\frac{T_1^2 * t_1 + T_2^2 * t_2 + T_3^2 * t_3}{t_1 + t_2 + t_3 + t_4}}$$

Where:
T_1: Acceleration Torque.
$T_2 = T_L$: Load Torque.
T_3: Braking Torque
$T_3 = T_2 - (T_1 - T_2)$ (if $t_1 = t_3$).
t_1: Acceleration Time.
t_2: Constant Speed Time.
t_3: Braking Time.
t_4: Idle Time.

- Average Torque T_A < Rated Torque T_N of the actuator.

Load and Operating Conditions:

Load Torque T_L = 5 Nm.
Load Speed n_L = 30 rpm.
Load Inertia J_L = 0.15 kgm^2.
Acceleration Time t_1 = 0.1 s.
Constant Speed Time t_2 = 0.2 s.
Braking Time t_3 = 0.1 s.
Idle Time t_4 = 0.6 s.

Actuator's Data:
T_N = 5.9 Nm
n_N = 30 rpm
J_A = 0.0816 kgm^2
T_m = 20 Nm

- The above procedure leads to the following selection Actuator RH - 14D - 3002 – E100AL

Available motors**:**

Table 5.2 specification of available motor

Reduction ratio	30.9:1
Maximum continuous torque(N·m)	0.431
No-Load speed (rad/s)	15.3
Peak Torque (N·m)	1.187
Torque constant(N·m/A)	0.044
Back EMFConstant(V/rad/s)	0.044
Resistance(Ω)	17
Inductance(mH)	9.35
Rated voltage(V)	24
Encoder	512 CPR

5.2.2 Motors Driver (H Bridge):

An H bridge is an electrical circuit that enables a voltage to be applied across a load in either direction, see Figure 5.3.

These circuits are often used in robotics and other applications to allow DC motors to run forwards and backwards, see Figure 5.2.

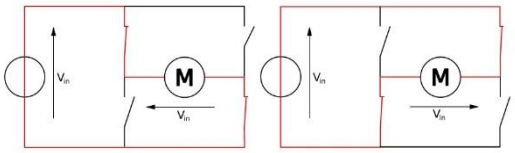


Figure 5.2 voltage direction in H-Bridge

Agile Eye robot need two H-bridge to control the direction and the operating of the three motors.

Motors Driver has the following characteristics:

1) Strong driving ability-low calorific value and strong anti-interference ability.
2) Logical voltage 5V. Drive voltage 5V-35V.
3) Drive current 2A (MAX single bridge), maximum power 25W
4) Large capacity filter capacitance, after flow protection diode, more stable and reliable.

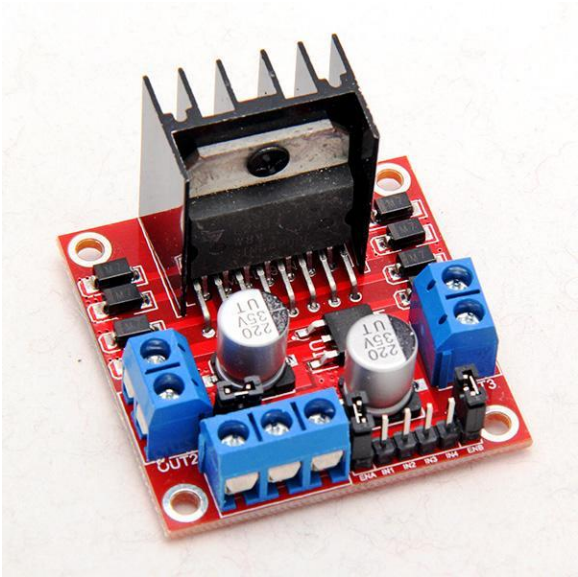


Figure 5.3 L298N Motor Driver Module3

5.2.3 Microcontroller

There were many choices of microcontroller to use, like PIC micro controller from microchip or Atmel microcontrollers, but in this project Arduino board based on Atmel is used. This is because it has some specifications not available on other micro controllers .For example PIC18F4550 has one Pulse with Modulation, but Arduino have 15 Pulse with Modulation. Also it's open source and one can deal with it easily, thus the Arduino Mega 2560 has been chosen to control the Agile Eye Robot, as shown in Figure 5.4.

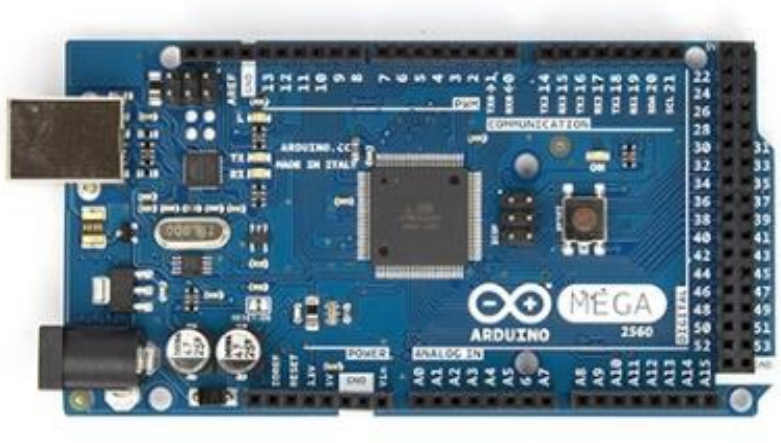


Figure 5.4 Arduino Mega 2560

The Arduino Mega 2560 has 54 digital input/output pins (of which 15 can be used as pulse with modulation outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, 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 to program and start using Arduino.

For the software environment, see Figure 5.5, the Arduino 1.0.5-r2 software is used to verify and download the code on Arduino board.

Figure5.5 Arduino 1.0.6-r2 software

5.2.4 Incremental rotary encoder

Is an electro –mechanical device that converts the angular position or motion of a shaft or axle to an analog or digital code.

There are two main types: absolute and incremental (relative), as shown in Figure 5.6. The output of absolute encoders indicates the current position of the shaft, making them angle transducers. The output of incremental encoders provides information about the motion of the shaft, which is typically further processed elsewhere into information such as speed, distance, and position.

Rotary encoders are used in many applications that require precise shaft unlimited rotation including industrial controls, robotics.

Servo motor use incremental encoders, the incremental encoder usually gives two types of squared waves out of phase for 90 electrical degrees. They are usually called channel A and B. The first channel gives information about the rotation speed while the second, basing on the states sequence produced by the two signals, provides the sense of rotation.

The incremental encoder precision depends on mechanical and electrical factors.

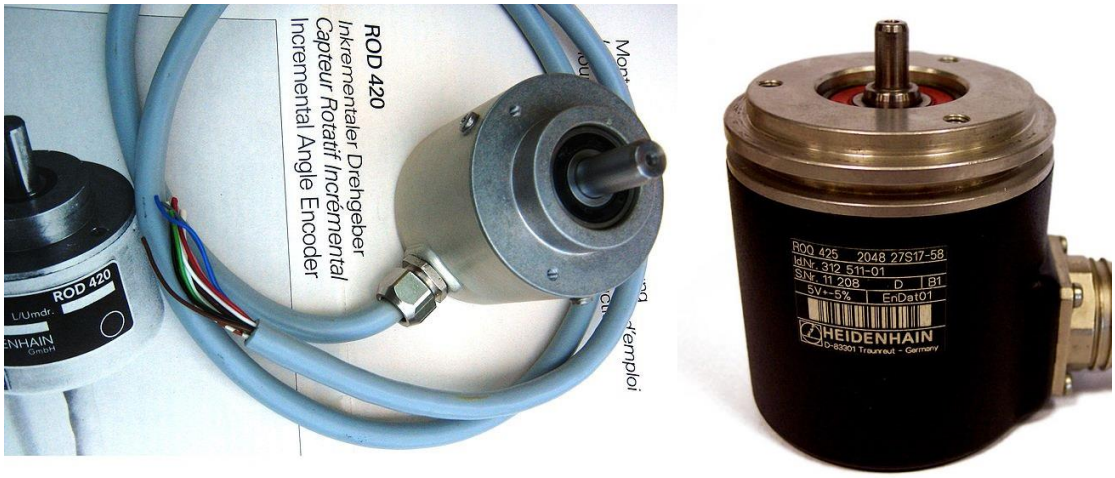


Figure 5.6 Incremental and Absolute rotary encoder

5.2.5 Power Supply

Power supply used to supply H-bridge 12V DC, as shown in Figure 5.7.

Figure 5.7 Power Supply

5.3 Mechanical component

There are many choices to use, like wood , echelon & aluminum , but in this project Aluminum used to build our prototype , we choose the aluminum because it has specifications which are not available in other material like : Light , relatively strong , workability , and soft.

