2 ROTATION MANIPULATOR (8051)

by

VISHWATH KUMAR B S	18BEC1289
YOGEESHWAR S	18BEC1343

A project report submitted to

Dr. M. JAGANNATH

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

ECE3003 – MICROCONTROLLER AND ITS APPLICATIONS

in

B.Tech. ELECTRONICS AND COMMUNICATION ENGINEERING

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

JUNE 2020

BONAFIDE CERTIFICATE

Certified that this project report entitled **“2 ROTATION MANIPULATOR”** is a bonafide work of **VISHWATH KUMAR B S - 18BEC1289, YOGEESHWAR S – 18BEC1343** who carried out the Project work under my supervision and guidance for **ECE3003 – MICROCONTROLLERS AND ITS APPLICATIONS.**

Dr. M. JAGANNATH

Associate Professor Senior

School of Electronics Engineering (SENSE),

VELLORE INSTITUTE OF TECHNOLOGY

Chennai – 600 127.

ABSTRACT

Manipulators with free joints are typically actuated mechanisms. In this project, we analyse the 2 link manipulator’s end effect and implement it with the help of 8051, keil uvision, proteus and scilab and find the deviation between the calculated and the actual values of the end effect.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. M. JAGANNATH,** Associate Professor Senior, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. A. Sivasubramanian,** Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Head of the Department **Dr. P. Vetrivelan** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

VishwathKumar. [illegible]

VISHWATH KUMAR B S **YOGEESHWAR S**

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVES

- Detect the input from the keypad (input points(X,Y))
- Parallelly display the given point on the LCD panel.
- To actuate the servos according to the given input points.
- To observe the end effect of the 2R manipulator and hence calculate the deviation from the calculated value.

1.2 BENEFITS

Our project proves that even 8051 being low on specifications, can be used as a powerful tool for plotting and graphing purposes. This automated actuation of the arms helps us swiftly plotting the points in a 2-D space using just 2 links and 2 rotations. Further improvement can prove much benefits in the area of robotics.

1.3 FEATURES

- MG995 High Torque Servo used for better performance and better accuracy.
- Keypad and LCD used for easy access and effective user interface.
- Microcontroller used is 8051.
- The efficient coding in the software tool called as Keil uVision5.

CHAPTER 2

2 ROTATION MANIPULATOR – DESIGN

2.1.1 FLOWCHART

The work flow of the 2R manipulator using 8051 in assembly language is given in **Figure 1**.

