Final Year Project Report

Autonomous Robotic Arm Based on Computer Vision

Noor Hassan

Ali Khan

Abdul Rasheed

A report submitted in part fulfilment of the degree of

Bachelor of Engineering (Electrical Engineering)

Supervisor: Dr Muhammad Asim

Department of Electrical Engineering

Sukkur IBA University, Sukkur

September 7, 2019

CERTIFICATE

This Thesis is written by **Noor Hassan, Ali khan and Abdul Rasheed** under the direction of their supervisor **Dr. Muhammad Asim** and approved by all the members of thesis committee, has been presented to and accepted by the Head of Department of Electrical Engineering, in partial fulfilment of the requirements of the degree of **Bachelors of Electrical Engineering.**

_____ _____

Project Supervisor HOD, Electrical Engineering

_____ _____

Internal Examiner External Examiner

_____

Vice Chancellor, Sukkur IBA University

DEDICATION

To our Parents and Teachers for their support, encouragement and endless love who taught us to think clearly & motivated us to try our hardest in everything we do, without them we could not have reached our goals and to all those who believe in the power of learning

ABSTRACT

A robotic arm is one type of mechanical arm which is similar to a human arm and can do most of the works that a human arm can do. It can perform all typical kinematic operations as a human arm do. A robotic arm is programmable and can do works through instructions. In today's world robotic arm are used in industries, rescue purpose and various research works. The presence of hazards to human health in chemical process plant and nuclear waste stores leads to the use of robots and more specifically unmanned spaces. Rapid and accurate performance of robotic arm movement and positioning, coupled with a reliable manipulator gripping mechanism for variable orientation and range of deformable or geometric and colour products, will lead to smart and intelligent operation of high precision equipment. Considering the facts we thought of designing and making some improvements in robotic arm.

Our robotic arm is an articulated type robotic arm and controlled by raspberry pi which can move 300 degrees and that can identify different objects located in its workspace according to their colour and places the objects in a certain location with help of video camera as a visual feedback. The arm has 4 Degrees of Freedom (DOF) with Dynamixels motors (smart servos) at each joints. It is able to move to a specific point of its workspace through x, y and z axis values automatically if the points are given. For implementing algorithm we are using U2D2 (USB to Dynamixel) controller it receives commands through Python (a programming software) and sends it to these Dynamixel motors.

This project introduces intelligence to industries to reduce human errors and increase the quality and mass production of industries. It can operate in under hazardous circumstances (high pressure, high temperature and locations that can be unstable for human operation), so it reduces that risk of human injuries. It can be updated and controlled easily to achieve a large number of applications.

TABLE OF CONTENTS

LIST OF TABLES/FIGURES

ABBREVATIONS

Abbreviation	Full Form
(DoF)	Degree of Freedom
(DH)	Denavit-Hartenburg
(U2D)	Universal to Dynamixel Converter
(CD)	Compact Disc
(DVD)	Digital Versatile Disc
(HDMI)	High Definition Multi-media Interface
(SMSC)	Standard Micro Systems
(PCB)	Printed Circuit Board
(Mb/s)	Megabits per second
(SD)	Secure Digital
(DSI)	Display Serial Interface
(IC)	Integrated Circuit
(GB)	Giga Bytes
(GPIO)	General Purpose Input/output
(EEPROM)	Electronically Erasable Programmable Read Only Memory
(OpenGL)	Open Graphics Library
(MHz)	Mega Hertz
(EAGLE)	Easily Applicable Graphical Layout Editor
(dpi)	Dot per inch
(2D)	Two Dimension
(3D)	Three dimensions
(png)	Portable Network Graphics
(mm/s)	Mili meter per second
(rml)	Redline Markup Language
(CAD)	Computer Aided Design
(I)	Identity Matrix
(Tp)	Position Matrix
(Tff)	Forward Transformation Matrix
(OCR)	Optical Character Recognition
(ANPR)	Automatic Number Plate Recognition
(CGI)	Computer Generated Imagery
(mocap)	Motion Capture
(XML)	Extensible Markup Language
(HTML)	Hypertext Markup Language
(WAV)	Wave
(PIL)	Python Imaging Library
(CV)	Computer Vision
(BCM)	Back Channel Message

ACKNOWLEDGEMENT

All praise be to ALLAH, the Almighty, the most Merciful and Compassionate, who granted us perseverance to undertake and accomplish this research work.

The work on this project has been an inspiring, often exciting, sometimes challenging, but always interesting experience. It has been made possible by many other people, who have supported us. So we would like to take this opportunity to express our sincere gratitude to all those who have contributed in completing this project.

Firstly, we would like to thank our thesis adviser **Dr. Muhammad Asim**, Associate Professor, Dept. of Electrical Engineering (EE), **Sukkur IBA University**; for his supportive guidance and feedbacks towards our project for its completion. Secondly we would like to thank **Engr. Syed Farhan Ali Shah** for his support in making and joining various parts in our project and others without whom our project would not be successful one.

Finally, We wish to thank our parents for their undivided support and interest, who has inspired us and encouraged us to go our own way, without whom we would unable to complete this project.

CHAPTER 1

INTRODUCTION

This chapter sheds light on evolution of industrial robotics. The chapter also provides comparison of basic manipulators used in industries.

1.1 Background

Robotics is emerging field in many areas. This field is present everywhere in newspapers and journals almost every day; about drone strikes in war, about robots in nursing homes keeping the elderly company or about cars with the ability to park or even drive without human supervision.

Robots are evolving; research is being made on producing more sociable robots, which are able to read facial expressions and hold conversations. A robot is “an automated machine or vehicle, capable of independent perception, reasoning, and action”. An autonomous robot is “a robot that does not require direct human involvement. In our modern society, the development of technological solutions has been in great focus. Robotic arms has revolutionized the industry by making it more cost efficient, improving build quality, decreasing production time. In the future, robotic arms will be even more important in manufacturing.
In automation Industrial robots are programmed for a single task using the sensory information. A vision system is considered to be the most sophisticated sensor in the present automation industry. The most common technology used at present in the industry is image processing. Due to the advent of powerful cameras, computers, controllers for controlling the machines and sophisticated tools image processing has become the most powerful emerging technology. Image processing is basically improving and enhancing the images taken in daily life using cameras considered as vision sensors for various applications. In the last few decades different techniques have been developed for detecting the objects using vision systems.

The evolution of the technical needs of society and the technological advances achieved because that robots are widely used in many areas such as agriculture, surgery assistance, and automotive assembly line. A robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks. Robotic Arm is one of the key parts of robot and also one of the most important industrial robots, but they are typically designed to repeat tasks. Integration of the robot with the information coming from the camera can extend the functionality of their in automatic industrial process.

1.2 Aims and objectives

This project meets the objective by developing an eye-bot, a typical model used to pick and place the desired colour objects from one location to another. The benefits are that it will increase the speed and accuracy of the colour sorting process. It will cut down the cost of colour sorting process. This project will also optimize the overall productivity of that particular industry.

1.3 Literature Review

Fig. 1.1 Unimate Arm.

Unimates introduced its first robotic arm in 1962 (Fig. 1.1). Industrial robots graduated from the laboratory to the factory. It is interesting that in this process the robotic arm's movements and the DoF incorporated nautical terms for robotics—pitch, yaw, and roll.

Fig. 1.2 KUKA Robotic Arm.

KUKA is German manufacturer of industrial robots for factory automation. This company was founded in 1898 (Fig.1.2). KUKA manipulator increase the production process in many industries.

1.4 Robotic Arm Comparison

The following table is a representation of the various off the shelf robot arms studied during the extensive review of literature. Many of these robot arms represent a whole series of options that all fall within the same electrical / mechanical specifications and price range. This table shows the spectrum of robot arms available, Along with how each arm meets the required constraints, and

why the decision was made to design an arm instead of purchasing one. Following table demonstrating the types of Robotic Arms and their specifications.

Table. 1.1 Types of Robotic Arms and their Specifications

Name	Company	D.O.F	Weight	Control	Feedback	Price
AL5A	Lynxmotion	4	23 oz.	Serial	None	$ 338.65
P-Series	Mobile Robots	7	50 oz.	Serial	None	$ 6000
HS-Series	Denso	4	2000 oz.	Serial	Position, Velocity	$ 3500
Pioneer Arm	Mobile Robots	5	50 oz.	Serial	None	$ 6000
Scorbot-ER9	Intelitek	5	2000 oz.	USB	Position	$ 30,000
Scorbot-ER4	Intelitek	5	400 oz.	USB	Position	$ 8000
Robotic Arm Edge	OWI	4	23 oz.	USB	None	$ 150

1.5 Purpose

A robotic arm is used to pick and place object at the specified locations in a given hemisphere 3D space. Robotic Autonomous manipulator control, can be activated by entering coordinates in the PC which are then transferred to the onboard chip computer and thereby trigger motion of the arm. A manipulator based on computer vision which takes input from the camera mounted on the top of the workspace which purpose is to sort the objects by recognizing their colors through OpenCV method accordingly. The whole process of detection is implemented in a modern programming language.

CHAPTER 2

AUTONOMOUS ROBOTIC ARM

Objective of this chapter is to introduce the mechanical components of a robotics manipulator and to lay down the framework to develop mathematical model of robotic arm.

2.1 Introduction to Robotic Arm

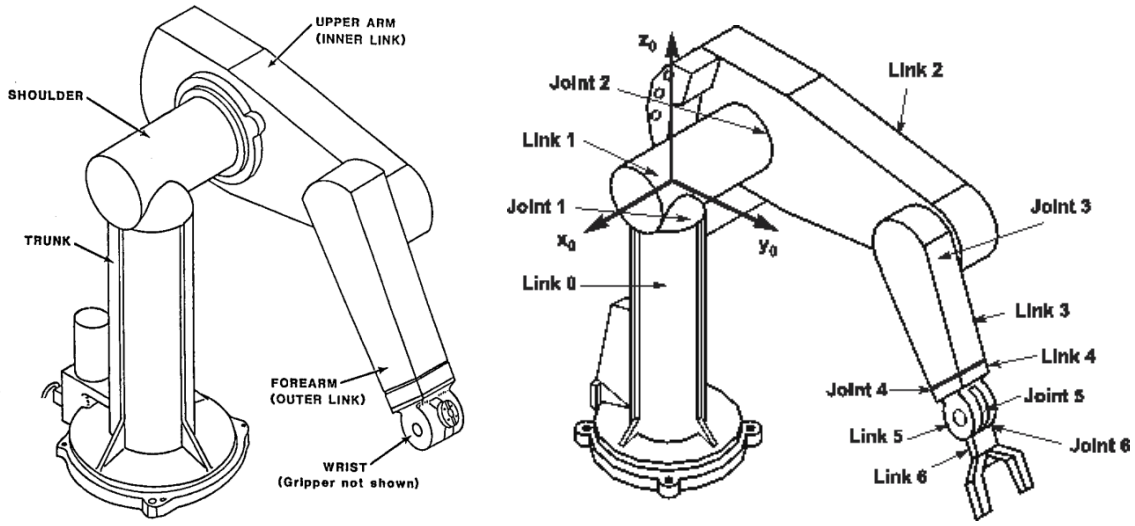

Fig. 2.1 Robotic Arm with DoF illustration

To understand automation, a general robotic arm is considered. This robotic arm have motions similar to those of a human arm. These machines comprise both a "shoulder" and "elbow" joint along with a "wrist" axis and vertical motion as illustrated in Fig. 2.1.

Robots are ideal for general-purpose applications requiring fast, repeatable, and accurate point-to-point movements such as palletizing, machine loading/unloading, and assembly. Other tasks include spray painting, welding, and sealing applications.

2.1.1 DoF of Manipulator

In mechanical engineering, aeronautical engineering and robotics, degrees of freedom (DOF) describes flexibility of motion. A mechanism or linkage that has complete freedom of motion (even if only in a limited area, or envelope) has six degrees of freedom. Three modes are translation - the ability to move in each of three dimensions. Three are rotation, or the ability to change angle around three perpendicular axes [4].

2.1.2 Shoulder joint

The shoulder joint is the highest load-bearing joint in the arm. The three degrees-of-freedom at the shoulder are pitch, yaw, and roll. The shoulder has the widest range of motion of any joint in the human body and is the foundation for most modern robotic arms. The horizontal flexion and extension (yaw) of the human shoulder is 160°. The forward flexion and hyperextension of the shoulder (pitch) is 240°. Finally, the medial and lateral rotation (roll) is 160°. In the normal human, the pitch and yaw are perpendicular to the arm, whereas the roll is in-line with the arm.

2.1.3 Elbow joint

The elbow joint provides extension, retraction, reach-around, and angular reorientation of the wrist and hand. Classically, the elbow provides 150° of pitch. Many types of mechanical elbow joint have been used in robotic arm manufacture. These include telescoping, revolute (subdivided by drive-train), intermediate, remote, and direct. Of these mechanical types, the revolute is most similar to the human arm. The telescoping was an early type of robotic arm joint, it deviates much from the human anatomic concept and applications have been limited.

2.1.4 Wrist joint

The wrist mechanisms developed for robotic arms were crucial in even the earliest prototypes. The wrist is the end effector terminus of the robotic arm and it allows the arm to be manipulated in three-dimensional space. The earliest robotic applications of wrists were in the very Wrist painting and welding robotic arms. Much more sophisticated wrists enable more dexterous teleported systems, but singularity problems are still a problem, and almost everyone who has used the da Vinci Surgical System has probably experienced gimbal locking of the wrist.

2.1.5 End Effector

Fig. 2.2 Silicon Suction Gripper.

The hand is a “differentiated” end-effector of the robotic arm that defines the purpose and the capacity of the arm. The hand is a multi-tasking tool capable of diverse functions, for example grasping, manipulating, and pushing. A robotic hand has multiple control issues, both motor and sensory perception. Many universities are currently investigating this topic, more so than in industry. In this project we have used pneumatic arm gripper, which basically pick and place objects with the help of vacuum pump.

2.2 Types of Robotic Arm

The common types of robotic arm are given as follows.

2.2.1 Cartesian

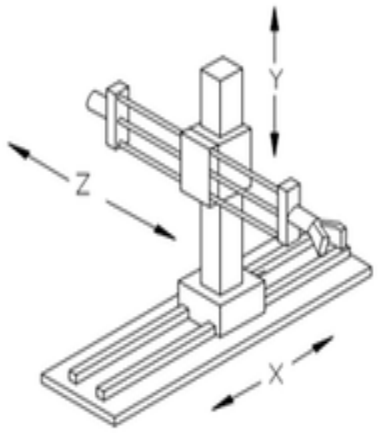

Fig. 2.3 Cartesian robotic arm

This type of robots are also called rectilinear or gantry robots. Cartesian robots have three linear joint that use the Cartesian coordinate system (x, y and z). They also may have an attached wrist to allow for rotational movement. The three prismatic joints deliver a linear motion along the axis.