The Figure 5.8 show the Agile Eye robot prototype.

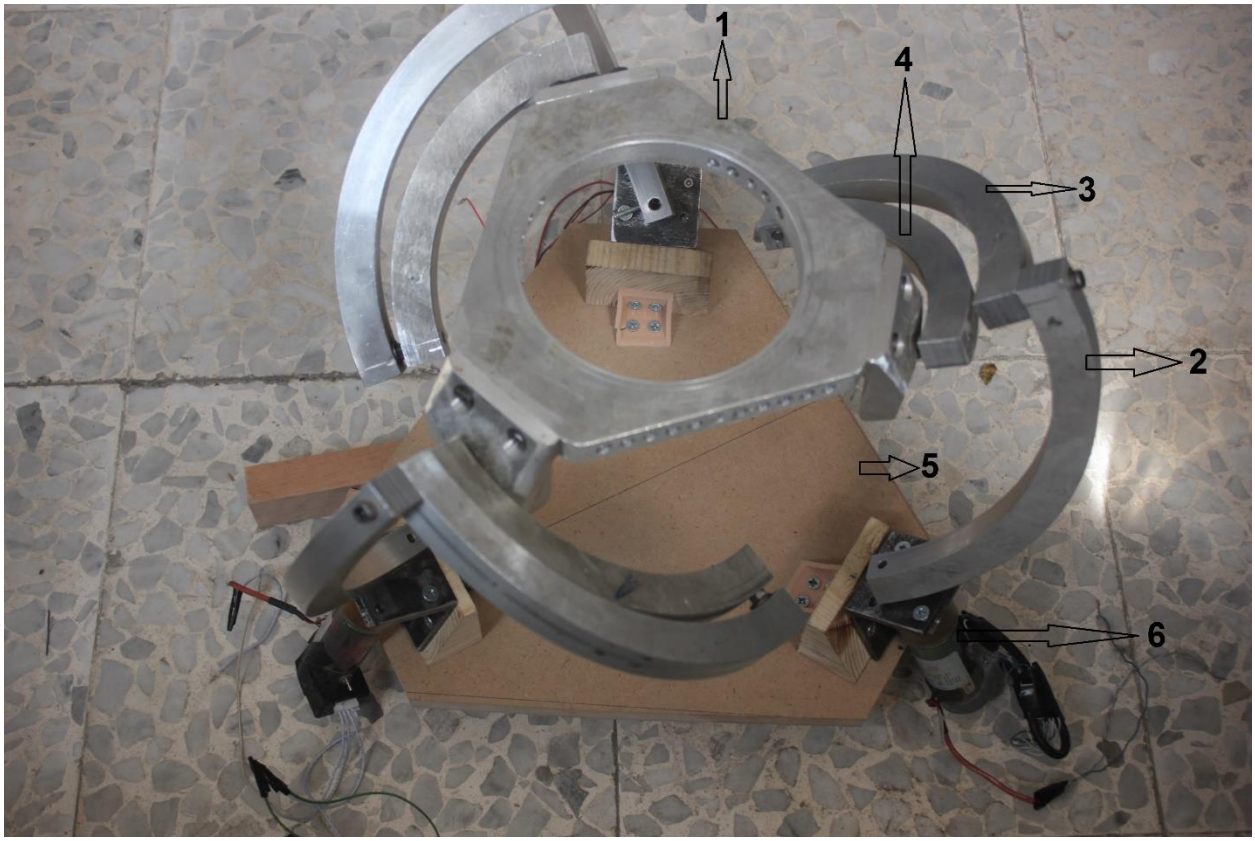


Figure 5.8: Agile Eye robot prototype.

Where

1: End-effector, Figure 5.9

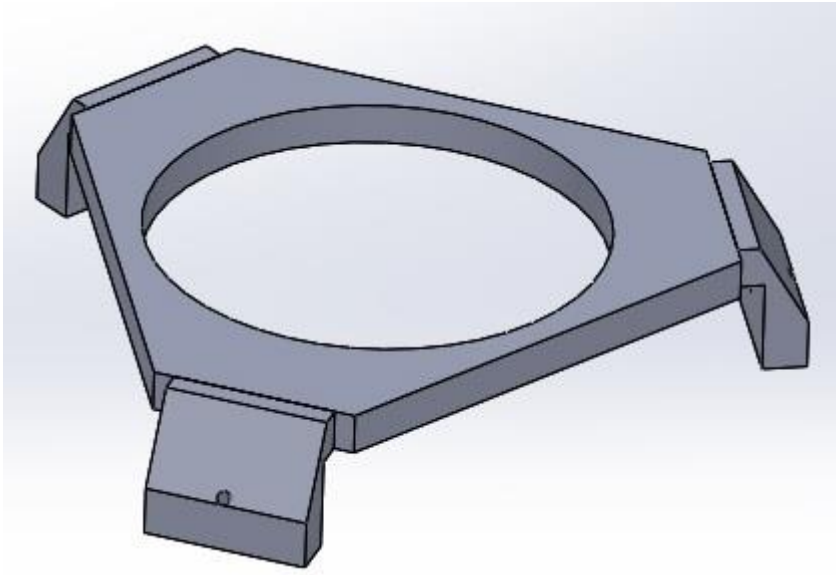


Figure 5.9: End effector

2: Active link, see Figure 5.10

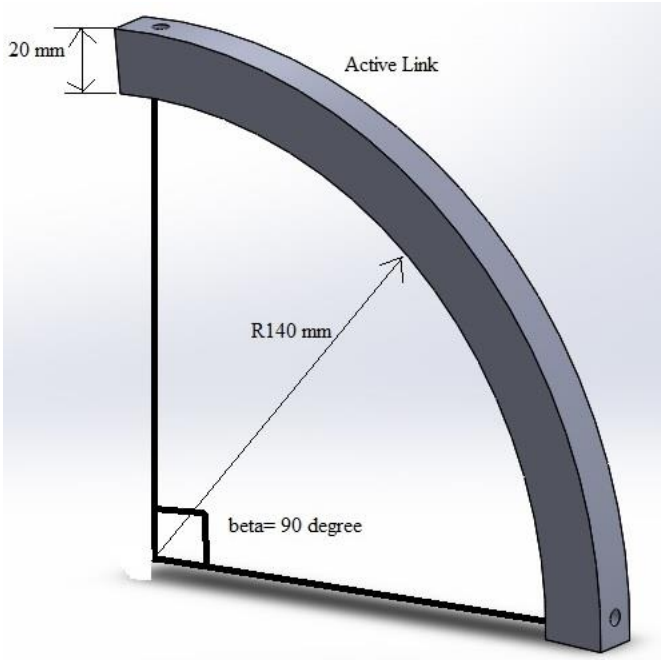


Figure 5.10: Active link.

3: Forearm link as shown in Figure 5.11

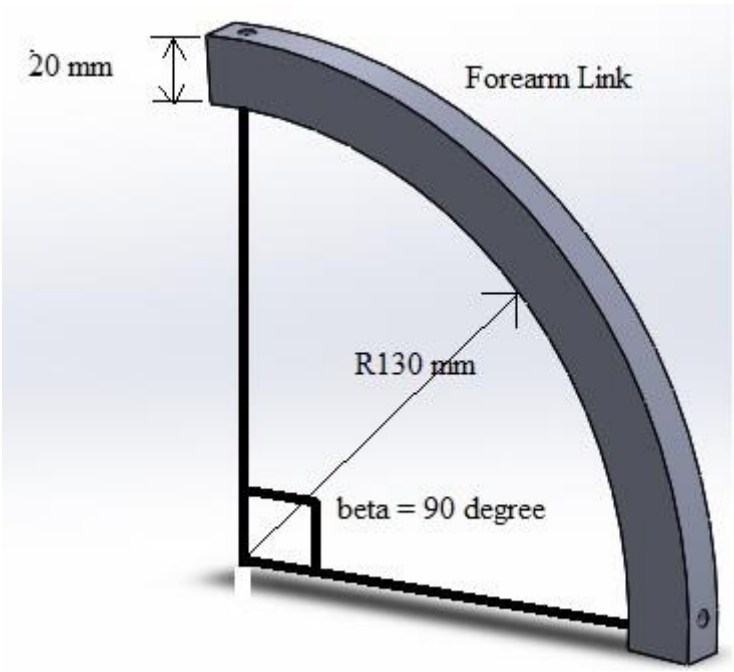


Figure 5.11: Forearm link.