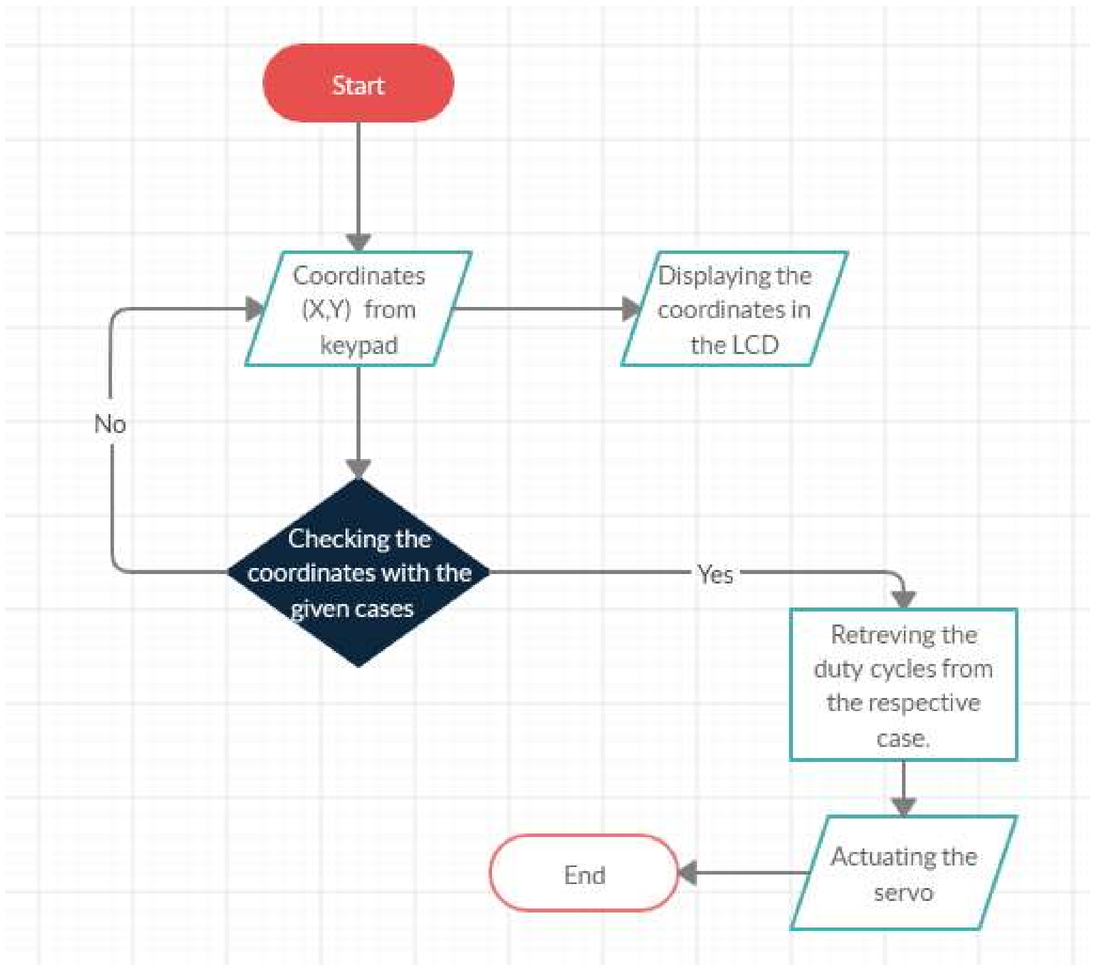

Start
Coordinates (X,Y) from keypad
Displaying the coordinates in the LCD
No
Checking the coordinates with the given cases
Yes
Retreving the duty cycles from the respective case.
Actuating the servo
End

Figure 1. FLOWCHART OF 2R MANIPULATOR

2.1.2 BLOCK DIAGRAM

The detailed working of the 2R manipulator using 8051 in assembly language is given in **Figure 2**:

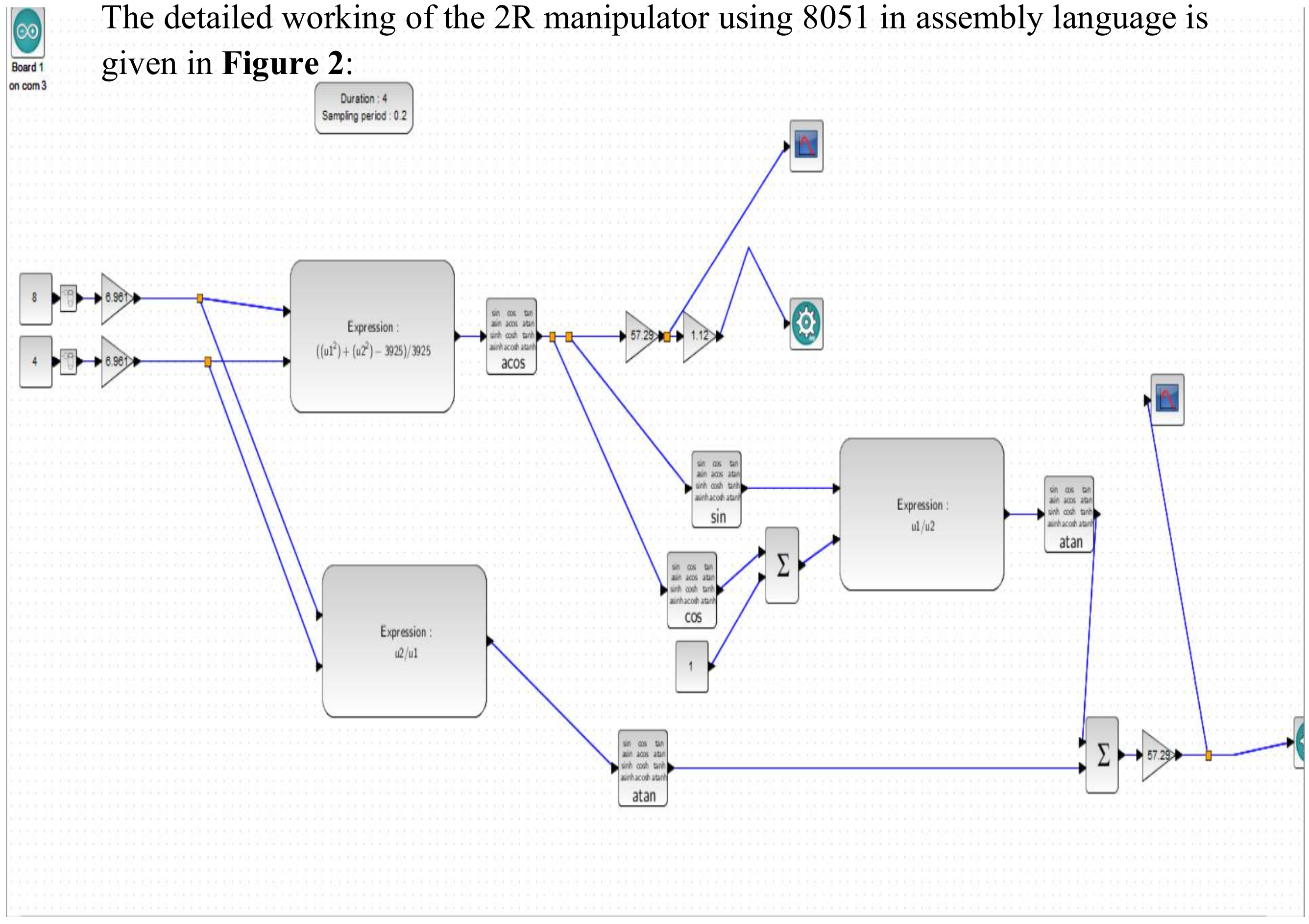


Figure 2. Block diagram for 2R manipulator in SciLab

2.2 HARDWARE SPECIFICATIONS

The prototype consists of :

- The power supply,
- Microcontroller 8051
- 8051 Programmer
- Actuating Rods (2nos)
- MG995 15kg-cm high torque servo (2 nos)
- 4x4 Matrix keypad
- 16x2 LCD display panel
- Push buttons
- Capacitors, Resistors and Jumper wires

2.2.1 8051 MICROCONTROLLER

- The AT89C51 [Refer **Figure 3**] is a low-power, high-performance CMOS 8-bit microcomputer with 4 Kbytes of Flash Programmable and Erasable Read Only Memory (EEPROM). The device is manufactured using Atmel's high density non-volatile memory technology and is compatible with the industry standard MCS-51 instruction set and pin-out.[12]

Figure 3. Atmel AT89c51 chip [8]

- The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.[12]

- The AT89C51 provides the following standard features: 4Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture [Refer **Figure 4**], a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 is designed with static memory.[12]

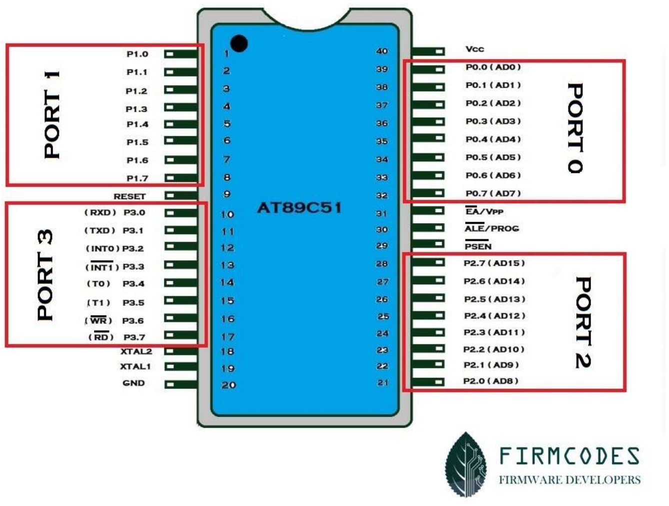


Figure 4. 8051 pin diagram [9].