2.2.2 Cylindrical

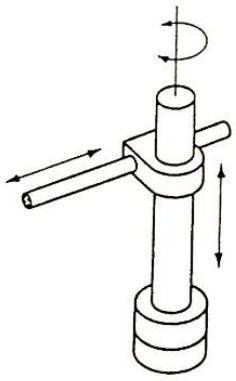

Fig. 2.4 Cylindrical robotic arm

Cylindrical robots have at least one rotatory joint and at least one prismatic joint to connect the link. The rotary joint uses a rotational motion along the joint axis, while the prismatic joint moves in a linear motion. Cylindrical robots operate within a cylindrical-shaped work space.

2.2.3 Articulated

An articulated type manipulator design features rotary joints and can range from simple two joint structures to 10 or more joints. The arm is connected to the base with a twisting joint. The links in

the arm are connected by rotary joints. Each joint is called an axis and provides an additional degree of freedom, or range of motion. Industrial robots commonly have four or six axes.

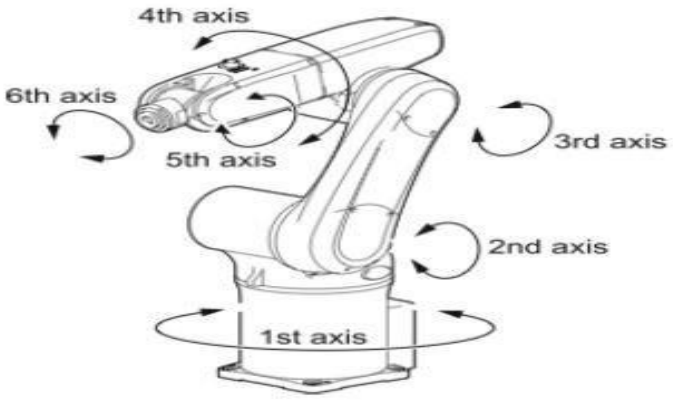

Fig. 2.5 Articulated robotic arm

2.2.4 Scara

Commonly used in assembly applications, this selectively compliant arm for robotic assembly is primarily cylindrical in design. It features two parallel joints that provide compliance in one selected plane.

2.2.5 Spherical

A spherical robot is a robot with two rotary joints and one prismatic joint; in other words, two rotary axes and one linear axis. Spherical robots have an arm which forms a spherical coordinate system.

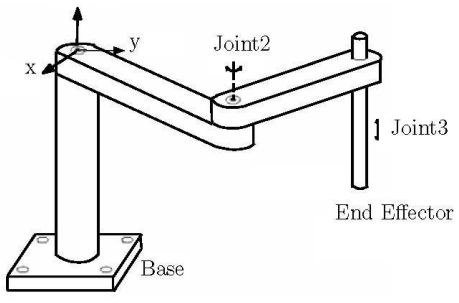

Fig. 2.6 Scara robotic arm

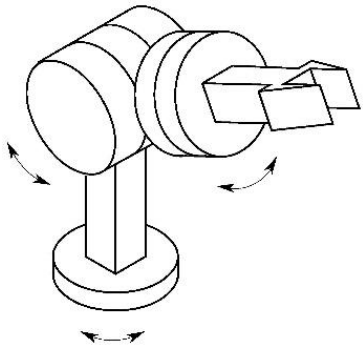

Fig. 2.7 Spherical robotic arm

2.3 Denavit-Hartenberg Parameters

While it is possible to carry out all of the analysis in this chapter using an arbitrary frame attached to each link, it is helpful to be systematic in the choice of these frames. A commonly used

convention for selecting frames of reference in robotic applications is the Denavit-Hartenberg, or D-H convention.

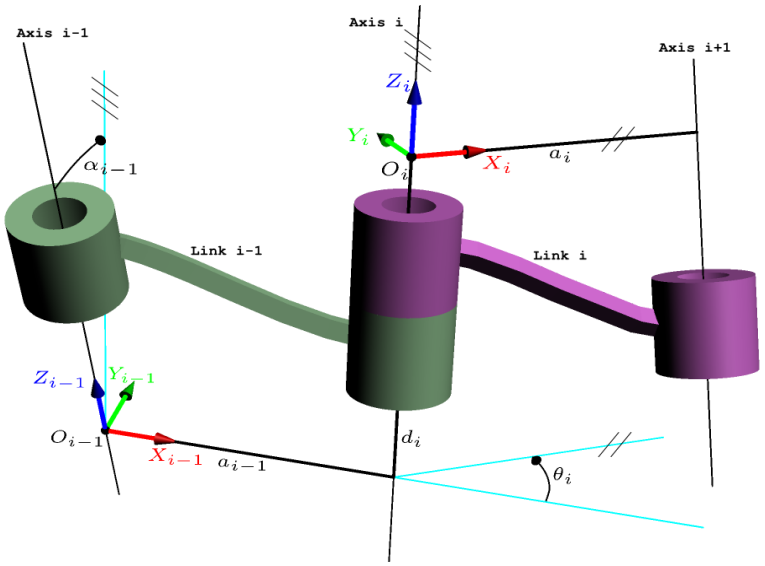

Fig. 2.8 DH parameters Illustration

Table. 2.1 DH parameters Description.

Parameters	Description
Joint Offset (d_i)	Distance between X_i and X_{i+1} along Z_i
Joint Angle (θ_i)	Angle between X_i and X_{i+1} about Z_i
Link Length (a_i)	Distance between Z_i and Z_{i+1} along X_{i+1}
Twist Angle (α_i)	Angle between Z_i and Z_{i+1} about X_{i+1}

Denavit-Hartenberg convention also referred as **DH** parameters. Neighbouring links have a common joint axis between them. One parameter of the interconnection has to do with the distance along this common axis from one link to the next. This parameter is called **link offset**. The offset at joint axis *i* is called d_i. The second parameter describes the amount of rotation about this common axis between one link and its neighbour. This is called the **joint angle** θ_i.

Calculating the position and orientation of the end-effector in terms of the joint variables is called as forward kinematics. Forward kinematics is a technique used in a manipulator in which the joint angles are given and with the help of angles or degrees, by calculating the object's x, y and z coordinates on workspace. Then calculate the end effector's coordinates.

The method through of forward kinematics is DH (Denavit-Hartenberg) convention method.The homogeneous transformation between two coordinate frames with DH parameters $\boldsymbol{\theta_i, d_i, a_i, \alpha_i}$ is,

$A_i = R_{z,\theta_i}\ \text{Trans}_{z,d_i}\ \text{Trans}_{x,\alpha_i}\ R_{x,\alpha_i}$

$$= \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0_i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{A_i} = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(2.1)}$$

Where, $C\theta = Cos\theta$ and $S\theta = Sin\theta$.

Where the four quantities θ_i, a_i, d_i, α_i are parameters associated with link *i* and joint *i*. The four parameters a_i, α_i, d_i, and θ_i in (2.1) are generally given the names **link length**, **link twist**, **link offset**, and **joint angle**, respectively. These names derive from specific aspects of the geometric relationship between two coordinate frames, as will become apparent below. Since the matrix A_i is a function of a single variable, it turns out that three of the above four quantities are constant for a given link, while the fourth parameter, θ_i for a revolute joint and d_i for a prismatic joint, is the joint variable.

CHAPTER 3

HARDWARE AND CIRCUIT DESIGN

Objective of this chapter is to review the hardware considered for implementation of 4DoF manipulator.

3.1 Introduction to Dynamixel Motors

The Dynamixels are also called smart actuators designed to connecting joints on a robot or any other mechanical structure. Dynamixels are designed to be modular and daisy chained on any mechanical design for powerful and flexible robotic movements. The dynamixel is a high performance actuator with a fully integrated DC (Direct Current) Motor + Reduction Gearhead + Controller + Driver + Network, all in one servo module actuator. Programmable, actuator status can be read and monitored through a data packet stream. Durability, sustainability and quality is proven by world class robotic teams from various countries.

Unlike any other Stepper motor, it does has a very high torque which makes it more efficient in the robotic industries. Dynamixels are very much used actuators in any type of manipulator. Dynamixels comes with its own U2D2 controller which controls the flow of these motors. It is programmable and the programming language that is used is Python. Different type of software's can also be included to be compatible with these motors like: RoboPlus 1.0, RoboPlus 2.0, Dynamixel SDK and RoboPlus Mobile etc.

3.1.1Dynamixel Types and Specifications

Dynamixel comes in many types and it has series which is unique from others with a little bit modification. It comes in A-series, M-series, R-Series, D-Series and X-Series etc. In this project the focused and used is A-series.

Fig. 3.1 Different types of Dynamixel motor series.

3.1.2 Dynamixel Pin Assignment

The connector pin assignments are as the following. The two connectors on the Dynamixel are connected pin to pin.
(Note: The pin number of connector's edge cut side is PIN1)

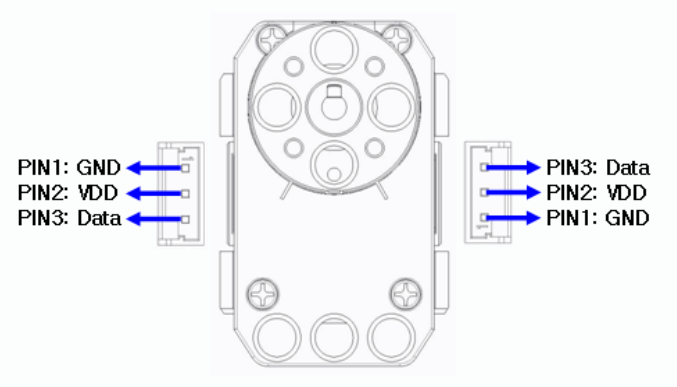

Fig. 3.2 Pin out configuration for Dynamixel motor.

Dynamixel motor are more advanced gadgets, it can do Write and also Read signals with single data connection. Dynamixel precision is more accurate and precise than other servos, it can be calibrated by gyro sensor equipped in it.

3.1.3 Dynamixel Series with their specifications.

A-Series Compatible Products: Controller: CM-5, CM-510, CM530, CM-700, OpenCM9.04 (+ OpenCM485 Expansion Board), OpenCR.

Interface: USB2Dynamixel, U2D2.

Table. 3.1 A-series Dynamixel motors Specification.

Dynamixel A-Series	AX-12W	AX-12/AX-12+/AX-12A	AX-18A/18F
Weight	52.9g	53.5g(AX-12/AX-12+),54.6g(AX-12A)	54.5g (AX-18F), 55.9g(AX-18A)
Dimension	32*50*40mm	32*50*40mm	32*50*40mm
Resolution	0.29°	0.29°	0.29°
Gear Reduction Ratio	32:01:00	254:01:00	254:01:00
No load speed	470 rpm	59rpm	97rpm
Operating Angle	0° to 300°	0° to 300°	0° to 300°
Operating Temperature	5°C to 70°C	5°C to 70°C	5°C to 75°C
Voltage	9 to 12V	9 to 12V	9 to 12V
Command Signal	Digital Packet	Digital Packet	Digital Packet
Protocol Type	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity
Link(Physical)	TTL level multi drop	TTL level multi drop	TTL level multi drop
ID	254-ID (0~253)	254-ID (0~253)	254-ID (0~253)
Baud Rate	57600 bps ~1Mbps	7343 bps ~1Mbps	7343 bps ~1Mbps
Feedback	Position,Temperature,Load,Input Voltage	Position,Temperature,Load,Input Voltage	Position,Temperature,Load,Input Voltage
Material	Engineering Material	Engineering Material	Engineering Material
Stall Torque		15.3 kg cm(at 12.0V, 1.5A)	18.3 (at 12.0V, 2.2A

3.2 Raspberry Pi

The Raspberry pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse [5]. A typical Raspberry pi board is given below in Figure 3.3. It is capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

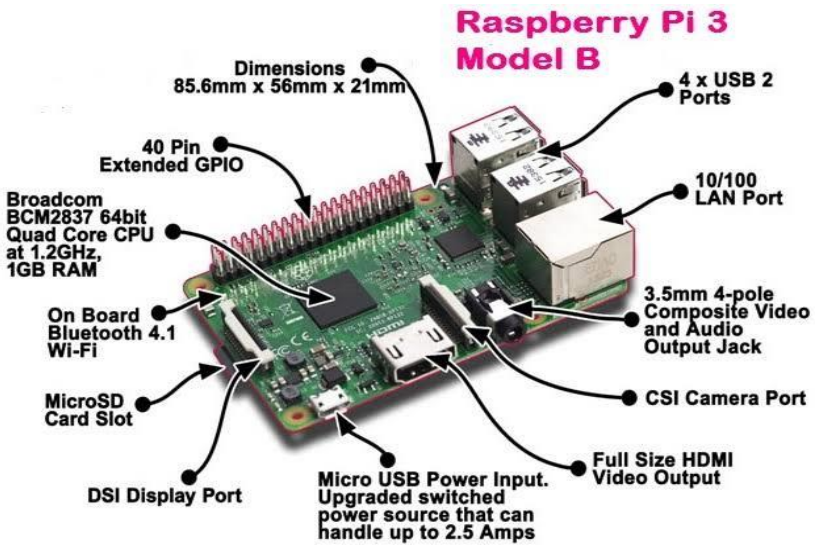

Fig. 3.3 A Typical Raspberry Pi 3 Board

The Raspberry pi is a small, inexpensive personal computer. Although it lacks the capacity for memory expansion and can't accommodate on-board devices such as CD, DVD (Digital Versatile Disc), hard drives, it has everything a simple personal requires. That is, it has two USB ports, and Ethernet port. There is special HDMI video, and even an Audio connector for sound. The Raspberry Pi has ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras.

Not larger than the deck of all playing cards, Raspberry pi has number of ports for connecting devices [6]. This section presents briefly about the components of the pi board that are used for the manipulator.

3.2.1 USB Port

Raspberry Pi B+ comes with 4 USB ports; USB is one of the most common methods for connecting peripherals and storage devices to a computer. The Raspberry Pi comes equipped with two of them, allows to hook up a keyboard and mouse when get started and a micro USB port for powering the device. The USB ports on the Pi are USB 2.0 and provided everything normally would be needed from a USB port with one small drawback. The maximum recommended current draw is 100mA for short, compared to the possible 500mA of a normal USB port in device-negotiated mode. To draw more than 100mA look at a powered USB hub or power the device

externally. Devices like a USB flash drive should be fine; a 2.5-inch external self-powered hard drive will not be fine on the other hand.