4: Passive link, see Figure 5.12

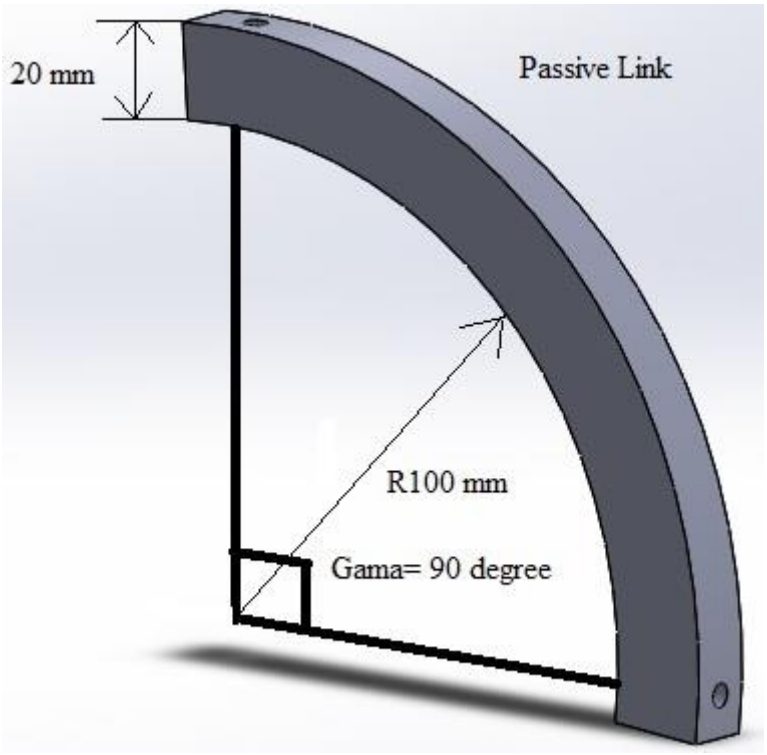


Figure 5.12: Passive link.

5: Base, as shown in Figure 5.13

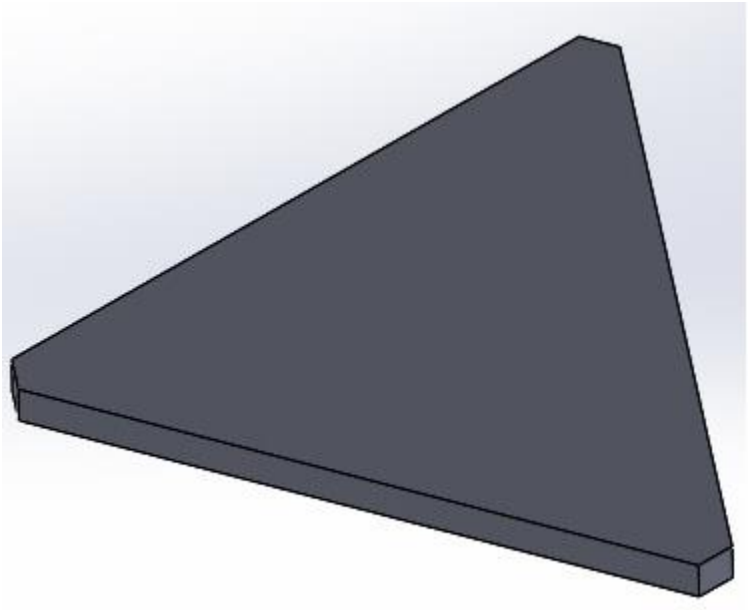


Figure 5.13: Base.

6: Motors

Chapter6

Project Assembly

This chapter view the steps of mechanical build and electrical connection for the robot and show the wiring diagram.

6.1: Mechanical Assembly

As shown in Figure 6.1, the motors are fixed on the base.