2.2.2 MG995 HIGH TORQUE SERVO

MG995 is a **servo motor** that is popular for its performance and low price. The motor is used in many applications mainly being robotics and drones.[5]

Pin Configuration

MG995 has three terminals as mentioned in pin diagram [Refer **Figure 5**] and the function of each pin is given in the **Table 1**.

Pin	**Name**	**Function**
1	Signal pin (Orange pin)	The PWM signal which states the axis position is given through this pin.
2	VCC (Red pin)	Positive power supply for servo motor is given to this pin.
3	Ground(Brown pin)	This pin is connected to ground of circuit or power supply.

Table 1. Pin configuration of MG995 servo.[5]

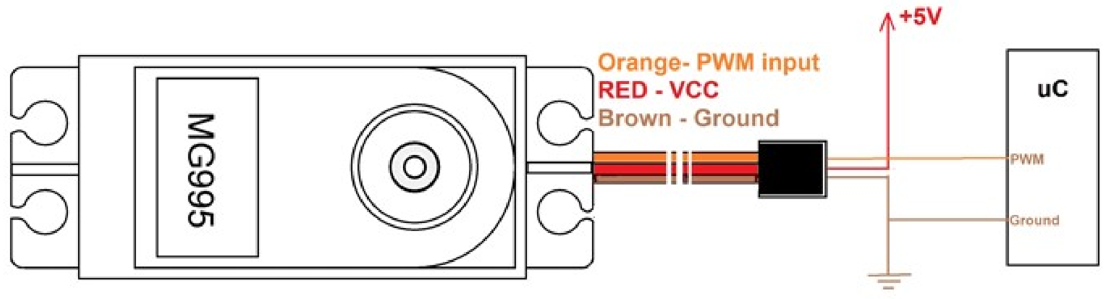


Figure 5. Pin diagram of MG995 servo.[5]

MG995 FEATURES AND ELECTRICAL CHARACTERISTICS [5]

- Constant torque throughout the servo travel range
- Operating voltage range: 4.8 V to 7.2 V
- Stall torque: 9.4kg/cm (4.8v); 11kg/cm (6v)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Rotational degree: 180º
- Dead band width: 5 µs
- Operating temperature range: 0ºC to +55ºC
- Current draw at idle: 10mA
- No load operating current draw: 170mA
- Current at maximum load: 1200mA

MG995 Servo Motor Overview

Since MG995 is a servo motor providing precise rotation over 180º range its applications are many and in them a few are stated below

- The servo is suited for designing robotic arm in which wear and tear of motor is high. Being metal geared, the servo has long life and can be installed on system like robotic arm were motor work is huge.
- The servo is also suited to be used in drones and toy planes. Having a satisfying torque which is enough to overcome air resistance and control wings of plane, the servo is preferred in toy planes and drones which need precision control no matter the condition.[5]

CONTROLLING MG995 WITH PWM

- **Frequency of PWM:** MG995 takes in PWM signal of frequency 50Hz and any higher and lower frequency PWM will lead to error. As shown in figure the every single cycle of PWM needs to be 20ms width for 50Hz frequency.
- **Duty cycle of PWM:** The duty cycle of PWM (or ratio of ON time to total cycle time) determines the position of servo axis. If we provide a PWM signal of 0.5ms ON time over 20mS complete cycle, the servo axis will move to 0º.

And if we provide a PWM signal of 1.5ms ON time over 20mS complete cycle, the servo axis will move to 90º. At last if we provide a PWM signal of 2.5ms ON time over 20mS complete cycle, the servo axis will move to 180º. Refer **Figure 6** for the pictorial representation.