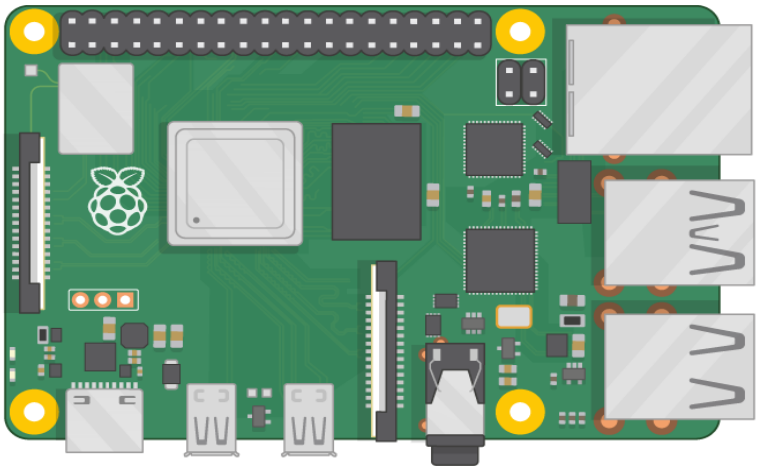

Fig. 3.4 Top View of Raspberry Pi [B+]

3.2.2 Ethernet Port

The Ethernet port is the Raspberry pi's main gateway to communicating with other devices and the internet. Ethernet port is used to plug Raspberry pi into a home router such as the one currently use to access internet, or a network switch if one setup is already done.

This Ethernet connector is a standard RJ45 (Registered Jack 45) connector. Simply plug the standard ethernet patch cable into the socket and connect the other end to either router or switch; if that is the to setup home network.

3.2.3 HDMI Port

The High Definition Multi-media Interface (HDMI) port allows the Raspberry pi to be hooked up to high-definition televisions and monitors that support technology. This provides an additional option to the composite RCA (Radio Corporation of America) port for video and additionally supports audio.

3.2.4 Power Connectors

The power connectors is a micro USB socket that is wired to pass the 5-volt (V) *direct current* (DC) lines from a micro USB plug, also shown in the Figure 3.3. No data connections are wired to this socket. Available any smart phone charger that has a micro USB connector can also be used or use the power supply that came with the Raspberry pi kit.

3.2.5 Micro SD card port

The main storage mechanism of the Raspberry pi is via the SD card port. The SD card will be where the operating can be installed and it will act as basic hard disk. Of course, this storage can be expanded upon using the USB ports.

3.2.6 SD Card Slot

SD card slot can be viewed by flipping up the Pi, because it is located at the rear end of the Pi. Raspberry pi board doesn't have built in storage but external storage device has to be connected. Normally 8GB SD card is used but it can be vary according to the application and operating system.

3.2.7 GPIOs

Row of GPIOs (General Purpose Input/Output) is powerful features of the Raspberry pi located along the top edge of the board as shown in Figure 3.5. Forty pins in B+ pins instead of 26 in A and B. The top/first 26 pins match the original layout, 9 additional GPIOs are used for ground and power and 2 EEPROM (Electronically Erasable Programmable Read Only Memory) Plate identification pins, GPIOs are the main way of connecting with other electronic board such as the Arduino. As the name suggests, the GPIO pin can be accepted input and output commands and thus can be programmed on the Raspberry Pi. Pins can be programmed to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer device.

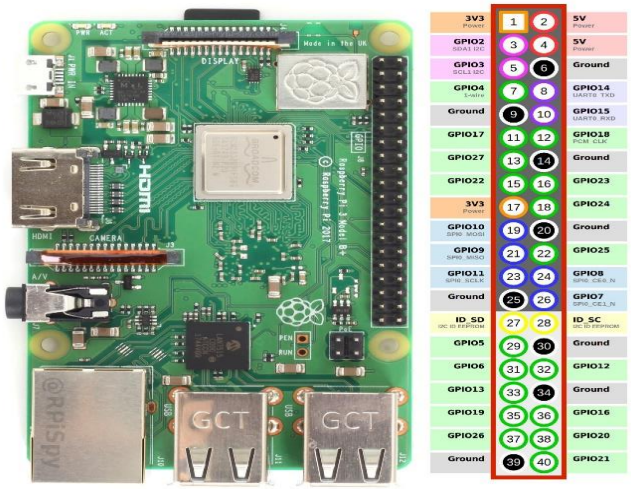

Fig 3.5 40 GPIOs of Raspberry Pi [B+]

3.3 Pneumatic Pump

In robotic arm there is an end effector which can perform pick and place the object one end to other end, it can be picking the objects with gripper that is attached at the end effector of robotic arm. We can also pick and place the objects with in workspace with pneumatic system. A DC sucking or pumper motor is attach with robotic arm, its gripper is mounted on end effector of

robotic arm, by the controlling circuitry it can pick and place the object. We have designed a circuitry to energize the relay with GPIO Controlling pins and here we have used few software which are mentioned below and with the help of those our hardware is being operated.

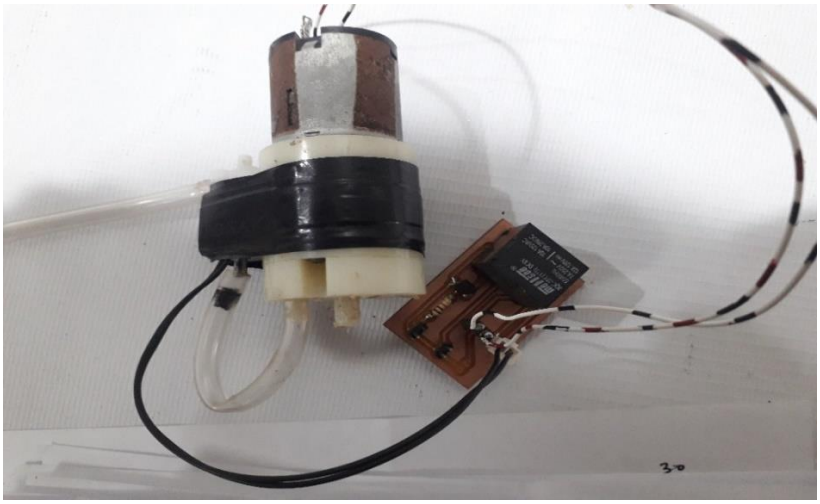

Fig 3.6 Pneumatic Pump with control circuitry.

3.3.1 Proteus Design Suite

This software is used to draw the schematic of any circuit for simulation purposes in order to avoid any electrical hazard before doing it practically. Proteus software is used widely to design the schematic of any type of electronic circuit and after that it can be converted into PCB layout. Below is the running schematic circuit in proteus which contains relay, transistor, and DC motor with pneumatic valve, three batteries and resistor.

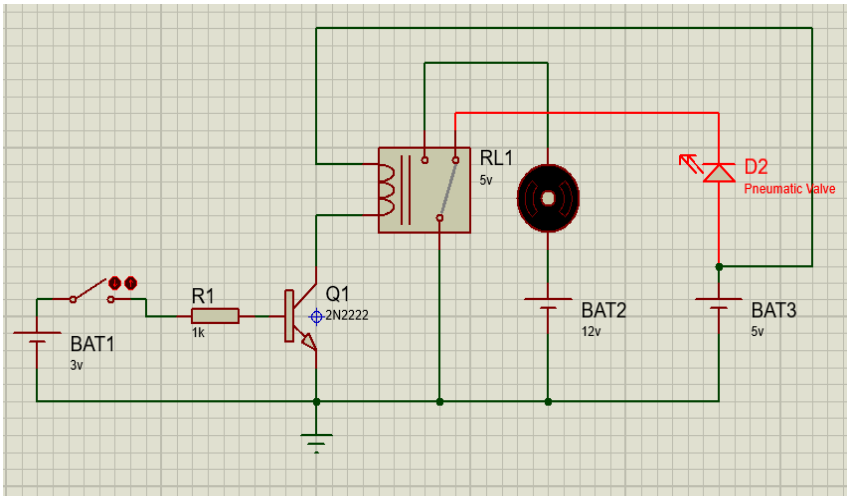

Fig. 3.7 Schematic of relay controlling circuit in Proteus.

In (figure 3.7), battery no:1 of 3 volts as a GPIO input pin to energize the transistor, when GPIO pin is (High), the VC(Collector voltage) of transistor becomes zero, so the relay will be energized

with 5v of the Battery no:3 and built in DC Motor will be started, when the GPIO pin is (Low), the VC voltage will be 3.3v that will de-energize the relay and the pneumatic valve will be activated which will help the sucker to easily leave the object at the required place, The PCB(Printed Circuit Board) of the circuit is designed in eagle software.

3.3.2 EAGLE 7.5.0

Eagle (Easily Applicable Graphical Layout Editor) is a scriptable electronic design automation application with schematic design, PCB layout, auto-router and computer-aided manufacturing features. A software which draws the schematic and it also converts it into PCB design which can be printed as well after components being routed properly. There are two types of routing first is upper layer routing and second one is lower layer routing. In (figure 3.8), the schematic of a relay controlling circuit in eagle is designed after proper wiring between components and ready to switch it to the board.

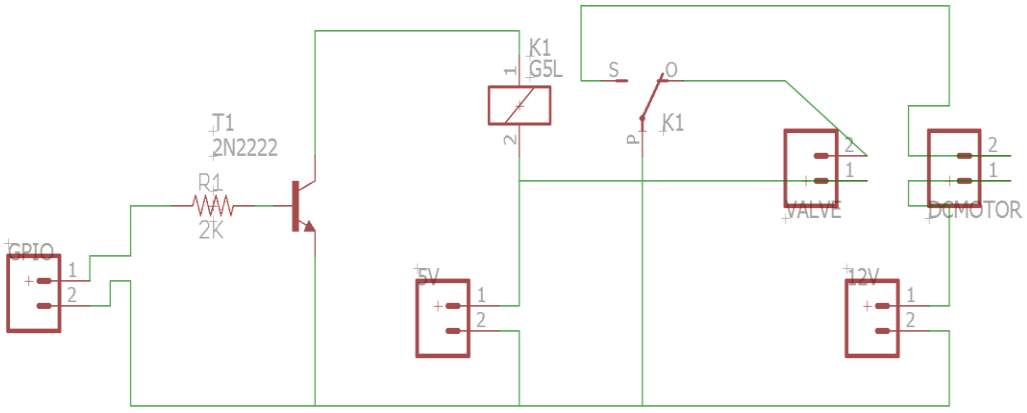

Fig. 3.8 Schematic of a relay controlling circuit in Eagle.

Eagle's PCB Board file of a relay controlling circuit:

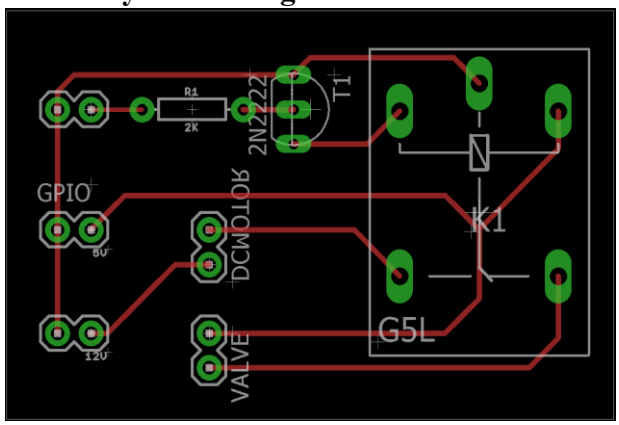

Fig. 3.9 PCB board file of a relay controlling circuit in EAGLE.

(figure 3.9), is the PCB board file of a relay controlling circuit which is switched from schematic to

PCB design in Eagle. After components are being routed through upper layer and the PCB layout is ready to print.

Eagle schematic to board:

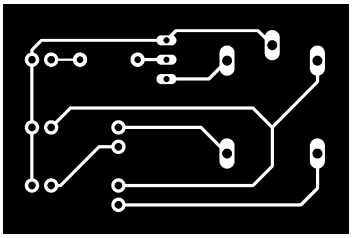

Fig. 3.10 Traces for PCB board

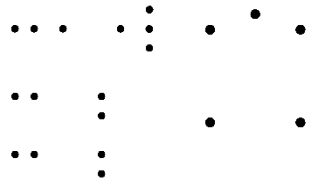

Fig. 3.11 Drills for PCB board

Fig. 3.12 Outline for PCB board

After successfully converting PCB layout to the board, now above figure shows the traces, drills and outline for our PCB board of a relay controlling circuit which is done by a PCB printing machine.

3.3.3 Roland Machine:

After processing the eagle PCB Board files along with their traces, drills and outline files.
Making the Eagle board file, that process the data for machine PCB printing, roland machine takes the .rml extension files, then making the rml file. From the Board file we Export the image file with 2000 dpi (dot per inch) resolution and check the image in monochrome.
Open the png file in paint and separate the drills and outlines from the traces files, for rml file, process the image in fabmodule.org by importing the traces image and select the Roland mill(.rml) output format and also select the PCB traces(1/64) in process, now in input change the dpi(2000), and in output select the machine (SRM-20), speed (mm/s) 4, x0, y0, z0 (0,0,0)mm then click on the calculate button, it will calculate the coordinates for the PCB milling machine.