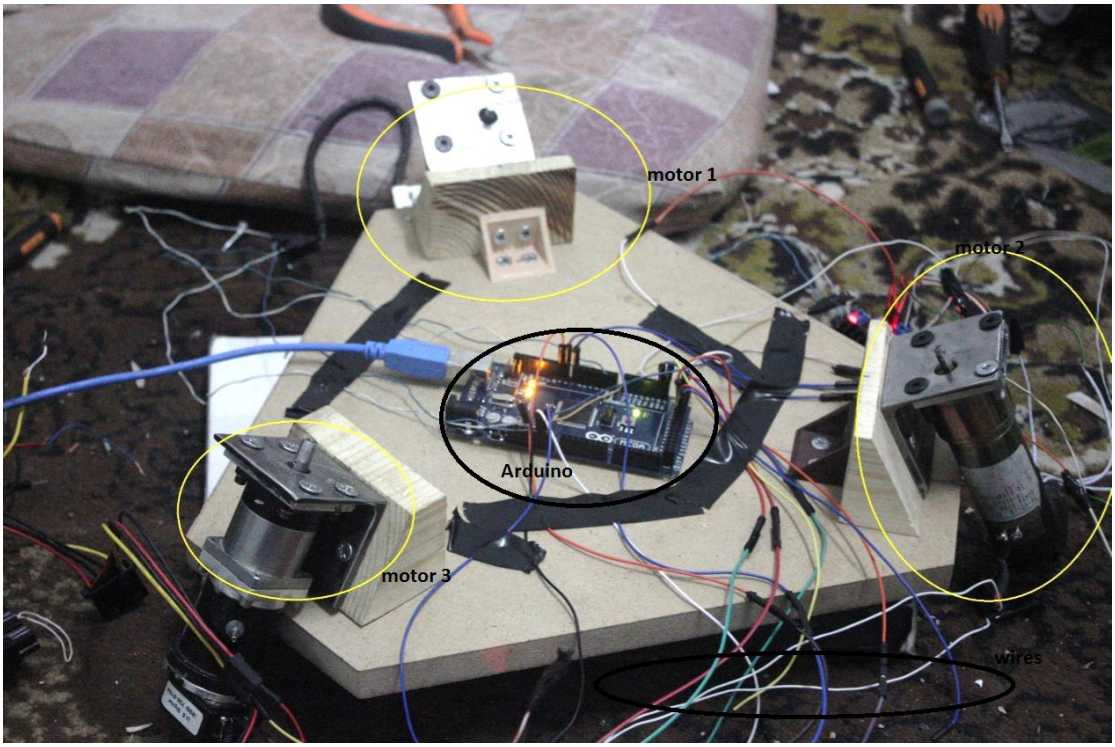


Figure 6.1 how to fix the motors on the base

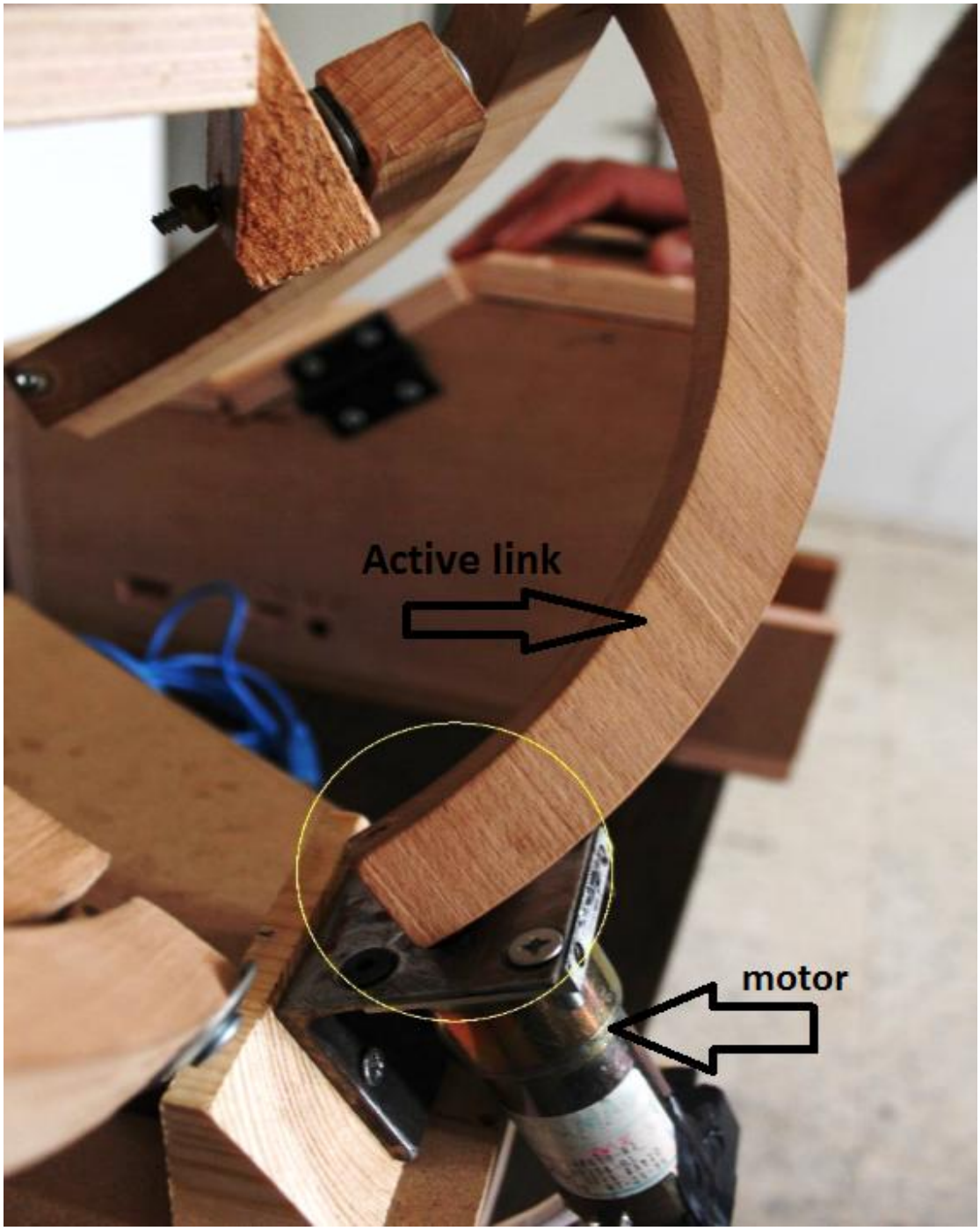


Figure 6.2 Active link connected with motor shaft

In this figure the active link is connected to motor shaft, in this location the revolute joint isn't needed.

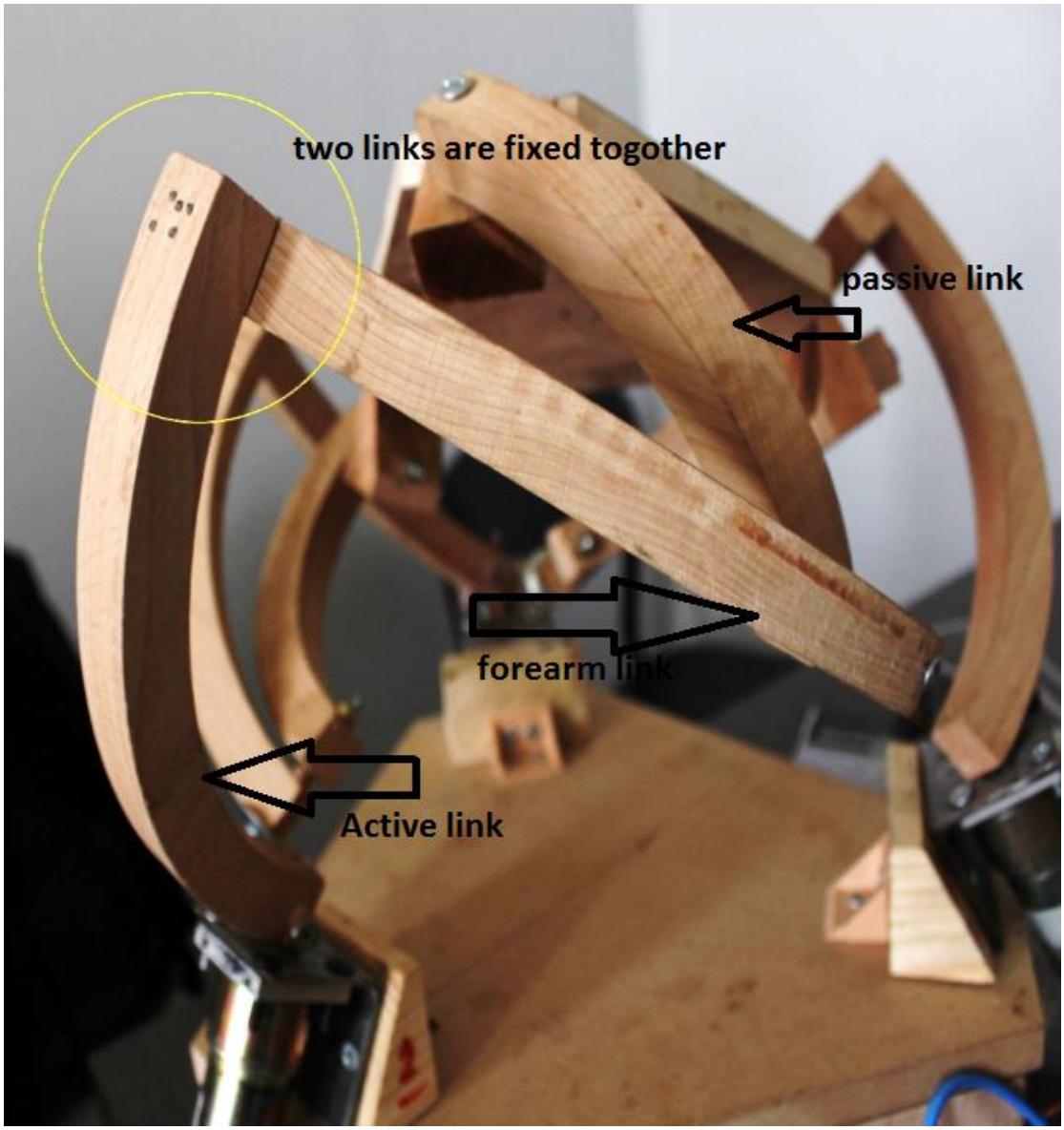


Figure 6.3 How Active link fixed to forearm link

Figure 6.3 show the links of the robot, and how active link fixed to forearm link.

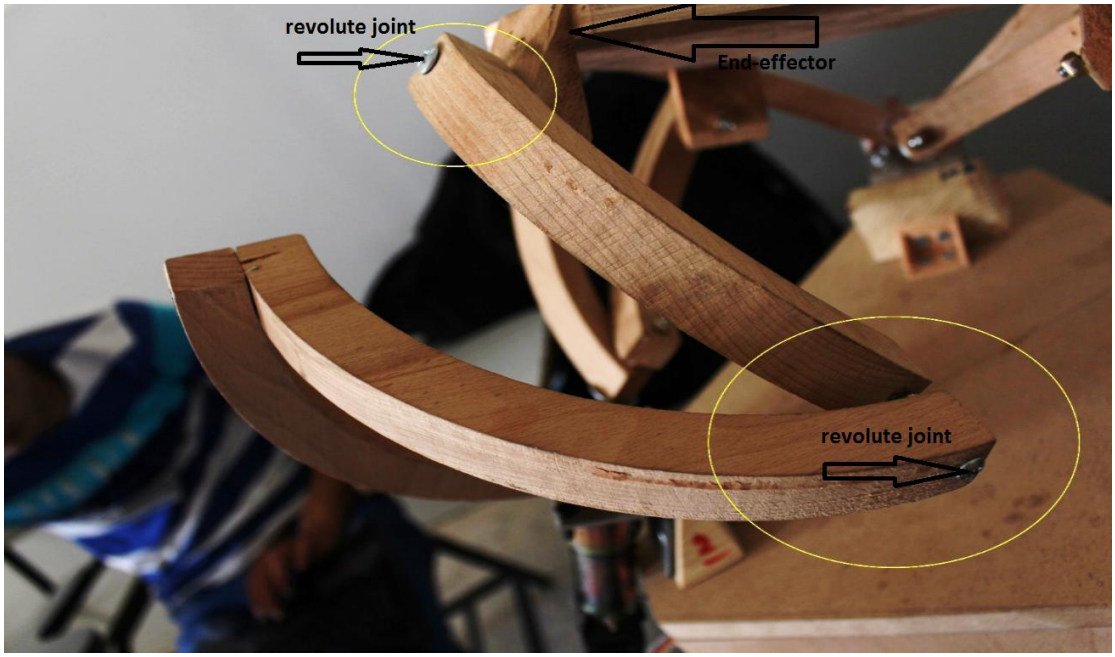


Figure 6.4 passive link connected to end effector and in the other side connected with forearm link

Revolute joint the used to connect forearm link with passive link, another revolute joint used to connect end-effector with passive link.