Based on these standard values we can also calculate any other degree of rotation. After calculation we just have to adjust the duty cycle of the PWM for the servo to read the signal and change to that stated position.[5]

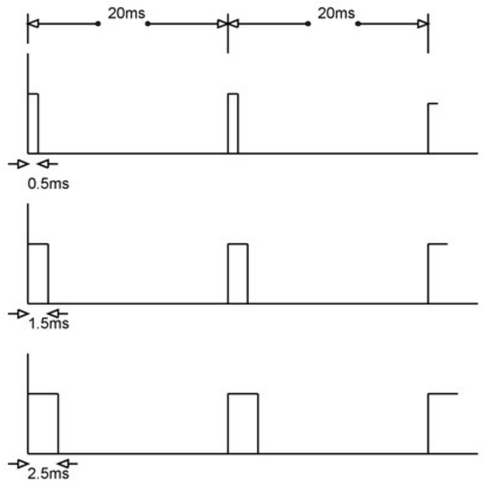


Figure 6. Duty cycle of the PWM for the servo.[5]

In this way we can control the servo position using PWM signal of any microcontroller or processor. In arduino and other development boards we will have libraries readily available to make it even easier to control the servo. If we use those libraries we can directly state the servo position instead of adjusting the PWM duty cycle every single time.[5]

2.2.2 4X4 MATRIX KEYPAD

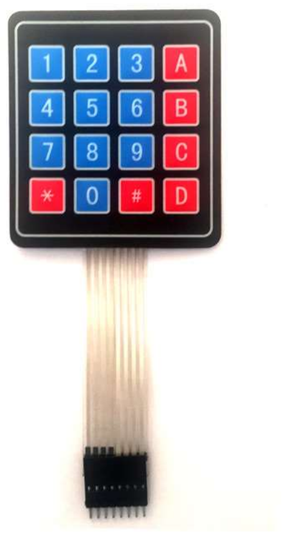


Figure 7. 4x4 Keypad.[10]

4X4 KEYPAD Pin Configuration:

4X4 KEYPAD MODULES are available in different sizes and shapes. They all have same pin configuration. 16 buttons is arranged in 4X4 KEYPAD in matrix formation as you can see in the **Figure 7**.[10]

Pin Number	Description
ROWS	
1	PIN1 is taken out from 1st ROW
2	PIN2 is taken out from 2nd ROW
3	PIN3 is taken out from 3rd ROW
4	PIN4 is taken out from 4th ROW

COLUMN	
5	PIN5 is taken out from 1st COLUMN
6	PIN6 is taken out from 2nd COLUMN
7	PIN7 is taken out from 3rd COLUMN
8	PIN8 is taken out from 4th COLUMN

Table 2. Pin configuration of the 4x4 Keypad.[10]

As given in **Table 2** a **4X4 KEYPAD** will have **EIGHT TERMINALS**. In them four are **ROWS of MATRIX** and four are **COLUMNS of MATRIX**. These 8 PINS are driven out from 16 buttons present in the MODULE. Those 16 alphanumeric digits on the MODULE surface are the 16 buttons arranged in MATRIX formation.[10]

The **internal structure of 4X4 KEYPAD MODULE** is shown in **Figure 8**.