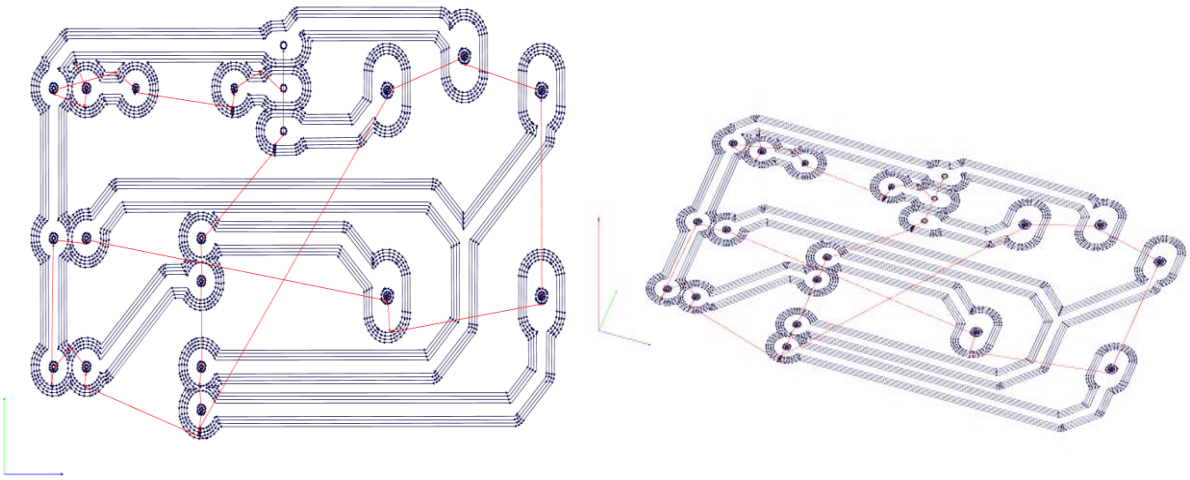

Fig. 3.13 Traces rml Extension files

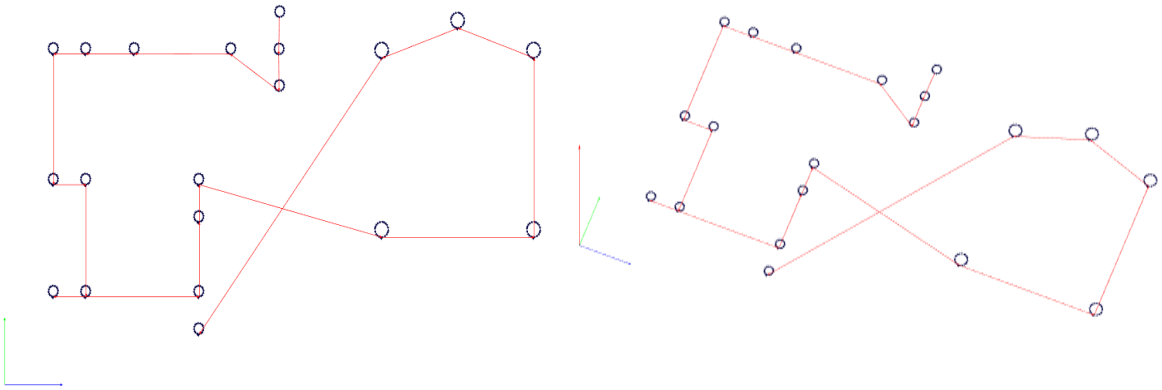

Fig. 3.14 Drills rml extension files

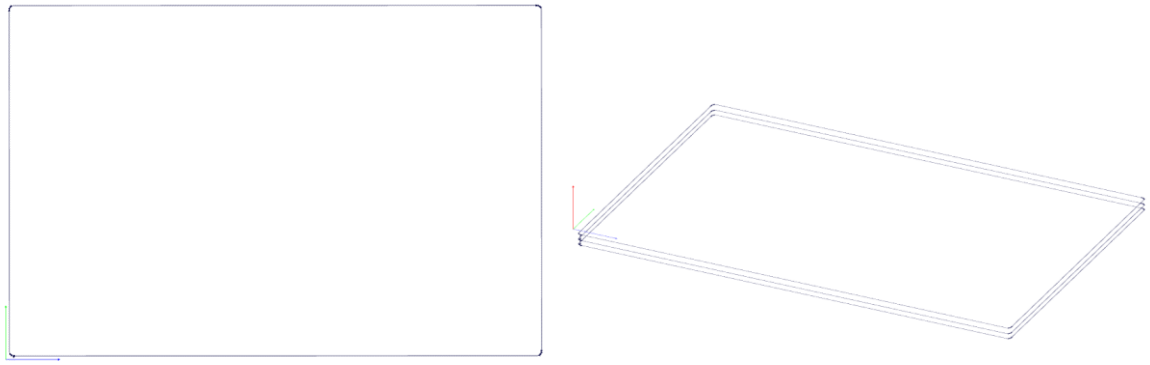

Fig. 3.15 Outline rml extension files

Now the Printed PCB design with assembled components is shown in (figure 3.16),

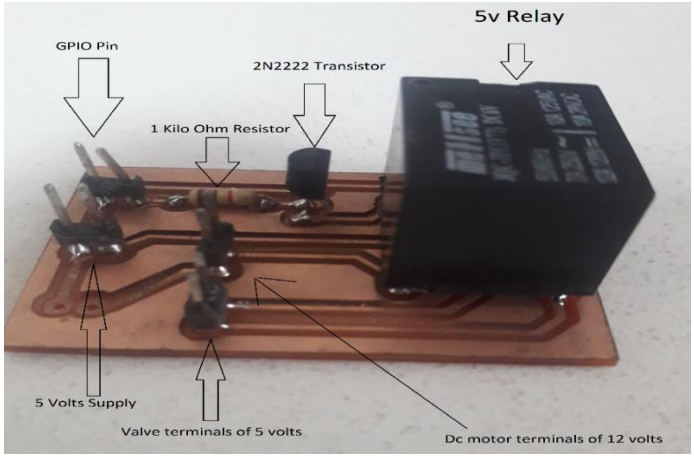

Fig. 3.16 Printed PCB design with assembled components.

In (figure 3.16), controlling circuit is printed through PCB machine and components including: relay, resistor, 2N2222 transistor, GPIO Pin, 5 and 12 volt batteries, 5volts valve terminals and DC motor terminals of 12 volts are inserted in PCB design to make controlling circuit in working condition.

3.4 Workspace:

Workspace of Arm depend on manipulator environment, manipulator can be installed in fixed environment or it can be in mobile environment. The workspace of a robot is the volume comprising the coordinates of all points that the end-effector can reach. (Figure 3.17), workspace for manipulator.

3.4.1 Main characteristics of the workspace:

In the computation of robot workspace, what is most important is its **shape and volume** (dimensions and structure). Both aspects have a significant importance due to their impact on the design and manipulability of the robot.

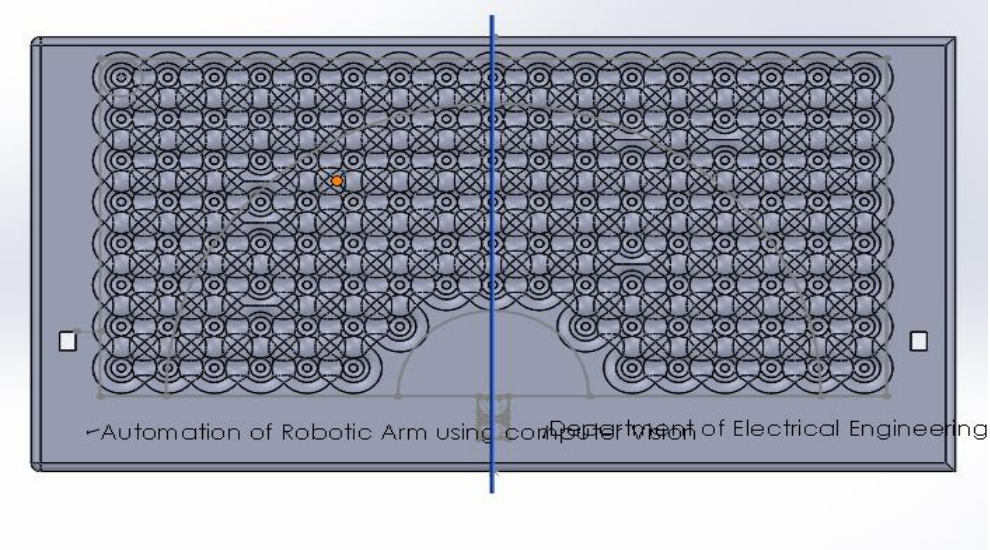

Fig. 3.17 Workspace of 4DOF Robotic Arm.

For using a robot, the exact knowledge about the shape, dimensions and structure of its workspace is important since:

- The shape is important for the definition of the environment where the robot will work.
- The dimensions are important for determination the reach of the end-effector.
- The structure of workspace is important for assuring kinematic characteristics of the robot which are in relation with the interactions of the robot to its environment.

For picking up and placing the object is ultimate goal for which we have designed following chain which illustrate both phenomenon.

3.5 To Pick And Place Object

Picking phase moves robot to object, picks it up, and moves to a safe position, as shown in the following diagram:

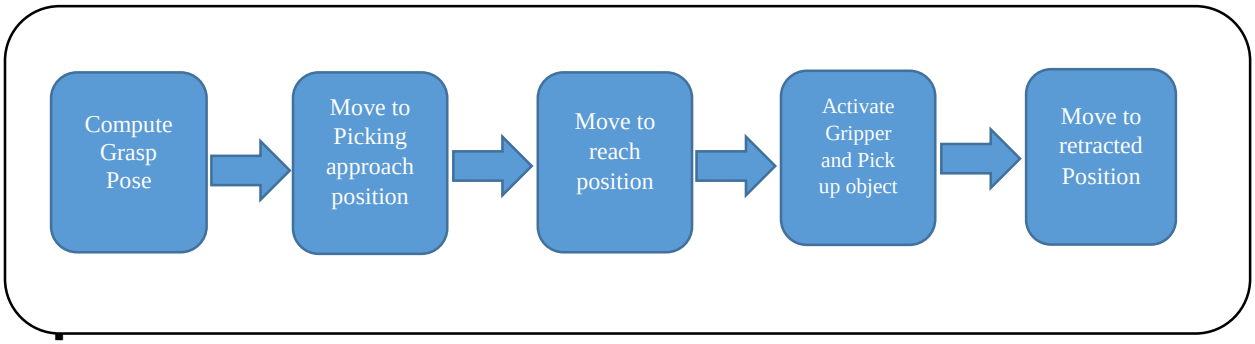

Fig 3.18 Robotic arm picking up object.

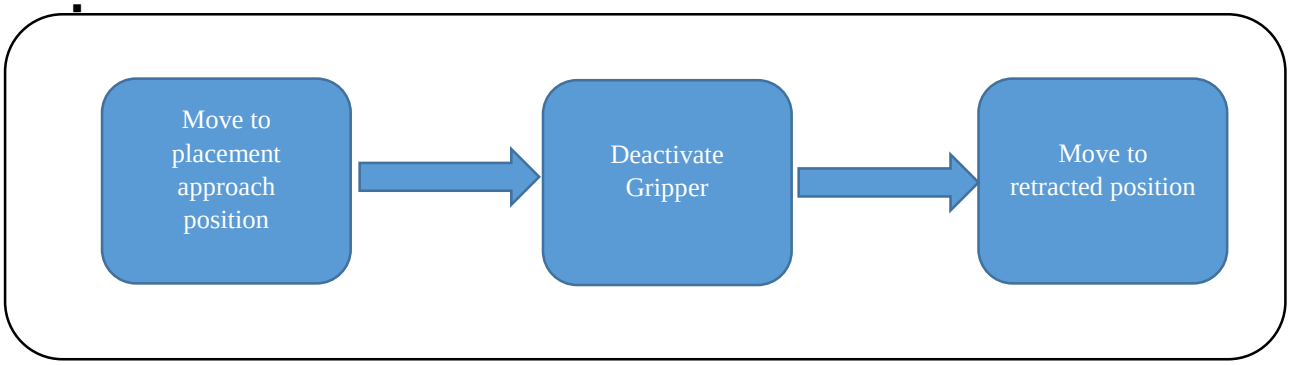

Fig 3.19 Robotic arm placing object.

CHAPTER 4

FORWARD AND INVERSE KINEMATICS OF 4DOF MANIPULATOR

Objective of this chapter is to introduce kinematics of a 4DOF manipulator and DH parameters along with forward and inverse equations.

4.1 Introduction to Kinematics:

Kinematics is the branch of science which is associated with the motion of objects without regarding the forces which creates the motion. In kinematics there are study of position, velocity, acceleration and all higher order derivatives of position variables with respect to time and any other variable. The study of the kinematics of manipulators are all based on time-based and geometrical properties of motion.

In this chapter we are considering orientation and position of 4 DOF manipulator links in static position.

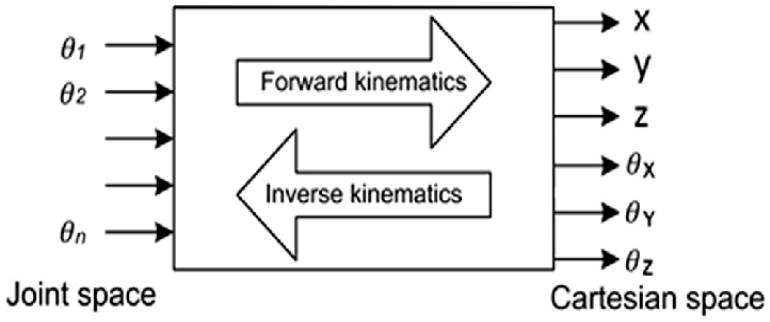

Fig. 4.1 Forward and Inverse kinematics Relationship.

In (figure 4.1), clearly describing the relationship between forward kinematics and inverse kinematics. Furthermore illustrating these terms as follows,

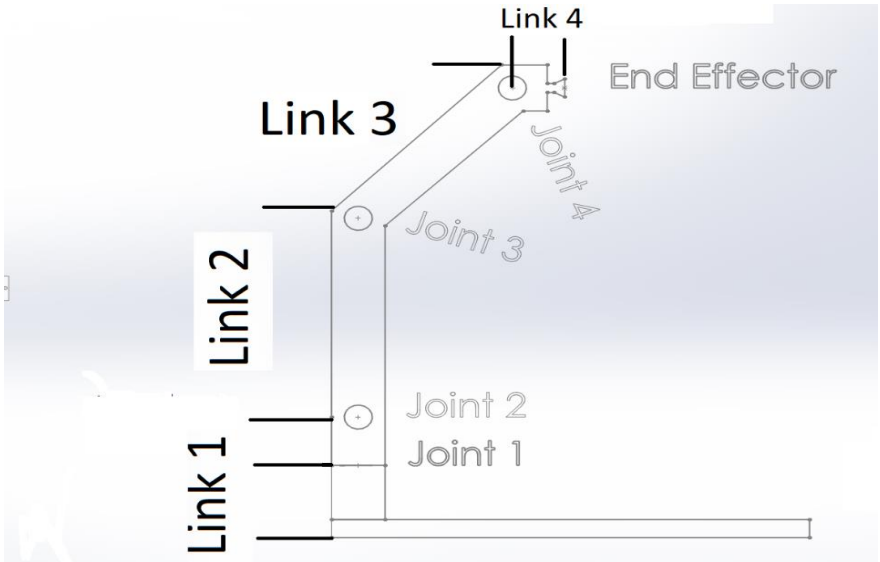

Fig.4.2 4DOF Articulated Manipulator.

4.2 Forward Kinematics of Manipulator:

A very basic problem in the study of mechanical manipulation is called forward kinematics. This is the static geometrical problem of computing the position and orientation of the end-effector of the manipulator. Specifically, given a set of joint angles, the forward kinematic problem is to compute the position and orientation of the tool frame relative to the base frame. Sometimes, we think of this as changing the representation of manipulator position from a joint space description into a Cartesian space description.

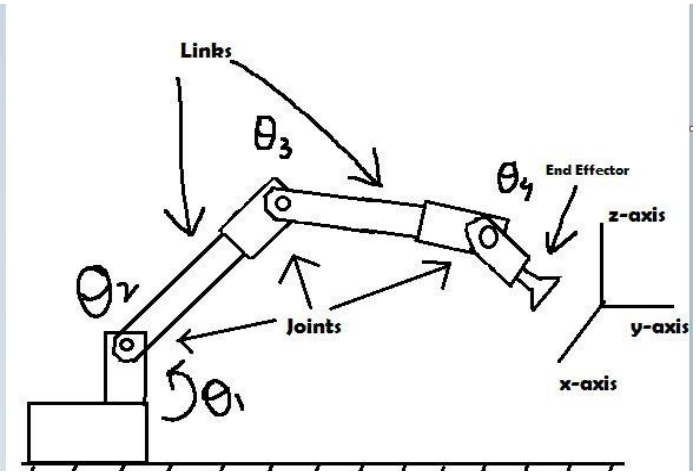

Fig. 4.3 4DOF Articulated Manipulator.

4.3 Forward Kinematics Equations:

For our 4DOF Articulated Manipulator with help of following DH parameter table, it can be calculate Forward Kinematics Equations of 4DoF manipulator.
Using equation 2.1 to derive Forward kinematic equation.