Figure 6.5 Aluminum prototype of the project

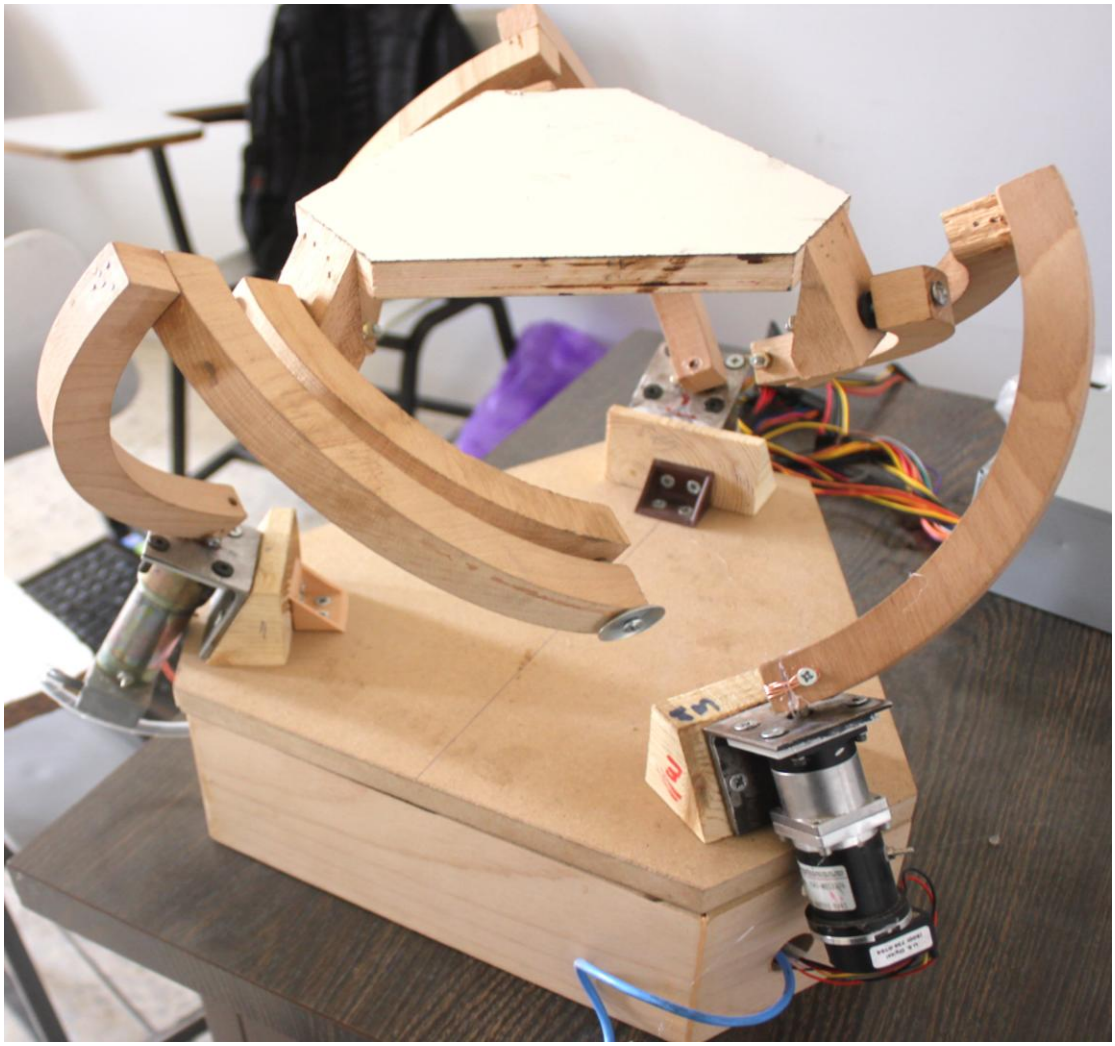


Figure 6.6 wood prototype

As shown in Figure 6.6, all of the wires, Arduino and H-bridges are hide in the case under the base of the project.

6.2 Assembly of electrical parts

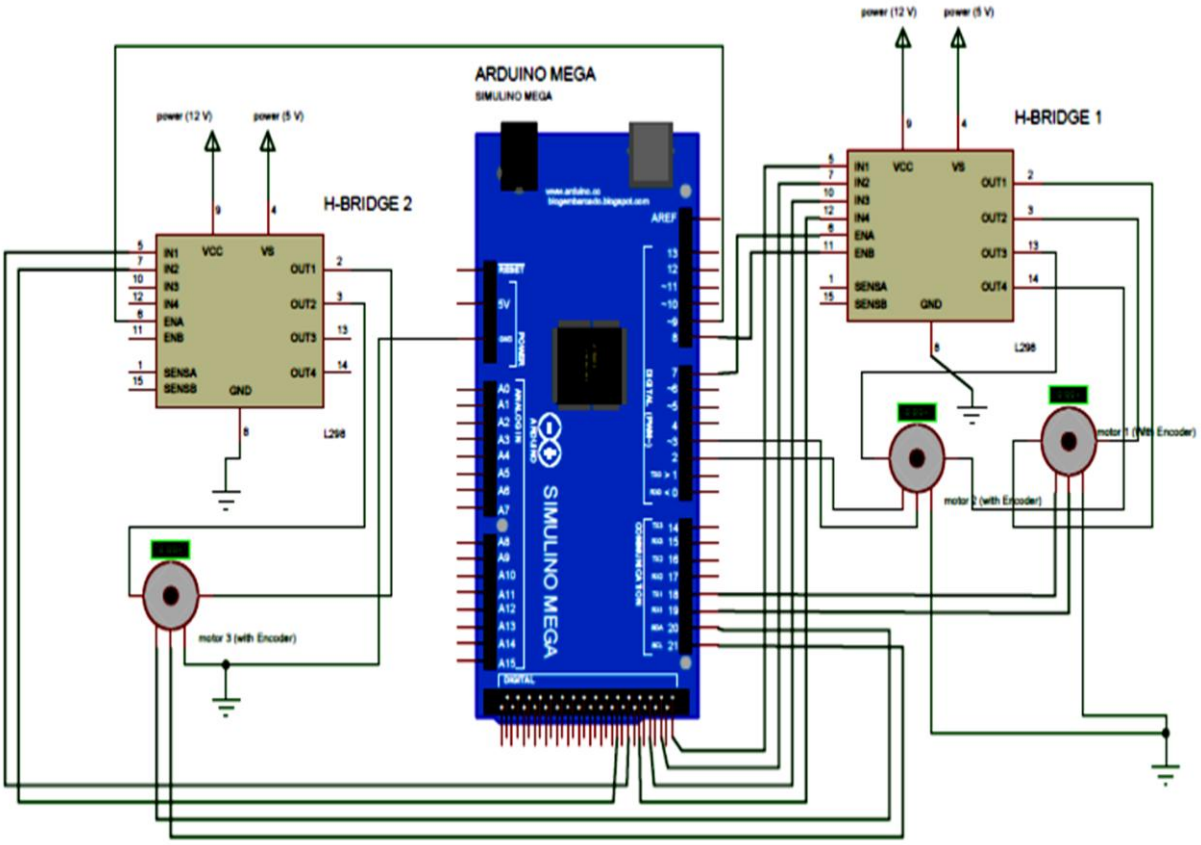


Figure 6.7 wiring diagram of the electrical circuit

Figure 6.7 are the wiring diagram of the project, it shows how to connect the motors with h-bridge and Arduino, and from this figure any one can connect the electrical circuit of the project.

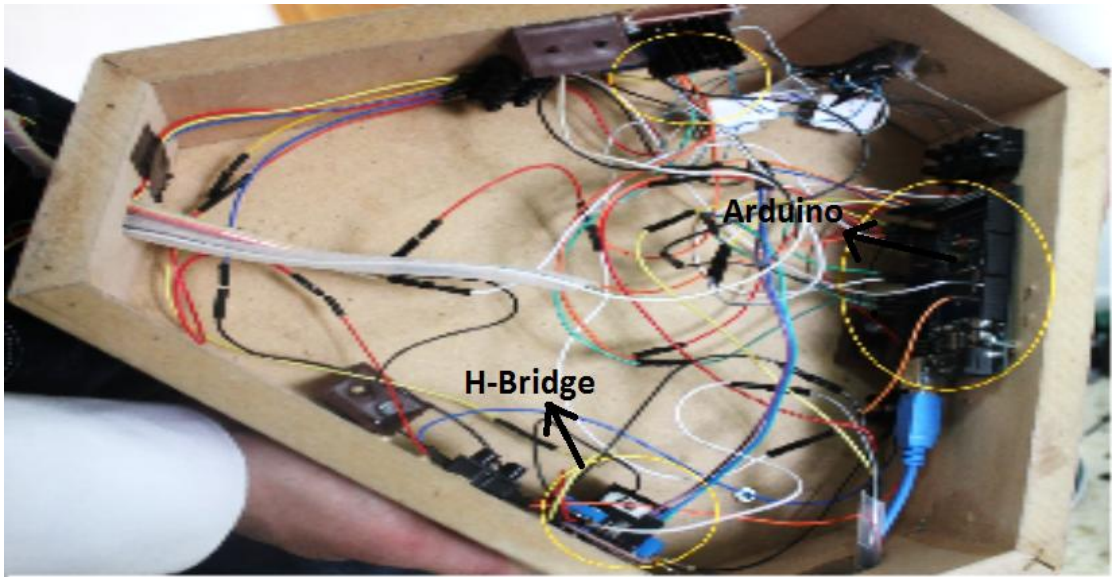


Figure 6.8 electrical connection inside the box of the robot

Figure 6.8 show the wires, h-bridge and Arduino, all these connection are in the case and not visible.