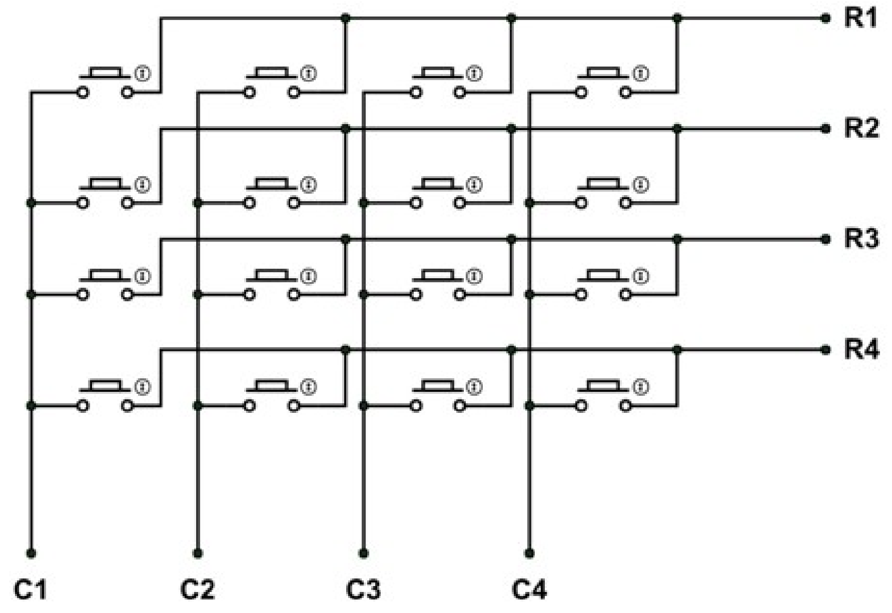


Figure 8. Internal structure of the 4x4 Keypad.[10]

2.2.3 16x2 LCD DISPLAY PANEL

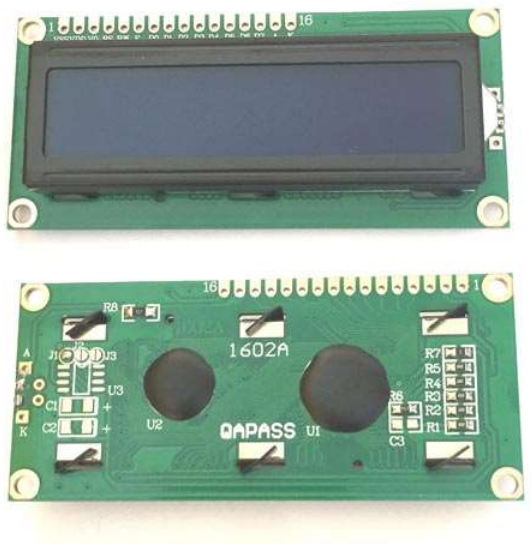


Figure 9. 16x2 LCD display.[11]

LCD modules [Refer **Figure 9**] are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. Most of us would have come across these displays in our day to day life, either at PCO's or calculators. The appearance and the pinouts have already been visualized above now let us get a bit technical. The pin configuration of the 16x2 LCD panel is shown in **Table 3**.[11]

Pin Configuration:

Pin No:	**Pin Name:**	**Description**
1	Vss (Ground)	Ground pin connected to system ground
2	Vdd (+5 Volt)	Powers the LCD with +5V (4.7V – 5.3V)
3	VE (Contrast V)	Decides the contrast level of display. Grounded to get maximum contrast.
4	Register Select	Connected to Microcontroller to shift between command/data register
5	Read/Write	Used to read or write data. Normally grounded to write data to LCD
6	Enable	Connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement

7	Data Pin 0	Data pins 0 to 7 forms a 8-bit data line. They can be connected to Microcontroller to send 8-bit data. These LCD's can also operate on 4-bit mode in such case Data pin 4,5,6 and 7 will be left free.
8	Data Pin 1	
9	Data Pin 2	
10	Data Pin 3	
11	Data Pin 4	
12	Data Pin 5	
13	Data Pin 6	
14	Data Pin 7	
15	LED Positive	Backlight LED pin positive terminal
16	LED Negative	Backlight LED pin negative terminal

Table 3. Pin configuration of 16x2 LCD display.[11]