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Called as Homogenous transformation matrix, for any given stage.

4.3.1 DH Parameters:

Table. 4.1 Calculated DH parameters.

I	α_i	a_i	d_i	θ_i
1	90	0	D	θ_1
2	0	L_2	0	θ_2
3	0	L_3	0	θ_3
4	90	0	0	θ_4

Above DH parameter table is to calculated Forward kinematics equations for our 4DOF Articulated Manipulator.

$$ {}^{0}_{1}T = \begin{bmatrix} C\theta_1 & 0 & -S\theta_1 & 0 \\ S\theta_1 & 0 & -C\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.1),} \qquad {}^{1}_{2}T = \begin{bmatrix} C\theta_2 & -S\theta_1 & 0 & l_2 * C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & l_2 * S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.2)} $$

$$ {}^{2}_{3}T = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & l_3 * C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & l_3 * S\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.3),} \qquad {}^{3}_{4}T = \begin{bmatrix} C\theta_4 & 0 & S\theta_4 & 0 \\ S\theta_4 & 0 & -C\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.4)} $$

By multiplying all the matrices **(4.1, 4.2, 4.3, 4.4)**, we get overall transformation matrix **(4.5)** that is,

$$ {}^{0}_{4}T = A_1 * A_2 * A_3 * A_4 $$

$$ \mathbf{{}^{0}_{4}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T} \quad \textbf{(4.5)} $$

And also multiply Identity matrix $\boldsymbol{I}$ with Position matrix $\boldsymbol{T_P}$.

$$ \boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{T_P} = \begin{bmatrix} 0 \\ 0 \\ l_4 \end{bmatrix} $$

After embedding single row at end of Transformation matrix to make it 4x4 matrix as given below.

$$ T = [0 \quad 0 \quad 0 \quad 1] $$

Therefore 4x4 Homogeneous Transformation matrix will look like,

$$Tp=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & l_4\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Therefore equation **(4.5)** redefined as;

$$\mathbf{{}^0_4T = {}^0_1T * {}^1_2T * {}^2_3T * {}^3_4T * T_p} \quad \textbf{(4.5)}$$

Here we have the final equations for Positions *Px, Py* and *Pz.*

$$\boldsymbol{Px} = l_4(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3)) + l_2C_1C_2 + l_3C_1C_2C_3 - l_3C_1S_2S_3 \quad \textbf{(4.6)}$$

$$\boldsymbol{Py} = l_4(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1)) + l_2C_2S_1 + l_3C_2C_3S_1 - l_3S_1S_2S_3 \quad \textbf{(4.7)}$$

$$\boldsymbol{Pz} = d_1 + l_2S_2 - l_4(C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + C_3S_2)) + l_3C_2S_3 + l_3C_3S_2 \quad \textbf{(4.8)}$$

4.4 Inverse Kinematics of Manipulator

The inverse kinematics problem of the serial manipulators has been studied for many decades. It is needed in the control of manipulators. Solving the inverse kinematics is computationally expansive and generally takes a very long time in the real time control of manipulators. Tasks to be performed by a manipulator are in the Cartesian space, whereas actuators work in joint space. Cartesian space includes orientation matrix and position vector. However, joint space is represented by joint angles. The conversion of the position and orientation of a manipulator end-effector from Cartesian space to joint space is called as inverse kinematics problem.

4.4.1 Piper's Solution

Another strategy for determining inverse kinematics is Piper's solution. Applied this technique to solve inverse kinematics of 4DOF Articulated Manipulator. Recall equation **(4.5)**

$${}^0_4T = {}^0_1T * {}^1_2T * {}^2_3T * {}^3_4T * T_P$$

This technique states that replace *Px, Py* and *Pz* equations which are found in forward kinematics with letters *x, y* and *z* in ${}^0_4T$ transformation matrix. And multiply inverse of all 4x4 transformation matrices both side to **(4.5)** equation as follows.

$${}^0_4T = \begin{bmatrix}3 & \text{x} & 3 & x\\ & & & y\\ & & & z\\ 0 & 0 & 0 & 1\end{bmatrix}$$, here 3x3 is rotation matrix *x, y* and *z* is replaced with *Px, Py* and *Pz.* Shown in equation **(4.9)**.

$$
{}^{0}_{4}T = \begin{bmatrix} -C_4(C_1S_2S_3 - C_1C_2C_3) - S_4(C_1C_2S_3 + C_1C_3S_2) & S_1 & C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3) & x \\ -C_4(S_1S_2S_3 - C_2C_3S_1) - S_4(C_2S_1S_3 + C_3S_1S_2) & -C_1 & -C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1) & y \\ C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3) & 0 & S_4(C_2S_3 + C_3S_2) - C_4(C_2C_3 - S_2S_3) & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.9)}
$$

Further, applying piper's technique to get $\boldsymbol{\theta_1}$, it states that multiply ${}^{0}_{1}T^{-1}$ frame both sides of equation **(4.5).**

$$
{}^{0}_{1}T^{-1} * {}^{0}_{4}T = ({}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T) * {}^{0}_{1}T^{-1}
$$

From the forward kinematics of the manipulator we know ${}^{0}_{1}T$ as defined in **(4.1)** therefore we can can calculate its inverse as;

$$
{}^{0}_{1}T^{-1} = \begin{bmatrix} C\theta_1 & S\theta_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ S\theta_1 & -C\theta_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.10)}
$$

Where ${}^{0}_{1}T^{-1}$ frame is calculated in Matlab using command inv(${}^{0}_{1}T$). And it can be calculated manually as well.

$$
\begin{bmatrix} b11 & b12 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \end{bmatrix} = \begin{bmatrix} c11 & c12 & c13 & c14 \\ c21 & c22 & c23 & c24 \\ c31 & c32 & c33 & c34 \\ c41 & c42 & c43 & c44 \end{bmatrix} \quad \textbf{(4.11)}
$$

Equation **(4.11)** states that elements of $\mathbf{b_{mn}}$ matrix equal to elements of $\mathbf{c_{mn}}$ Matrix.

Where $\mathbf{b_{mn}}$ as:

$b_{11} = -S_1(C_4(S_1S_2S_3 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2)) - C_1(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2))$

$b_{12} = 0$

$b_{13} = S_1(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1)) + C_1(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3))$

$b_{14} = xC_1 + yS_1$

$b_{21} = C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)$

$b_{22} = 0$

$b_{23} = S_4(C_2S_3 + C_3S_2) - C_4(C_2C_3 - S_2S_3)$

$b_{24} = z - d_1$

$b_{31} = C_1(C_4(S_1S_2S_3 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2)) - S_1(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2))$

$b_{32} = C_1^2 + S_1^2$

$b_{33} = S_1(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3)) - C_1(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 + C_2C_3S_1))$

$b_{34} = xS_1 - yC_1$

$b_{41} = 0$
$b_{42} = 0$
$b_{43} = 0$
$b_{44} = 1$

Where $\mathbf{c_{mn}}$ as:

$c_{11} = C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + C_3S_2)$
$c_{12} = 0$
$c_{13} = C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)$
$c_{14} = l_2C_2 + l_3C_2C_3 - l_3S_2S_3$
$c_{21} = C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)$
$c_{22} = 0$
$c_{23} = S_4(C_2S_3 + C_3S_2) - C_4(C_2C_3 - S_2S_3)$
$c_{24} = l_2S_2 + l_3C_2S_3 + l_3C_3S_2$
$c_{31} = 0$
$c_{32} = 1$
$c_{33} = 0$
$c_{34} = 0$
$c_{41} = 0$
$c_{42} = 0$
$c_{43} = 0$
$c_{44} = 1$

By equating equation $\mathbf{b_{34}}$ with $\mathbf{c_{34}}$, to get $\boldsymbol{\theta_1}$;

$b_{34} = xS_1 - yC_1$, $c_{34} = 0$

$x\sin\theta_1 - y\cos\theta_1 = 0$	$x\sin\theta_1 = y\cos\theta_1$
$\frac{\sin\theta_1}{\cos\theta_1} = \frac{y}{x}$	$\tan\theta_1 = \frac{y}{x}$

$$\boldsymbol{\theta_1} = atan2(\tfrac{y}{x}) \qquad \textbf{(4.12)}$$

$\boldsymbol{\theta_2}$ is calculated as;

$$\boldsymbol{\theta_2} = atan2\left(\frac{\frac{a^2+b^2}{34.2}}{\pm\sqrt{a^2+b^2-\left(\frac{a^2+b^2}{34.2}\right)^2}}\right) - atan2\left(\frac{b}{a}\right) \qquad \textbf{(4.13)}$$

$\boldsymbol{\theta_3}$ is calculated as;

$$\boldsymbol{\theta_3} = atan2\left(\frac{cos\theta_2(a+b)+sin\theta_2(a-b)-17.1}{\pm\sqrt{(17.1)^2+(17.1)^2-(cos\theta_2(a+b)+sin\theta_2(a-b)-17.1)^2}}\right) - atan2(\frac{17.1}{17.1}) \quad \textbf{(4.14)}$$

And $\boldsymbol{\theta_4}$ is calculated as;

$$\boldsymbol{\theta_4} = \varphi - \theta_2 - \theta_3 \quad \textbf{(4.15)}$$

Solutions in detail for remaining $\boldsymbol{\theta_2}$, $\boldsymbol{\theta_3}$ and $\boldsymbol{\theta_4}$ refer Appendix C.

Forward Kinematics

$Px = l_4(C\theta_4(C\theta_1C\theta_2S\theta_3 + C\theta_1C\theta_3S\theta_2) - S\theta_4(C\theta_1S\theta_2S\theta_3 - C\theta_1C\theta_2C\theta_3)) + l_2C\theta_1C\theta_2 + l_3C\theta_1C\theta_2C\theta_3 - l_3C\theta_1S\theta_2S\theta_3$
$Py = l_4(C\theta_4(C\theta_2S\theta_1S\theta_3 + C\theta_3S\theta_1S\theta_2) - S\theta_4(S\theta_1S\theta_2S\theta_3 - C\theta_2C\theta_3S\theta_1)) + l_2C\theta_2S\theta_1 + l_3C\theta_2C\theta_3S\theta_1 - l_3S\theta_1S\theta_2S\theta_3$
$Pz = d_1 + l_2S\theta_2 - l_4(C\theta_4(C\theta_2C\theta_3 - S\theta_2S\theta_3) - S\theta_4(C\theta_2S\theta_3 + C\theta_3S\theta_2)) + l_3C\theta_2S\theta_3 + l_3C\theta_3S\theta_2$

Inverse Kinematics

$\theta_1 = atan2(\frac{y}{x})$
$\theta_2 = atan2\left(\frac{\frac{a^2+b^2}{34.2}}{\pm\sqrt{a^2+b^2-\left(\frac{a^2+b^2}{34.2}\right)^2}}\right) - atan2\left(\frac{b}{a}\right)$
$\theta_3 = atan2\left(\frac{cos\theta_2(a+b)+sin\theta_2(a-b)-17.1}{\pm\sqrt{(17.1)^2+(17.1)^2-(cos\theta_2(a+b)+sin\theta_2(a-b)-17.1)^2}}\right) - atan2(\frac{17.1}{17.1})$
$\theta_4 = \varphi - \theta_2 - \theta_3$

CHAPTER 5

COMPUTER VISION

Objective of this chapter is to introduce computer vision and python programming language along with multiple colour detection process.

5.1 Introduction to Computer Vision

Computer vision is a subfield of Artificial Intelligence where the goal is to build a computer replicating the visual intelligence of human brain. Machine Learning is a generic term for teaching machines anything, but computer vision specifically deals with visual data. In machine learning, we deal more with statistical tools whereas computer vision could include both — statistical as well non-statistical tools.

As humans, we perceive the three-dimensional structure of the world around us with apparent ease. Computer vision, basically, is to infer different factors such as camera model, lighting, colour, texture, shape and motion that affect images and videos, from visual inputs. A word, computer vision is an inverse processing of the forward process of image formation and graphics. In this sense, as many people agree, vision is a much more challenging problem than computer graphics, because it is full of uncertainties.

5.2 Python

Python is independent and open source language. It is general purpose widely used, high level programming. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

5.2.1 Python Installation

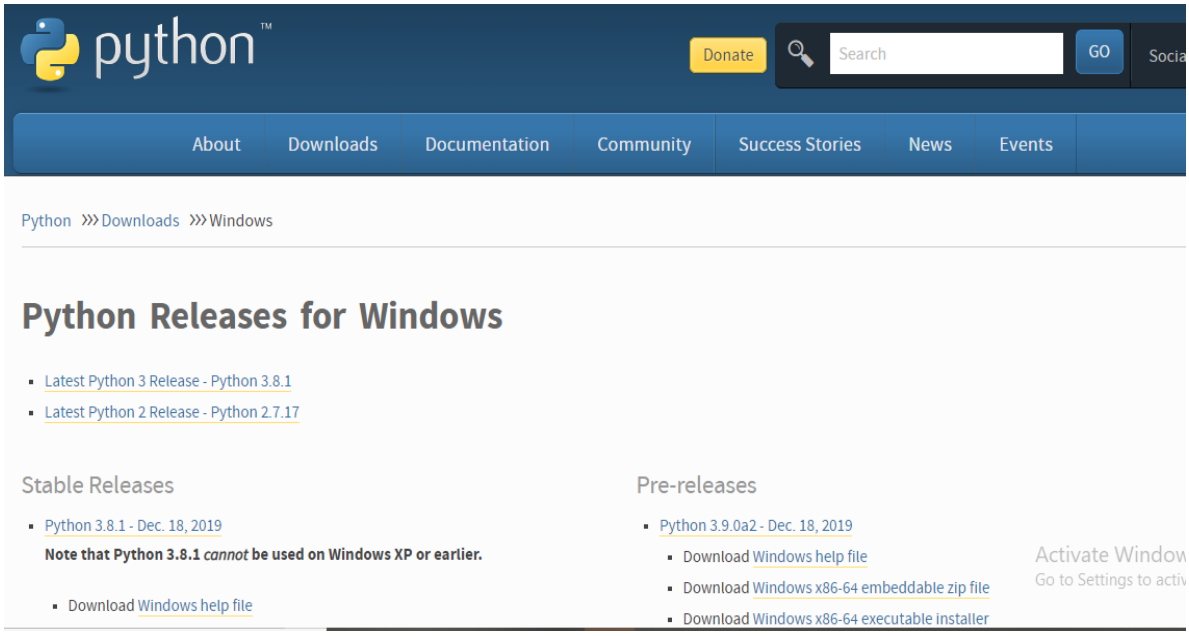

Fig.5.1 .Python Home page.

Step-1: Install python on Computer. To download latest version of Pyhton, visit wwww.python.org and select your operating system (Windows/Linux/Mac). Here it is installed for windows. For window users installation is very simple follow the given GUI (Graphical User Interface).