Chapter7

Controller Design and Response

7.1 Introduction

There are many controllers that are good to make the control of the project, like PID, PID with tracker controller and other controllers.

In this project we use tracker controller to control the project because this controller can deal with more than 2-dof. See Figure 7.1

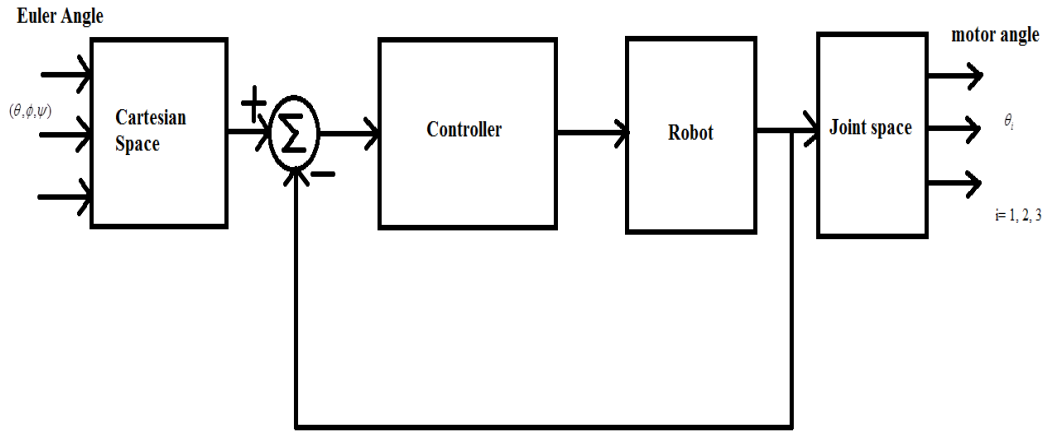


Figure 7.1 General block diagram for the control

7.2 Controller Design

Before taking any reading from the encoder, the Kpm must found for each motor.

Where Kpm is the number of pulses per revolution

The first step in this work is how to find the parameter of the controller of each motor, using MATLAB software.

The steps for finding the parameters:

1) Finding the transfer function of each motor

To find the transfer function for the motor; an encoder or a sensor is needed to give the position of the motor, then the arduino microcontroller is used to obtain the result.

In this stage we connect the motors and encoders with Arduino microcontroller, then the code is installed to microcontroller, we give the motors a simple rotation. The Arduino is give a reading of pulses (number of pulses) .We take these number to matlab and plot with time .The graph is first order , relation of voltage with speed .We take the integral of the relation that .we have to get another relation of voltage with position

$$G = \frac{14.95}{s(s+4.27)} = \frac{\theta}{v}$$
$$(s^2 + 4.27s)\theta = 14.95v \qquad (7.1)$$

2) Find the state space model of each motor

Take the Laplace inverse of equation (1.1) that give:

$$\ddot{x} + 4.27\dot{x} = 14.95v$$
$$x_1 = x$$
$$x_2 = \dot{x} = \dot{x}_1 \qquad (7.2)$$
Then
$$\dot{x}_2 = \ddot{x} = 14.95v - 4.27x_2$$

From equation (1.2) the state space representation can be calculate:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$
$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -4.27 \end{bmatrix} x + \begin{bmatrix} 0 \\ 14.95 \end{bmatrix} v$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

The input u is the voltage (v), and the feed forward matrix D is equal to zero.

Now convert the matrix above to discrete matrix, A to Ad, B to Bd and C to Cd, Ts= 0.006 sec, then by using this relation to convert the matrix the result is:

$$A_d = I + T_s * \mathrm{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + 0.006 \begin{bmatrix} 0 & 1 \\ 0 & -4.27 \end{bmatrix} = \begin{bmatrix} 1 & 0.006 \\ 0 & 0.02562 \end{bmatrix} \qquad (7.3)$$

$$B_d = T_s * B = 0.006 \begin{bmatrix} 0 \\ 14.95 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.0897 \end{bmatrix} \qquad (7.4)$$

.(Then the extended system matrix was calculated as in equation (7.5
The matlab file is in appendix A.

$$A_e = [A_d \quad \text{zeros}(2,1); -C \quad 0] = \begin{bmatrix} 1 & 0.006 & 0 \\ 0 & 0.0256 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$
$$B_e = [B_d; 0] = \begin{bmatrix} 0 \\ 0.0897 \\ 0 \end{bmatrix} \qquad (7.5)$$

3) Now the optimal method (Quadratic optimal regulator) used to find the values of the gain.

After applying these steps in the matlab, the controller gain of each motor is:

For first motor: Kp= 4.2071 , Kd = 0.024254 , KI= 10

For second motor: Kp= 4.2071 , Kd= 0.024254 , KI= 10

For third motor: Kp= 5.5 , Kd=0.06, KI= 7

Kp= proportional gain, Kd= derivative gain

Then this value of gain is used in the code of microcontroller.

4) Experimental result

The experiment starts by giving a specific angle for each motor then the angle was checked

To check this angle the code is install on the arduino and the result angle will appear on the screen.

7.3 controller used

The digital tracker controller is used to control the robot. Figure 7.2 show .the simulation block of the tracker

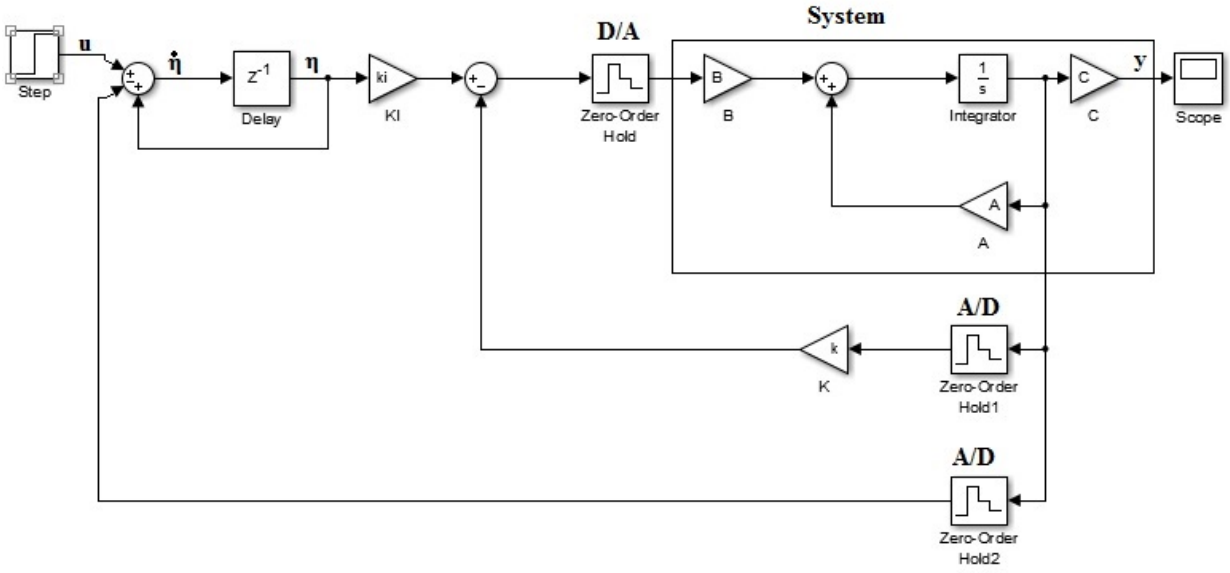


Figure 7.2: Tracker Simulink model

$$\dot{\eta} = (\mathrm{r} - \mathrm{y})\,\mathrm{T} + \eta \qquad (0.6)$$

$$\eta_{(\mathrm{k}+1)} = (\mathrm{r} - \mathrm{y})\,\mathrm{T}_{ss} + \eta \qquad (0.7)$$

Where T_{ss} settling time

7.4Expermental Result

This chapter discuss the response of the motor angles in terms of peak time, overshoot and steady state error .The plotting is exucuted using MATLAB.

The control of the robot is in the joint space and kinemtaic level. The forces and the dynamics of the robot were not insidered in this work. The motion in the joint space is generated based on the otienation of the eneffector through the inverse kinematics model.