2.3 SOFTWARE SPECIFICATIONS

- Keil uVision5
- Proteus Professional
- SciLab

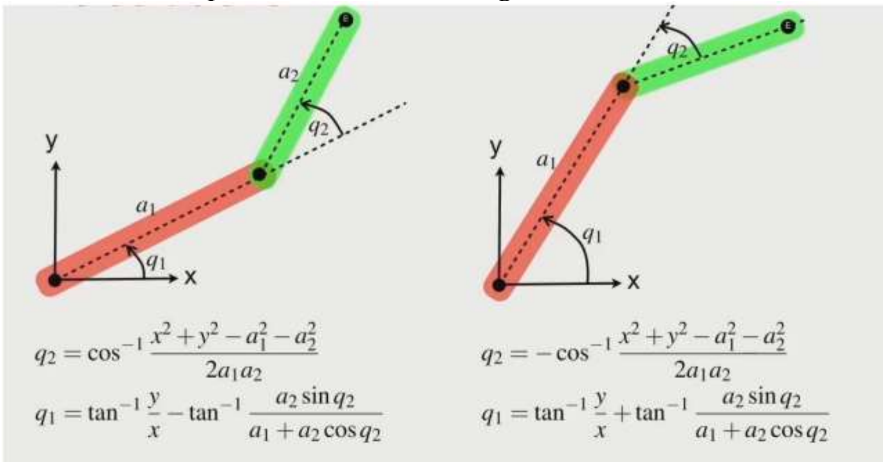
CHAPTER 3
SYSTEM IMPLEMENTATION AND ANALYSIS

This section describes system implementation and results with inferences.

3.1 APPROACH

3.1.1 INVERSE KINEMATICS

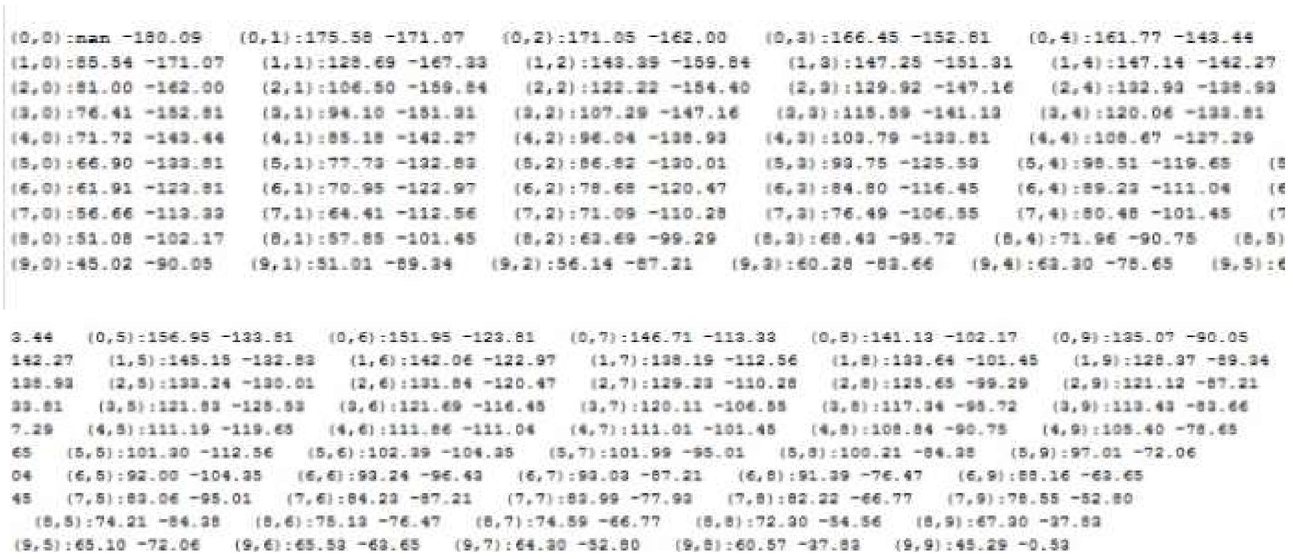
The inverse kinematic equations are shown in **Figure 10.**



$$q_2 = \cos^{-1}\frac{x^2+y^2-a_1^2-a_2^2}{2a_1a_2} \qquad q_2 = -\cos^{-1}\frac{x^2+y^2-a_1^2-a_2^2}{2a_1a_2}$$

$$q_1 = \tan^{-1}\frac{y}{x} - \tan^{-1}\frac{a_2\sin q_2}{a_1+a_2\cos q_2} \qquad q_1 = \tan^{-1}\frac{y}{x} + \tan^{-1}\frac{a_2\sin q_2}{a_1+a_2\cos q_2}$$