Step-2: Pycharm Insatllation

Pycham is IDE for python programming, its installation is also very simple. Just go to page https://www.jetbrains.com/pycharm/download/#section=windows as shown.

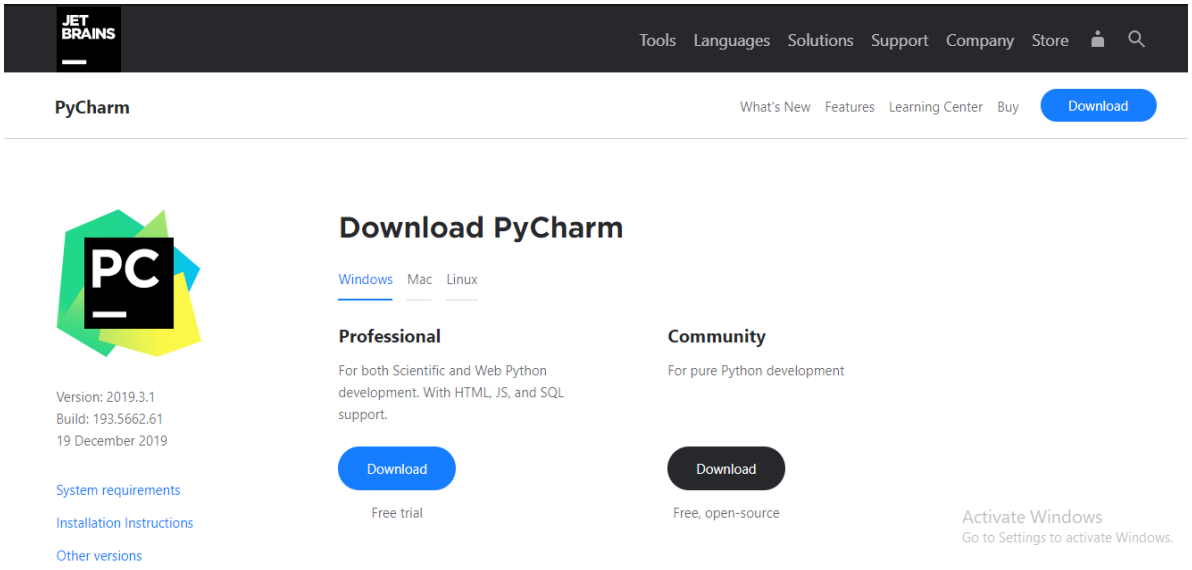

Fig. 5.2 Pycharm IDE Home page.

And download free trial version of pycharm IDE as shown in Fig.3.

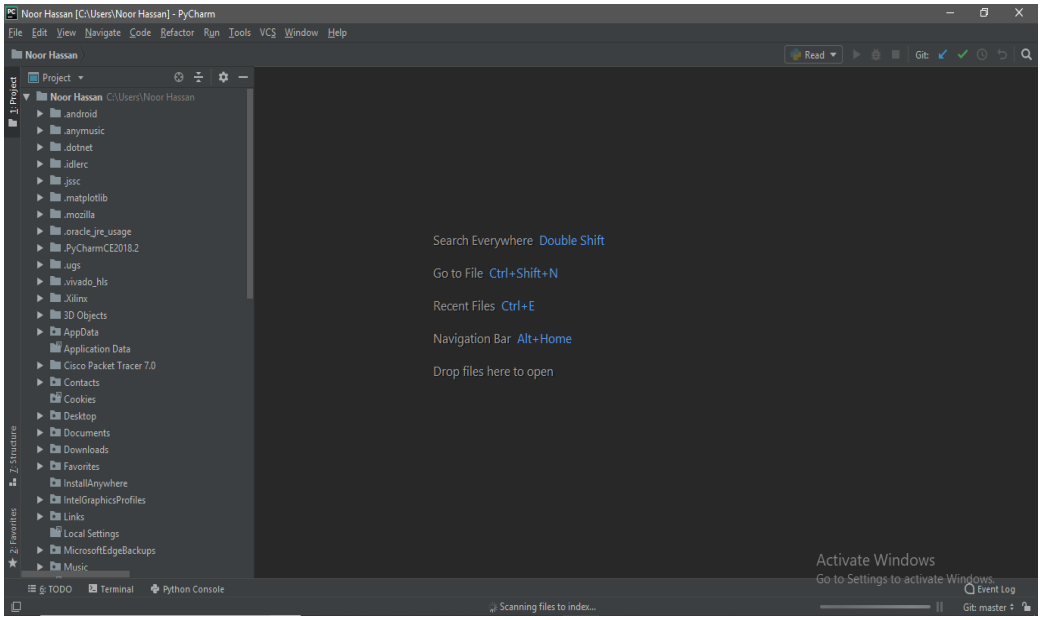

Fig. 5.3 Pycharm Build up.

Then open pycharm IDE and you would have this type of screen in front. Just click on File and create new project with any name.

Step-3: Intalling Modules and Libraries

Following are the modules and libaries installed for this project. Open the cmd command in search bar.

Install pip and other commands in command prompt like;

Install opencv

```
>pip install opencv
```

Install numpy

```
>pip install numpy
```

Similarly, install all required modules and libraries same way as these two libraries are installed.

After installing all required libraries then move to start coding.

5.3 Multiple Colors Detection: To detect multiple colors simulteneously is not an easy task, because if we extend scale then many unwanted colors also gets detected. To prevent these unwanted colors and to be specific, different techniques and algorithms are used.

Here we have detected two colors; Red and Green Specifically.

```
import time
import cv2
import numpy as np
```

Here we import the required modules. 'cv2' imports OpenCV and numpy is imported as np.

```
cap= cv2.VideoCapture(0)
```

Here we use cv2's video capture method to start the webcam and record. It is stored in variable 'cap'. Here the argument '0' defines default webcam. if your device has a secondary camera you can use it by replacing '0' with '1'.

```
while (1):
        _, img = cap.read()
```

The next while loop checks if Image is being received. When it is True video is streaming and read() function is used to read the video frames and store in variable 'img'.

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Here we convert frame img i.e BGR to HSV (hue-saturation-value)

```
red_lower = np.array([136, 87, 111], np.uint8)
red_upper = np.array([180, 255, 255], np.uint8)
```

Defining range of red color with upper and lower intensity limits.

```
green_lower = np.array([50, 100, 100], np.uint8)
green_upper = np.array([70, 255, 255], np.uint8)
```

Defining range of green color with upper and lower intensity limits.

```
red = cv2.inRange(hsv, red_lower, red_upper)
green = cv2.inRange(hsv, green_lower, green_upper)
```

Finding the range of red and green color in the image.

```
kernal = np.ones((5, 5), "uint8")

red = cv2.dilate(red, kernal)
res = cv2.bitwise_and(img, img, mask=red)
```

Morphological transformation, dilation and compare the range of red color through bitwise operation.

```
green = cv2.dilate(green, kernal)
res2 = cv2.bitwise_and(img, img, mask=green)
```

Morphological transformation, dilation and compare the range of green color through bitwise operation.

```
# Tracking the Red Color
(_, contours, hierarchy) = cv2.findContours(red, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
for pic, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    if (area > 300):
        x, y, w, h = cv2.boundingRect(contour)
        img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
        cv2.putText(img, "RED Ball", (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255))

Xaxis = round((x - 308) / 8.8, 4)
Yaxis = -round((y - 455) / 8.8, 4)
print("R_X =", Xaxis, "cm", "R_Y =", Yaxis, "cm")
x = y = 0
```

Tracking the Red Color and draws a rectangle around it. Here converting camera pixels into centimeter and rounding off positive and negative values of detected red color.

```
# Tracking the Green Color
(_, contours, hierarchy) = cv2.findContours(green, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
for pic, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    if (area > 300):
        x2, y2, w, h = cv2.boundingRect(contour)
        img = cv2.rectangle(img, (x2, y2), (x2 + w, y2 + h), (0, 255, 0), 2)
        cv2.putText(img, "Green Ball", (x2, y2), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0))
X2axis = round((x2 - 308) / 8.8, 4)
Y2axis = -round((y2 - 455) / 8.8, 4)
print("G_X =", X2axis, "cm", "G_Y =", Y2axis, "cm")
x2 = y2 = 0
```

Tracking the green color and draws a rectangle around it. Here converting camera pixels into centimeter and rounding off positive and negative values of detected green color.

```
cv2.imshow("Color Tracking", img)

if cv2.waitKey(1) & 0xFF == ord('q'):
    cap.release()
    cv2.destroyAllWindows()
    break
```

cv2.imshow (img) show the normal video with red and green colors being tracked. Just execute the code, Make sure your webcam is connected.

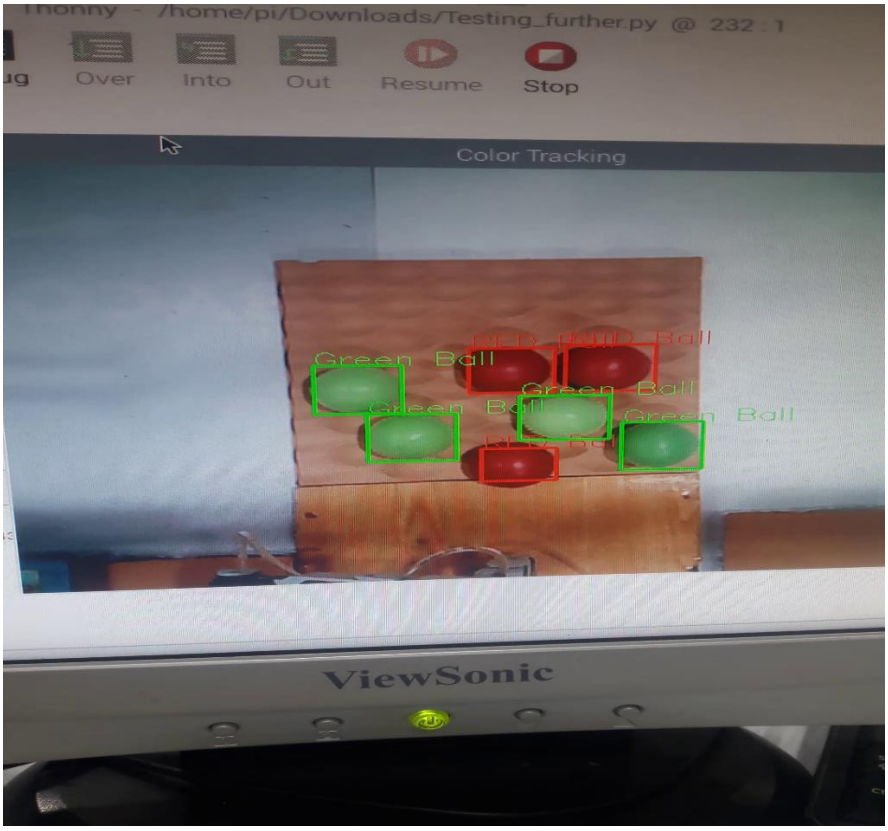

Fig. 5.4 Red and Green Colour Detection.

CHAPTER 6

PROJECT DEVELOPMENT

Objective of this chapter is to implement, develop and interface Hardware and Software to accomplish a desired prototype.

In this chapter we focus on development of our 4DoF Articulated Manipulator. It was quite rigorous process we gone through. Our project is combination of two categories i-e. Hardware and Software.

In Hardware we connect all the physical components to build a 4DoF Articulated Manipulator shown in Fig.6.1. Our Manipulator consist of;

- Smart Servos (Dynamixel Motors)
- Rigid Body (Links)
- Silicon Suction Gripper
- Nylon pipe
- Workspace
- Raspberry Pi
- Pneumatic Pump
- Pneumatic Control Circuit
- U2D Controller
- Valve
- Web Camera
- DC Power Supply (12, 5) Volts

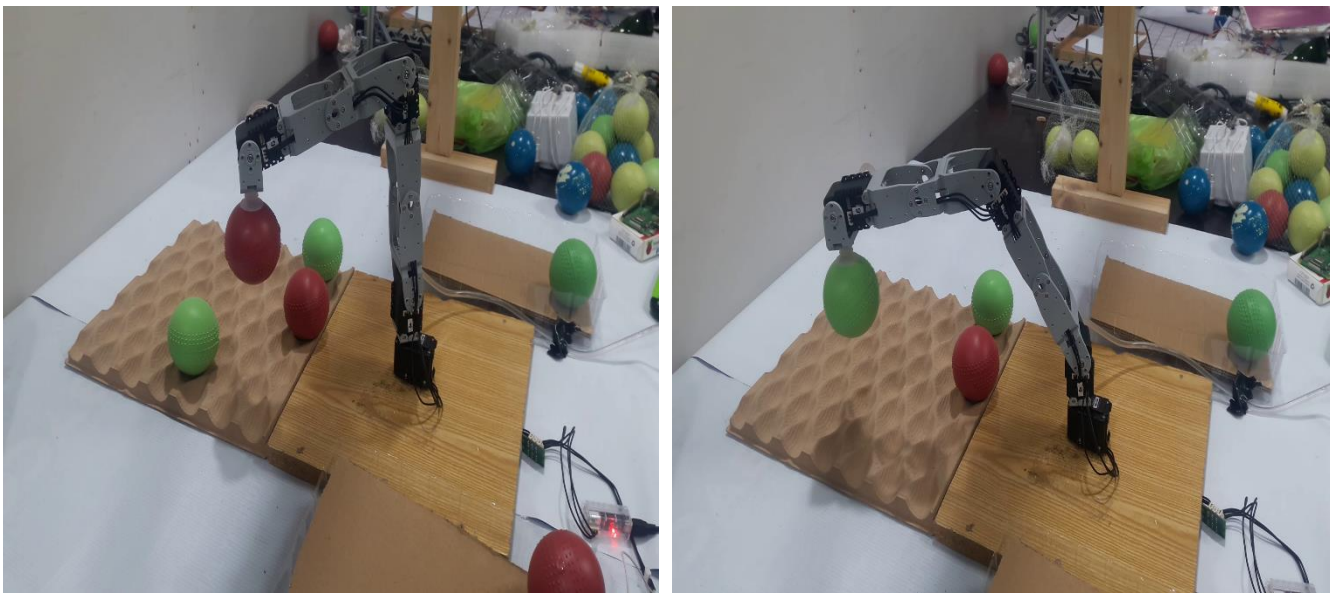

Fig. 6.1 4DOF Robotic Arm picking Red colour and Green colour balls respectively.

6.1 Methodology:

The methodology of the Robotic Arm is to take input from sensor and Process data and perform the action according to the input. In 4DoF Robotic Arm we take the data through webcam as a sensor, Data from Camera is converted to form frame of images and process these images through Raspberry Pi and after processing, set the angles of motor to reach at the object in the workspace and hovering the end-effector at the top of the object at workspace, then through GPIOs send signal to circuitry to pick the object with end effector and leave it on the desired location.