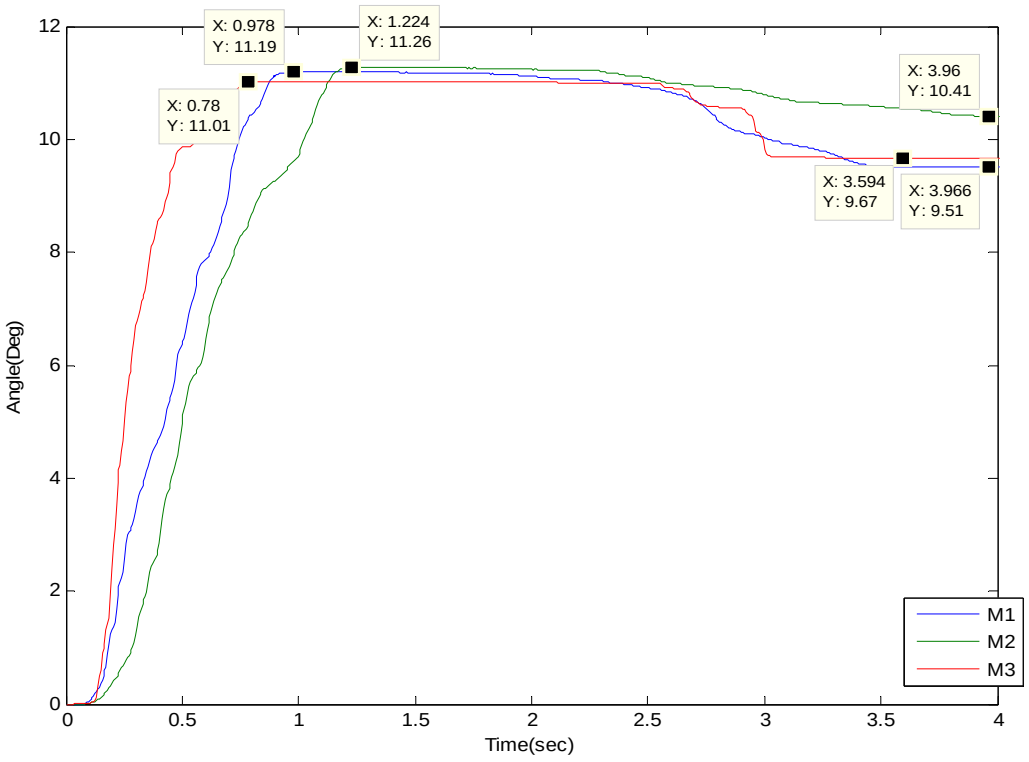


Figure 7.3 Response of 3 motors @ angle 10^0

For motor 1:

$$\%OS = 15.01\%$$

$$Tp = 0.978s$$

For motor 2:

$$\%OS = 10.41\%$$
$$Tp = 1.224s$$

For motor 3:

$$\%OS = 12.17\%$$
$$Tp = 0.78s$$

.

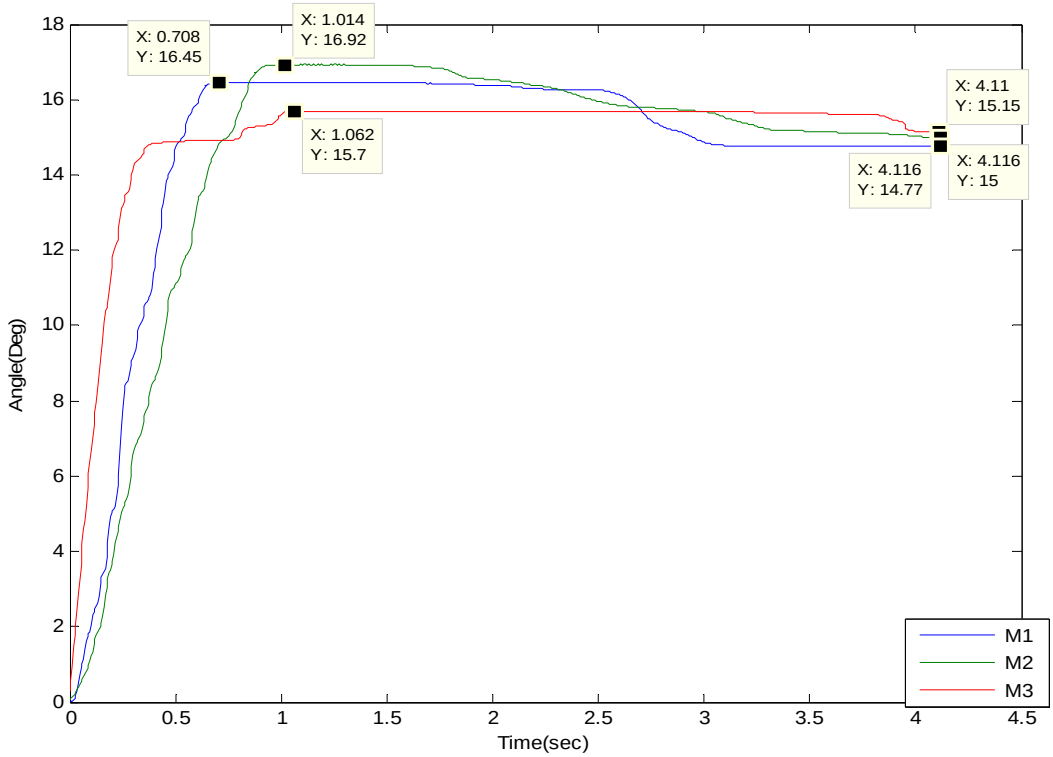


Figure7.4 Response of 3 motors @ angle 15^0

For motor 1:

$$\%OS = 11.62\%$$
$$Tp = 0.708s$$

For motor 2:

$$\%OS = 11.34\%$$
$$Tp = 1.014s$$

For motor 3:

$$\%OS = 3.5\%$$
$$Tp = 1.062s$$

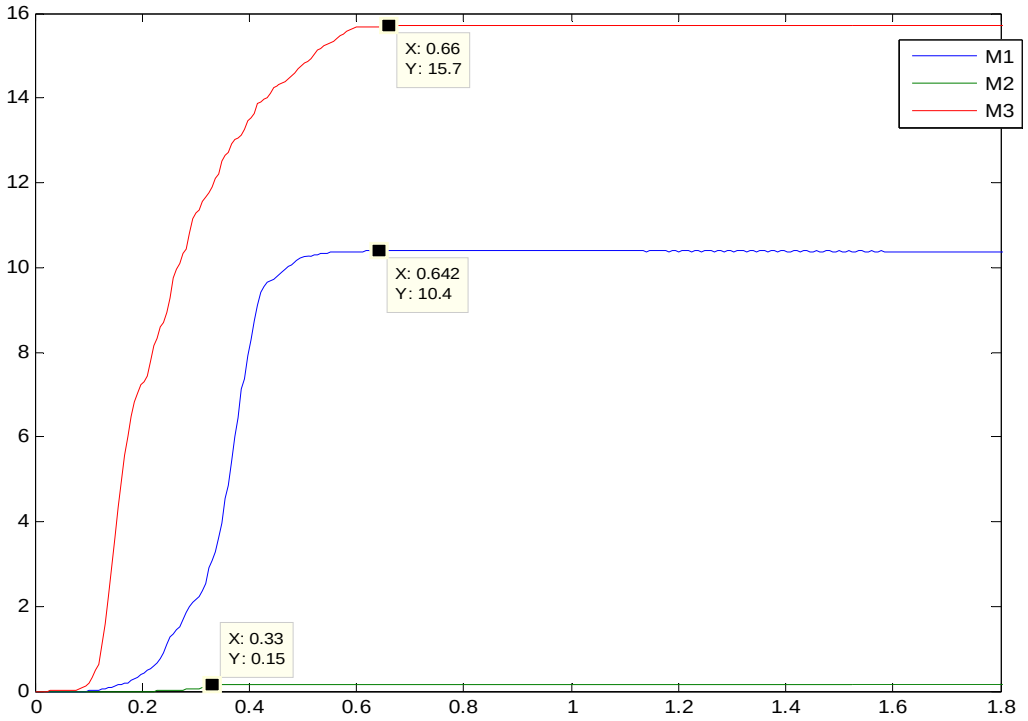


Figure 7.5 Response of the 3 motors @ different angles

For motor 1:

$$\%error = 3.86\%$$
$$Tp = 0.662s$$

For motor 2:

$$\%error = 33.33\%$$
$$Tp = 0.33s$$

For motor 3:

$$\%error = 4.45\%$$
$$Tp = 0.642s$$

From figures above its notice that the inverse kinematics was used to decide the angle that we must entered to the robot. For example if we need end effecter to reach an angles: α, β, γ

By invers kinematics calculations the input angles: $\theta_1, \theta_2, \theta_3$

Since the same controller was used in different motors we have two different responses, however to solve this problem the parameters must be tuned for the two motors to get the same responses. Backlash between gears and deflection on motor shaft affect to response on motor 3.