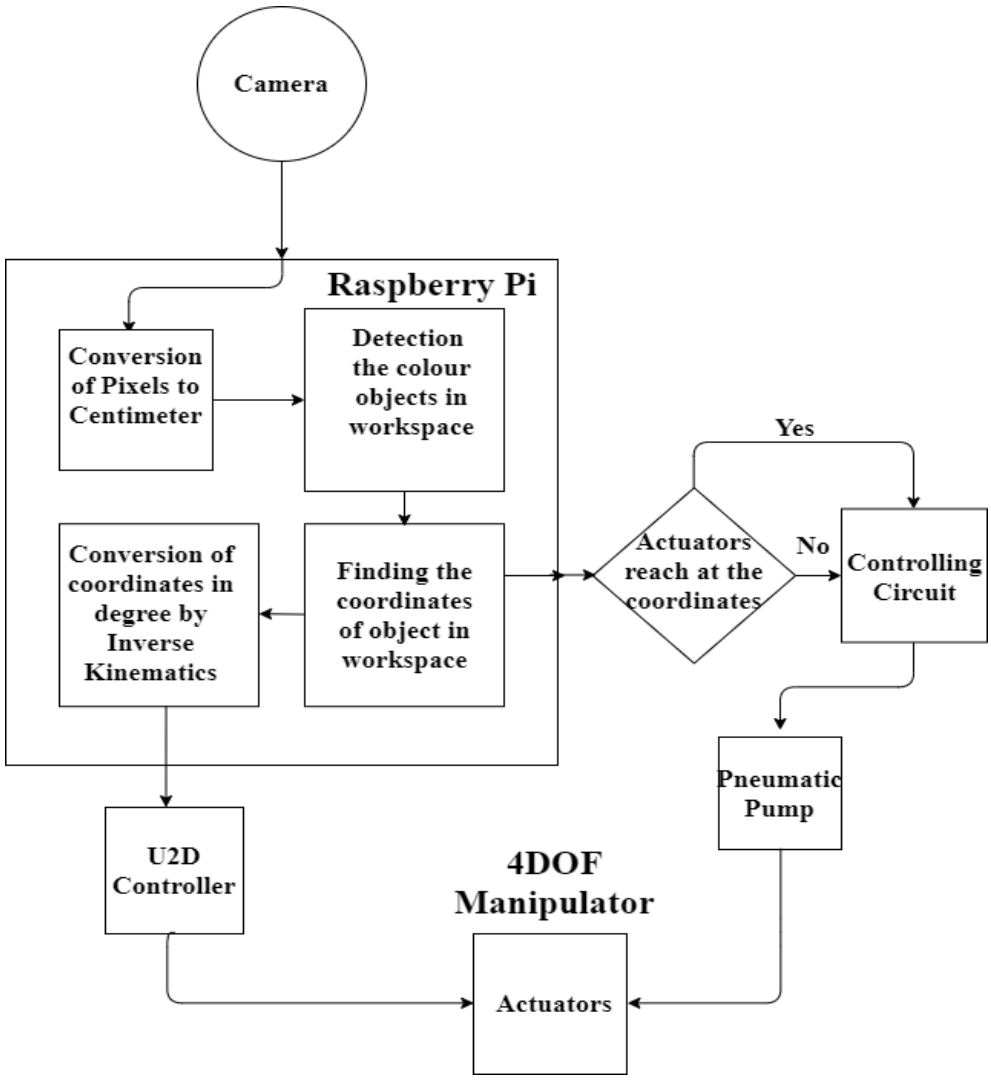

Fig. 6.2 Flow Chart of Hardware.

6.1.1 Camera:

Camera is mounted on the top of the workspace that detects the object present in workspace through its colour. Camera is plugged into raspberry pi through communication port its purpose here is to simply observe the objects or detect objects and continues it's observing the objects in the workspace.

6.1.2 Raspberry Pi:

When camera detects objects and gives input to the raspberry pi, it converts the picture displayed on the monitor from pixels to centimetres after detection the colour and find coordinates of the object in workspace. After finding coordinates it converts coordinates into degree through Inverse Kinematics method. These degrees send to the Actuators for reaching at the coordinate position through U2D controller.

6.1.3 Controlling Circuit:

Controlling circuit energizes by 5v DC supply and it contains components like male connectors that connects resistor and transistor that energizes relay. If the relay contact is Normally Open (NO) then pneumatic pump will operate else the contact is Normally Close (NC) then Valve terminal of 5 volts will energize.

6.1.4 Pneumatic Pump:

Through GPIO pins from raspberry pi the controlling circuit will be controlled, when GPIO pin is in active mode (Yes) it will send the signal to the control circuit to turn ON the pneumatic pump for picking the object and if GPIO pin is in deactivate mode (No) it will also send a signal to the control circuit to turn off the pneumatic pump or leave the object from suction cup attached to pneumatic pump.

6.1.5 U2D2 Controller:

U2D2 is a small size USB communication converter that enables to control and to operate the DYNAMIXEL with the PC. U2D controller takes the input from Raspberry pi and send that input signal to the Actuators.

6.1.6 Actuators:

Actuators, also known as drives, are mechanisms for getting robots to move. These actuators are controlled or set their rotary angle through data come from U2D controller and reaches at the position in workspace that Cartesian points converted into joint angles.
After getting command from raspberry pi Actuators/dynamixel motors controlled by U2D controller reach the coordinates calculated from Inverse method and controlling circuit operates. To get the centimetre from pixels, it is require to calibrate camera mounted on the top the workspace.
After assembling all components we operate Raspberry pi and write the program in python for this whole detection, recognition and performing action in PyCharm IDE, the code is given in Appendix-B.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The prepared mechanism has been successfully assembled and it can autonomously pick and place objects like ball and place them at different locations.

Several technological advances have always tried the minds of men. But one thing is for sure, technology has always been there to help us in our everyday situations. Just think of how you could get food to stay fresh without a fridge. In fact, technology has given us the power to support our rapidly growing needs in order to survive.

Just like with manufacturing industry, most of the companies today that have reach success have upgraded their machine in to something automated. Productions have increased since they have improved the machines that they use in producing their products. If you are using 30 years old machine and it does not provide the certain number of products required you will definitely lose a lot on your income, this is true especially on bottling and packaging department. If your workers and machines are too old to handle big orders you might torture your equipment and cost a lot the damage that it will incur if you force it to work continuously.

A simple robotic arm in this demonstrated a vision of how an automated system can ease your life. From industry to your home automation can help you build better system. Technology has always been a gift to our lives. It is just us use it well. If we can do things faster and better, then it must be a gift from the God.

7.2 Future Scope

Given more time a higher DOF manipulator could have been studied and realised. The dynamics of the manipulator could have been studied in further detail.
Moreover instead of using pneumatic arm gripper, hand is multi-tasking device capable of diverse functions for example grasping, manipulating, and pushing. Hand can be controlled by both motors and sensory perceptions according to the applications.
Obstacle avoidance can be performed by making arm more intelligent through programming.
Furthermore varieties and more flexibility to add or replace any part according to the requirements can be done to improve its use, increase field of usage and to make it more universal or flexible.

References

1. Mr. Manjunatha H. S, Ms. Megha T. M, Ms. Kanikashree A. P, Ms. Heena kauser." *Camera based color identification using robotic arm."* Project Reference no.: 39s_be_0844.

2. Daniel Scott Stuart," *Implementation of Robot Arm Networks and Experimental Analysis of Consensus-Based Collective Motion".* Utah State University, 2009

3. Ameer Mohammed Rajah, "*Creating Artificial Vision of Artificial Life*". Published on Feb 21, 2012,

4. John J. Craig, *"Introduction to Robotics*" Published on 2005 Pearson Education, Inc.

5. Matt Richardson and Shawn Wallance, 2012, getting started with raspberry pi, 2nd ed, United States of America: O'Reily Media, Inc.

6. John J. Craig, *"Introduction to Robotics Mechanics and Control Third Edition".*

7. Zhang Zhiyang, He Dongjian, Tang Jin Lei, Meng Lingshuai, *"Picking Robot Arm Trajectory Planning Method"*. College of Mechanical and Electronic Engineereing, Northwest A&F University, Yangling, Shaanxi, 712100, China.

8. Ravikumar Mourya, Amit Shelke, Sourabh Satpute, Shushant Kakade, Manjo Botre, *"Design and Implementation of Pick and Place Robotic Arm "*. Department of Mechanical Engineering, JSPM Narhe Technical Campus Pune.

9. Serdar Kucuk and Zafer Bingal, *"Robot Kinematics: Forward and Inverse Kinematics".*

10. https://hackernoon.com/top-8-python-libraries-for-machine-learning-and-artificial-intelligence-y08id3031.Accessed on (October 10,2019)
11. https://www.geeksforgeeks.org/python-introduction-matplotlib/. Accessed on (October 20,2019)
12. https://linuxhint.com/10_best_math_libraries_python/. Accessed on (October 28,2019)
13. https://stackabuse.com/the-python-math-library/. Accessed on (November 5,2019)
14. https://stackabuse.com/introduction-to-opencv-with-python/. Accessed on (November 15,2019)
15. https://docs.python.org/3/library/time.html. Accessed on (November 22,2019)
16. https://www.raspberrypi-spy.co.uk/2012/05/install-rpi-gpio-python-library/. Accessed on (December 2,2019)

17. http://www.robotis-shopjp.com/?act=shop_jp.goods_view&GS=3288&GC=GD0C0107. Accessed on (December 20,2019)
18. http://www.electronicsteacher.com/robotics/robotics-technology/actuators.php. Accessed on (December 22,2019)

Appendix – A

GLOSSARY

Digital Video Recorder (DVR)
A device that records audio and video input, typically from a television signal, on to a hard disk.

Ethernet
A System for connecting a number of computer system from a local area network, with protocols to control the passing of information and to avoid simultaneous transmission by two or more systems.

GPIO
General-purpose Input/output (GPIO) is a generic pin on an integrated circuit whose behaviour, including whether it is an input or output pin. Can be controlled by the user at run time. GPIO pins have no special purpose defined, and go unused by default.

GPU
A Graphics Processing Unit (GPU) is a single-chip processor primarily used to manage and boost the performance of video and graphics.

HDMI
High Definition Multimedia Interface (HDMI) is a proprietary Audio/video interface for transferring uncompressed video data and compressed or uncompressed digital audio data from an HDMI-compliant source device, such as a display controller, to a compatible computer monitor, video projector, digital television, or digital audio device. HDMI is a digital replacement for analog video standards.

Open Source Computer Vision (OpenCV)
OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel research centre in Nizhny Novgood (Russia), later supported by Willow Garage and now maintained by Itseez.

Python
A high-level general-purpose programming language.

Raspberry Pi
The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

Appendix - B

PROGRAM Codes

Command Window

Updating Raspberry Pi

```
sudo apt-get install rpi-updates
sudo rpi-update
sudo apt-get update
sudo apt-get upgrade
```

Enabling Camera

```
sudo raspi-config
enable camera
finish
```

PyCharm IDE

Importing Libraries

```
import time
import RPi.GPIO as GPIO
import cv2
import math
import numpy as np
```

Declairing Variables

```
channel = 12
x1 = 0
y1 = 0
x = 0
y = 0
i = 0
```

Capturing Video From Camera

```
cap= cv2.VideoCapture(0)
```

Function of Inverse Kinematics

```
def inverse_kinamtics(x, y, z, phi):
        distance = math.sqrt(x ** 2 + y ** 2 + (z - 5.1) ** 2)
        print("distance is ==", distance)
        if distance <= 34.2:
            theta1 = math.atan2(y, x)
        # print(theta1)
            theta1_degree = theta1 * 57.2957795
            print("theta1=", theta1_degree)

       ##########################Theta2################
            dami1 = (z - 8.1)
            dami2 = (x * math.cos(theta1) + y * math.sin(theta1))
            dami3 = (dami1 ** 2 + dami2 ** 2) / 34.2
```

```
            theta2 = math.atan2(dami3, -math.sqrt(dami1 ** 2 + dami2 ** 2
- dami3 ** 2)) - math.atan2(dami2, dami1)
            theta2_degree = theta2 * 57.2957795 + 7
            print("theta2=", theta2_degree)

        ########################Theta3################
            dami4 = 17.1
            dami5 = 17.1
            dami6 = ((math.cos(theta2) * (dami1 + dami2)) +
(math.sin(theta2) * (dami1 - dami2)) - (17.1))
            theta3 = math.atan2(dami6, math.sqrt(dami4 ** 2 + dami5 ** 2 -
dami6 ** 2)) - math.atan2(dami5, dami4)
        # print(theta3)
            theta3_degree = theta3 * 57.2957795
            print("theta3=", theta3_degree)

        ########################Theta4################
            theta4 = (phi - theta2 - theta3)
        # print(theta4)
            theta4_degree = theta4 * 57.2957795
            print("theta4", theta4_degree)
            return (theta1_degree, theta2_degree, theta3_degree,
theta4_degree)

        else:
            print("end factor is out of reach")
```

Function of Mapping Angles

```
def maprange(a, b, s):
    (a1, a2), (b1, b2) = a, b
    return b1 + ((s - a1) * (b2 - b1) / (a2 - a1))


def Map_fun(theta1_degree, theta2_degree, theta3_degree, theta4_degree):
    #########this rounding is for third quardraten##################
    if x < 0 and y < 0:
        angle1 = -round(theta1_degree, 2) + 60 + 90
    else:
        angle1 = round(theta1_degree, 2) + 60
    ############rounding and offset of motor hardware
    angle2 = round(theta2_degree, 2) + 60
    angle3 = round(theta3_degree, 2) + 90 + 60
    angle4 = round(theta4_degree, 2) + 60
    ################################################
    ###################### Mapping function###############
    ####this for to map the joint angle with motor angle########
    angle1_map = maprange((0, 300), (-150, 150), angle1)
    print(angle1_map)
    angle2_map = maprange((0, 300), (-150, 150), angle2)
    print(angle2_map)
    angle3_map = maprange((0, 300), (-150, 150), angle3)
    print(angle3_map)
    angle4_map = maprange((0, 300), (-150, 150), angle4)
    print(angle4_map)
```

```
    # print("%2g maps to %g" % (s, maprange((0, 300), (-150, 150), s)))
    # Connect to the serial port
    from pyax12.connection import Connection
    serial_connection = Connection(port="/dev/ttyUSB0", baudrate=1000000)
    ############ID of dynamixel motors###############
    dynamixel_id1 = 8
    dynamixel_id2 = 4
    dynamixel_id3 = 5
    dynamixel_id4 = 6
    # voltage = Connection.get_present_voltage(dynamixel_id=dynamixel_id1)
    #######Method to move motors to desired angle with desired
speed###########
    serial_connection.goto(dynamixel_id1, angle1_map, speed=80,
degrees=True)
    time.sleep(0.005)  # Wait 1 second
    serial_connection.goto(dynamixel_id2, angle2_map, speed=80,
degrees=True)
    time.sleep(0.005)  # Wait 1 second
    serial_connection.goto(dynamixel_id3, angle3_map, speed=80,
degrees=True)
    time.sleep(0.005)  # Wait 1 second
    serial_connection.goto(dynamixel_id4, angle4_map, speed=80,
degrees=True)
    # Close the serial connection
    serial_connection.close()
    # y = y + 0.36
    # time.sleep(2)
    # time.sleep(0.1)    # Wait 1 second
```

Function for Pneumatic Pump

```
def motor_on(pin):
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(channel, GPIO.OUT)
    GPIO.output(pin, GPIO.HIGH)

def motor_off(pin):
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(channel, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)
```

Defining the Intensity of Colors

```
while (1):
        _, img = cap.read()

        # converting frame(img i.e BGR) to HSV (hue-saturation-value)

        hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

        # definig the range of red color
        red_lower = np.array([136, 87, 111], np.uint8)
```

```
        red_upper = np.array([180, 255, 255], np.uint8)


        # defining the Range of green color
        green_lower = np.array([50, 100, 100], np.uint8)
        green_upper = np.array([70, 255, 255], np.uint8)

        # finding the range of red and green color in the image
        red = cv2.inRange(hsv, red_lower, red_upper)
        green = cv2.inRange(hsv, green_lower, green_upper)

        # Morphological transformation, Dilation
        kernal = np.ones((5, 5), "uint8")

        red = cv2.dilate(red, kernal)
        res = cv2.bitwise_and(img, img, mask=red)

        green = cv2.dilate(green, kernal)
        res2 = cv2.bitwise_and(img, img, mask=green)

        # Tracking the Red Color
        (_, contours, hierarchy) = cv2.findContours(red, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        for pic, contour in enumerate(contours):
            area = cv2.contourArea(contour)
            if (area > 300):
                x, y, w, h = cv2.boundingRect(contour)
                img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0,
255), 2)
                cv2.putText(img, "RED Ball", (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255))

        Xaxis = round((x - 308) / 8.8, 4)
        Yaxis = -round((y - 455) / 8.8, 4)
        print("R_X =", Xaxis, "cm", "R_Y =", Yaxis, "cm")
        x = y = 0


        # Tracking the Green Color
        (_, contours, hierarchy) = cv2.findContours(green, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        for pic, contour in enumerate(contours):
            area = cv2.contourArea(contour)
            if (area > 300):
                x1, y1, w, h = cv2.boundingRect(contour)
                img = cv2.rectangle(img, (x1, y1), (x1 + w, y1 + h), (0,
255, 0), 2)
                cv2.putText(img, "Green Ball", (x1, y1),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0))
        X1axis = round((x1 - 308) / 8.8, 4)
        Y1axis = -round((y1 - 455) / 8.8, 4)
        print("G_X =", X1axis, "cm", "G_Y =", Y1axis, "cm")
        x1 = y1 = 0
```

Calling the Function to Activate the Manipulator

```
        i = i + 1
         try:
             while (i > 5):
                 ####################For Red ########################
                 if (Xaxis > -35):
                     cv2.imshow("Color Tracking", img)
                     theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics( (Xaxis+2),( Yaxis-2), 15, 0)
                     Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                     time.sleep(2)

                     motor_on(channel)
                     time.sleep(1)

                     theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics((Xaxis+2),( Yaxis-2), 9, 0)
                     Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                     time.sleep(2)

                     theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics((Xaxis+2), ( Yaxis-2), 20, 0)
                     Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                     time.sleep(2)

                     theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics(-18, 0, 16, 0)
                     Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                     time.sleep(3)

                     motor_off(channel)
                     time.sleep(1)
                  elif(X2axis > -35):
                     cv2.imshow("Color Tracking", img)


                ##############for Green################################
                     theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics((X2axis+2),( Y2axis-2), 15, 0)
                     Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                     time.sleep(2)

                     motor_on(channel)
                     time.sleep(1)

                     theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics((X2axis+2),( Y2axis-2), 9, 0)
                     Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
```

```
                    time.sleep(2)

                    theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics((X2axis+2), (Y2axis-2), 20, 0)
                    Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                    time.sleep(2)

                    theta1_degree, theta2_degree, theta3_degree,
theta4_degree = inverse_kinamtics(18, 0, 16, 0)
                    Map_fun(theta1_degree, theta2_degree, theta3_degree,
theta4_degree)
                    time.sleep(3)

                    motor_off(channel)
                    time.sleep(1)
                else:
                    print("there is no ball")

                i = 0

                cv2.imshow("Color Tracking", img)

                if cv2.waitKey(1) & 0xFF == ord('q'):
                    cap.release()
                    cv2.destroyAllWindows()
                    break
        except TypeError:
            print("error")
```

Appendix - C

MATHEMATICAL MODELLING

Solving for θ_2

Again applying pieper's technique to get θ_2, it states that multiply ${}^1_2T^{-1}$ frame both sides of equation **(4.5).**

$${}^1_2T^{-1} * {}^0_1T^{-1} * {}^0_4T = {}^1_2T^{-1} * {}^0_1T^{-1} * ({}^0_1T * {}^1_2T * {}^2_3T * {}^3_4T) \qquad \textbf{(4.5)}$$

From the forward kinematics of the manipulator we know ${}^1_2T$ as defined in **(4.2)** therefore we can calculate its inverse as;

$${}^1_2T^{-1} = \begin{bmatrix} C\theta_2 & S\theta_2 & 0 & -L_2 \\ -S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where ${}^1_2T^{-1}$ frame is calculated in Matlab using command inv (${}^1_2T$). And it can be calculated manually as well.

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \qquad \textbf{(C.1)}$$

Equation **(C.1)** states that elements of $\mathbf{b_{mn}}$ matrix equal to elements of $\mathbf{c_{mn}}$ Matrix.

Where b_{mn} as:

$b_{11} = S_2(C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)) - C_2S_1(C_4(S_1S_2S_3 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2)) - C_1C_2(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2))$

$b_{12} = 0$

$b_{13} = C_2S_1(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1)) - S_2(C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + C_3S_2)) + C_1C_2(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3))$

$b_{14} = zS_2 - d_1S_2 - l_2 + yC_2S_1 + xC_1C_2$

$b_{21} = C_2(C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)) + C_1S_2(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2)) + S_1S_2(C_4(S_1S_2S_2 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2))$

$b_{22} = 0$

$b_{23} = -C_2(C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + C_3S_2)) - C_1S_2(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3)) - S_1S_2(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1))$

$b_{24} = zC_2 - d_1C_2 - xC_1S_2 - yS_1S_2$

$b_{31} = C_1(C_4(S_1S_2S_3 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2)) - S_1(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2))$

$b_{32} = C^2{}_1 + S^2{}_1$

$b_{33} = S_1(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3)) - C_1(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1))$

$b_{34} = xS_1 - yC_1$

$b_{41} == 0$

$b_{42} = 0$

$b_{43} = 0$

$b_{44} = 1$

Where c_{mn} as:

$c_{11} = C_3C_4 - S_3S_4$

$c_{12} = 0$

$c_{13} = C_\theta S_4 + C_4S_3$

$c_{14} = l_3C_3$

$c_{21} = C_3S_4 + C_4S_3$

$c_{22} = 0$

$c_{23} = S_3S_4 - C_3C_4$

$c_{24} = l_3S_3$

$c_{31} = 0$

$c_{32} = 1$

$c_{33} = 0$

$c_{34} = 0$

$c_{41} = 0$

$c_{42} = 0$

$c_{43} = 0$

$c_{44} = 1$

By equating equations **b_{14}** and **b_{24}** with **c_{14}** and **c_{24}**, to get **θ_2**;.

$b_{14} = zS_2 - 8.1S_2 - 17.1 + yC_2S_1 + xC_1C_2$

$b_{24} = zC_2 - 8.1C_2 - yS_1S_2 - xC_1S_2$

$c_{14} = 17.1C_3$

$c_{24} = 17.1S_3$

By squaring and adding equations **b_{14}** and **b_{24}** with **c_{14}** and **c_{24}**,.

$$(S_2(z-8.1) + C_2(xC_1 + yS_1) - 17.1)^2 + (C_2(z-8.1) - S_2(xC_1 + yS_1))^2 = (17.1C_3)^2 + (17.1S_3)^2$$

Let

a = $z-8.1$, **b=** xC_1+yS_1

$$(aS_2+bC_2-17.1)^2+(aC_2-bS_2)^2=(17.1)^2$$

$$a^2S^2{}_2+b^2C^2{}_2+(17.1)^2+2abS_2C_2-34.2bC_2-34.2aS_2+a^2C^2{}_2+b^2S^2{}_2-2abC_2S_2=(17.1)^2$$

$$a^2+b^2-34.2(aS_2+bC_2)=0$$

$$aS_2+bC_2=\frac{a^2+b^2}{34.2} \qquad \textbf{(A)}$$

Applying this property:

$$aS\theta_2+bC\theta_2=c$$

Then,

$$\theta=a\tan 2\left(\frac{c}{\pm\sqrt{a^2+b^2-c^2}}\right)-a\tan 2\left(\frac{b}{a}\right) \qquad \textbf{(4.16)}$$

Putting **(A)** on **(4.16)** to get $\boldsymbol{\theta_2}$

$$\theta_2=a\tan 2\left(\frac{\frac{a^2+b^2}{34.2}}{\pm\sqrt{a^2+b^2-\left(\frac{a^2+b^2}{34.2}\right)^2}}\right)-a\tan 2\left(\frac{b}{a}\right)$$

Solving for $\boldsymbol{\theta_3}$

Again applying pieper's technique to get $\boldsymbol{\theta_3}$, it states that multiply ${}^1_2T^{-1}$ frame both sides of equation **(4.5).**

$${}^1_2T^{-1}*{}^0_1T^{-1}*{}^0_4T={}^1_2T^{-1}*{}^0_1T^{-1}*({}^0_1T*{}^1_2T*{}^2_3T*{}^3_4T)$$

From the forward kinematics of the manipulator we know ${}^1_2T$ as defined in **(4.2)** therefore we can calculate its inverse as;

$${}^1_2T^{-1}=\begin{bmatrix} C\theta_2 & S\theta_2 & 0 & -L_2 \\ -S\theta_2 & C\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where ${}^1_2T^{-1}$ frame is calculated in Matlab using command inv (${}^1_2T$). And it can be calculated manually as well. Recall equation **(C.1)**

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \qquad \textbf{(C.1)}$$

Equation **(C.1)** states that elements of $\mathbf{b_{mn}}$ matrix equal to elements of $\mathbf{c_{mn}}$ Matrix.

where b_{mn} as

$b_{11} = S_2(C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)) - C_2S_1(C_4(S_1S_2S_3 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2)) - C_1C_2(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2))$

$b_{12} = 0$

$b_{13} = C_2S_1(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1)) - S_2(C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + C_3S_2)) + C_1C_2(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3))$

$b_{14} = zS_2 - d_1S_2 - l_2 + yC_2S_1 + xC_1C_2$

$b_{21} = C_2(C_4(C_2S_3 + C_3S_2) + S_4(C_2C_3 - S_2S_3)) + C_1S_2(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2)) + S_1S_2(C_4(S_1S_2S_2 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2))$

$b_{22} = 0$

$b_{23} = -C_2(C_4(C_2C_3 - S_2S_3) - S_4(C_2S_3 + C_3S_2)) - C_1S_2(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3)) - S_1S_2(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1))$

$b_{24} = zC_2 - d_1C_2 - xC_1S_2 - yS_1S_2$

$b_{31} = C_1(C_4(S_1S_2S_3 - C_2C_3S_1) + S_4(C_2S_1S_3 + C_3S_1S_2)) - S_1(C_4(C_1S_2S_3 - C_1C_2C_3) + S_4(C_1C_2S_3 + C_1C_3S_2))$

$b_{32} = C^2{}_1 + S^2{}_1$

$b_{33} = S_1(C_4(C_1C_2S_3 + C_1C_3S_2) - S_4(C_1S_2S_3 - C_1C_2C_3)) - C_1(C_4(C_2S_1S_3 + C_3S_1S_2) - S_4(S_1S_2S_3 - C_2C_3S_1))$

$b_{34} = xS_1 - yC_1$

$b_{41} == 0$

$b_{42} = 0$

$b_{43} = 0$

$b_{44} = 1$

Where c_{mn} as:

$c_{11} = C_3C_4 - S_3S_4$

$c_{12} = 0$

$c_{13} = C_\theta S_4 + C_4S_3$

$c_{14} = l_3C_3$

$c_{21} = C_3S_4 + C_4S_3$

$c_{22} = 0$

$c_{23} = \quad S_3 S_4 - C_3 C_4$

$c_{24} = \quad l_3 S_3$

$c_{31} = \quad 0$

$c_{32} = \quad 1$

$c_{33} = \quad 0$

$c_{34} = \quad 0$

$c_{41} = \quad 0$

$c_{42} = \quad 0$

$c_{43} = \quad 0$

$c_{44} = \quad 1$

By equating equations $\mathbf{b_{14}}$ and $\mathbf{b_{24}}$ with $\mathbf{c_{14}}$ and $\mathbf{c_{24}}$, to get $\boldsymbol{\theta_3}$;.

$b_{14} = \quad zS_2 - 8.1S_2 - 17.1 + yC_2S_1 + xC_1C_2$

$b_{24} = \quad zC_2 - 8.1C_2 - yS_1S_2 - xC_1S_2$

$c_{14} = \quad 17.1C_3$

$c_{24} = \quad 17.1S_3$

By adding equations $\mathbf{b_{14}}$ and $\mathbf{b_{24}}$ with $\mathbf{c_{14}}$ and $\mathbf{c_{24}}$, to get $\boldsymbol{\theta_3}$

$$S_2(z-8.1) + C_2(xC_1 + yS_1) - 17.1 + C_2(z-8.1) - S_2(xC_1 + yS_1) = 17.1C_3 + 17.1S_3$$

Let

a = $z - 8.1$, **b=** $xC_1 + yS_1$

$$aS_2 + bC_2 - 17.1 + aC_2 - bS_2 = 17.1C_3 + 17.1S_3$$

$$S_2(a-b) + C_2(a+b) - 17.1 = 17.1C_3 + 17.1S_3$$

Let

c = $S_2(a-b) + C_2(a+b) - 17.1$

By Applying Property:

$$aS\theta_3 + bC\theta_3 = c \qquad \textbf{(A)}$$

Then,

$$\theta = a\tan 2\left(\frac{c}{\pm\sqrt{a^2+b^2-c^2}}\right) - a\tan 2\left(\frac{b}{a}\right) \qquad \textbf{(4.16)}$$

Putting **(A)** on **(4.16)** to get $\boldsymbol{\theta_3}$

$$\theta_3 = a\tan 2\left(\frac{S_2(a-b)+C_2(a+b)-17.1}{\pm\sqrt{17.1^2+17.1^2-\left(S_2(a-b)+C_2(a+b)-17.1\right)^2}}\right) - a\tan 2\left(\frac{17.1}{17.1}\right)$$

Solving for $\boldsymbol{\theta_4}$

$\boldsymbol{\theta_4} = \varphi - \theta_2 - \theta_3$