On the other hand the overall real response represent a high overshoot compared to the simulated one because of sampling time was too high.

Chapter 8

Conclusion & Suggestion

8.1 Conclusion

The model of the robot was built and the controller was designed .Different methods Can be used to control the motor and different embedded systems can used.

8.2 Suggestion for the future work

Although the robot has been successfully build from Aluminum and wood, material can be replaced with other material like echelon.
The matlab simulation should be used to simulate the controller of the robot, and other controller should be used to control the motors.
On the other hand Arduino microcontroller can be replaced by other embedded controller that is faster to give best control.

Appendix

A.1 Controller Code (m-file)

```
%% ======================================================%%
%%                  Agile Eye Robot                     %%
%%===========    Tracker Controller    =================%%

Ts=0.006;           % sampling Time
A=[0 1;0 -4.27]     % system matrix of motor 1
B=[0 14.95]';       % Input matrix
Ad=[1 0.006;0 0.02562]; % Discrete system matrix
Bd=[0 0.0897]'; % Discrete input matrix
C=[1 0]; % output matrix
Ae=[Ad zeros(2,1);-C 0]; % Extended system matrix
Be=[Bd;0]; % Extended input matrix
q=eye(3); % state weight matrix
r=1.5; % input weight matrix
q(3,3)=150
ke=lqr(Ae,Be,q,r); % Evaluate the gain by using lqr command, where Ke
is the extended gain
k1=ke(1)
k2=ke(2)
ki=-ke(3)
k=ke(1:2)
```

A.2 Microcontroller C Code

```
#define MAX_BUF        (64)  // What is the longest message Arduino can store?
char buffer[MAX_BUF];  // where we store the message until we get a ';'
int count;  // how much is in the buffer
#include <Encoder.h>
#include <TimerOne.h>
Encoder x_dir(2,3);
Encoder y_dir(18,19);
Encoder z_dir(20,21);
const int mcw1=22;
const int mccw1=24;
const int pwm1=7;
const int mcw2=26;
const int mccw2=28;
const int pwm2=8;
const int mcw3=32;
const int mccw3=30;
const int pwm3=9;

float x1=0.0;
float x1_dot=0.0;
float y1=0.0;
float y1_dot=0.0;
float z1=0.0;
float z1_dot=0.0;

float lastTime=0.0;
float sx=0.0;//////////
float sy=0.0;////////////
float sz=0.0;////////////
```

```
float outx=0.0,etax=0.0,eta_nx=0.0;
float outy=0.0,etay=0.0,eta_ny=0.0;
float outz=0.0,etaz=0.0,eta_nz=0.0;

float x1_p=0.0,y1_p=0.0;
float z1_p=0.0;

float forcex=0.0,voltagex=0.0,Kfvx=1;
float forcey=0.0,voltagey=0.0,Kfvy=1;
float forcez=0.0,voltagez=0.0,Kfvz=1;

double K1x=4.3;
double K2x=0.048;
double Kix=8.89;

double K1y=4.3;
double K2y=0.048;
double Kiy=4.95;

double K1z=5;
double K2z=0.07;
double Kiz=6.782;

double X,Y,Z;
double x_in=0.0,y_in=0.0,z_in=0.0;
double Kpm_x=360.0/32000.0;/////////////
double Kpm_y=360.0/62000.0;/////////////////
double Kpm_z=360.0/62500.0;/////////////////
float T=0.03;
```

```
void setup()
{
  pinMode(mcw1,OUTPUT);
 pinMode(mccw1,OUTPUT);
  pinMode(pwm1,OUTPUT);
   pinMode(mcw2,OUTPUT);
 pinMode(mccw2,OUTPUT);
  pinMode(pwm2,OUTPUT);
   pinMode(mcw3,OUTPUT);
 pinMode(mccw3,OUTPUT);
  pinMode(pwm3,OUTPUT);
    Timer1.initialize(30000);
  Timer1.attachInterrupt( timerIsr );
    Serial.begin(9600);
  Serial.println("Basic Encoder Test:");
}

void loop()
{

  // listen for serial commands
  while(Serial.available() > 0)
  {  // if something is available
    char c=Serial.read();  // get it
    if(count<MAX_BUF) buffer[count++]=c;  // store it
    if(buffer[count-1]==';') ;  // entire message received
  }//while

    if(count>0 && buffer[count-1]==';')
  {// we got a message and it ends with a semicolon
```

```
    buffer[count]=0;  // end the buffer so string functions work right
    Serial.print(F("\r\n"));  // echo a return character for humans
    processCommand();  // do something with the command
    ready();
  }//if

 if(outx >=0.0){
 digitalWrite(mcw1,1);
digitalWrite(mccw1,0);
if(outx<5){outx=0.0;}
 analogWrite(pwm1,outx);
  }
  if(outx < 0.0) {
digitalWrite(mcw1,0);
digitalWrite(mccw1,1);
if(-5<outx){outx=0.0;}
 analogWrite(pwm1,0-outx);
  }
  ////////////////////////////
  ////////////////////////////

   if(outy>=0.0){
 digitalWrite(mcw2,1);
digitalWrite(mccw2,0);
if(outz<5){outz=0.0;}
 analogWrite(pwm2,outy);
  }
  if(outy < 0.0) {
 digitalWrite(mcw2,0);
digitalWrite(mccw2,1);
```

```
if(-5<outy){outy=0.0;}
 analogWrite(pwm2,0-outy);
  }
  //////////////////////////
  //////////////////////
    if(outz>=0.0){
 digitalWrite(mcw3,1);
digitalWrite(mccw3,0);
if(outz<5){outz=0.0;}
 analogWrite(pwm3,outz);
  }
  if(outz < 0.0) {
 digitalWrite(mcw3,0);
digitalWrite(mccw3,1);
if(-5<outz){outz=0.0;}
 analogWrite(pwm3,0-outz);
  }
}
  ////////////////////////////////
  ////////////////////
   void timerIsr()
   {
  unsigned long now;
    now = millis();
    double dT,errorx=0.0,dErrx=0.0;
    dT =(float)(now - lastTime); //Our Sampling time
    x1=((float)(x_dir.read()))*Kpm_x;
    y1=((float)(y_dir.read()))*Kpm_y;
    z1=((float)(z_dir.read()))*Kpm_z;
  Serial.print(x1);
```

```
Serial.print("    ");
Serial.print(y1);
Serial.print("    ");
Serial.println(z1);
  X=x1;
  Y=y1;
  Z=z1;

  x1_dot=(float)((x1-x1_p)/(T));
  y1_dot=(float)((y1-y1_p)/(T));
  z1_dot=(float)((z1-z1_p)/(T));

  eta_nx=(sx-X)*T+etax;
  eta_ny=(sy-Y)*T+etay;
  eta_nz=(sz-Z)*T+etaz;

  forcex=Kix*etax-K1x*x1-K2x*x1_dot;
  voltagex=forcex*(Kfvx);
  outx=voltagex;//*255.0/5.0;
  if(outx>250){outx = 250;}
  else if(outx<-250){outx = -250;}
  x1_p=x1;
  etax=eta_nx;

forcey=Kiy*etay-K1y*y1-K2y*y1_dot;
  voltagey=forcey*(Kfvy);
  outy=voltagey;//*255.0/5.0;

if(outy>250){outy= 250;}
else if(outy<-250){outy = -250;}
```

```
    y1_p=y1;
   etay=eta_ny;

     forcez=Kiz*etaz-K1z*z1-K2z*z1_dot;
   voltagez=forcez*(Kfvz);
   outz=voltagez;//*255.0/5.0;

 if(outz>250){outz= 250;}
 else if(outz<-250){outz = -250;}
    z1_p=z1;
   etaz=eta_nz;
 }
 void processCommand(){
 sx = parsenumber('a',-1);
 sy = parsenumber('b',-1);
 sz = parsenumber('c',-1);
 Serial.println("Ready .............");
 Serial.print ("Theta_1  =   ");
 Serial.println(sx);
 Serial.print ("Theta_2  =   ");
 Serial.println(sy);
 Serial.print ("Theta_3  =   ");
 Serial.println(sz);
 Serial.println(">>");
 Serial.println("******************");
}//processCommand();

float parsenumber(char code,float val) {
 char *ptr=buffer;
 while(ptr && *ptr && ptr<buffer+count) {
```

```
    if(*ptr==code) {
      return atof(ptr+1);
    }//if
    ptr=strchr(ptr,' ')+1;
  }//while
  return val;
}//parsenumber
void ready() {
  count=0;  // clear input buffer
  Serial.print(F(">"));  // signal ready to receive input
}//ready();
```