Figure 10. Inverse kinematic equation for 2 link.[1]



```
(0,0):nan -180.09   (0,1):175.58 -171.07   (0,2):171.05 -162.00   (0,3):166.45 -152.81   (0,4):161.77 -143.44
(1,0):85.54 -171.07   (1,1):128.69 -167.33   (1,2):143.39 -159.84   (1,3):147.25 -151.31   (1,4):147.14 -142.27
(2,0):81.00 -162.00   (2,1):106.50 -159.84   (2,2):122.22 -154.40   (2,3):129.92 -147.16   (2,4):132.93 -138.93
(3,0):76.41 -152.81   (3,1):94.10 -151.31   (3,2):107.29 -147.16   (3,3):115.59 -141.13   (3,4):120.06 -133.81
(4,0):71.72 -143.44   (4,1):85.18 -142.27   (4,2):96.04 -138.93   (4,3):103.79 -133.81   (4,4):108.67 -127.29
(5,0):66.90 -133.81   (5,1):77.73 -132.83   (5,2):86.82 -130.01   (5,3):93.75 -125.53   (5,4):98.51 -119.65   (5
(6,0):61.91 -123.81   (6,1):70.95 -122.97   (6,2):78.68 -120.47   (6,3):84.80 -116.45   (6,4):89.23 -111.04   (6
(7,0):56.66 -113.33   (7,1):64.41 -112.56   (7,2):71.09 -110.28   (7,3):76.49 -106.55   (7,4):80.48 -101.45   (7
(8,0):51.08 -102.17   (8,1):57.85 -101.45   (8,2):63.69 -99.29   (8,3):68.43 -95.72   (8,4):71.96 -90.75   (8,5)
(9,0):45.02 -90.05   (9,1):51.01 -89.34   (9,2):56.14 -87.21   (9,3):60.28 -83.66   (9,4):63.30 -78.65   (9,5):6
```

```
3.44   (0,5):156.95 -133.81   (0,6):151.95 -123.81   (0,7):146.71 -113.33   (0,8):141.13 -102.17   (0,9):135.07 -90.05
142.27   (1,5):145.15 -132.83   (1,6):142.06 -122.97   (1,7):138.19 -112.56   (1,8):133.64 -101.45   (1,9):128.37 -89.34
138.93   (2,5):133.24 -130.01   (2,6):131.84 -120.47   (2,7):129.23 -110.28   (2,8):125.65 -99.29   (2,9):121.12 -87.21
33.81   (3,5):121.83 -125.53   (3,6):121.69 -116.45   (3,7):120.11 -106.55   (3,8):117.34 -95.72   (3,9):113.43 -83.66
7.29   (4,5):111.19 -119.65   (4,6):111.86 -111.04   (4,7):111.01 -101.45   (4,8):108.84 -90.75   (4,9):105.40 -78.65
65   (5,5):101.30 -112.56   (5,6):102.39 -104.35   (5,7):101.99 -95.01   (5,8):100.21 -84.38   (5,9):97.01 -72.06
04   (6,5):92.00 -104.35   (6,6):93.24 -96.43   (6,7):93.03 -87.21   (6,8):91.39 -76.47   (6,9):88.16 -63.65
45   (7,5):83.06 -95.01   (7,6):84.23 -87.21   (7,7):83.99 -77.93   (7,8):82.22 -66.77   (7,9):78.55 -52.80
  (8,5):74.21 -84.38   (8,6):75.13 -76.47   (8,7):74.59 -66.77   (8,8):72.30 -54.56   (8,9):67.30 -37.83
(9,5):65.10 -72.06   (9,6):65.53 -63.65   (9,7):64.30 -52.80   (9,8):60.57 -37.83   (9,9):45.29 -0.53
```

Figure 11. Lookup tables for coordinates vs angles.

The respective angles for the given coordinates in the input are looked up from the **Figure 11.**

3.1.2. DUTY CYCLE CALCULATION

Duty cycle for 1st servo =[(q1/18)+2.5] [Refer **Figure 12** for further details]

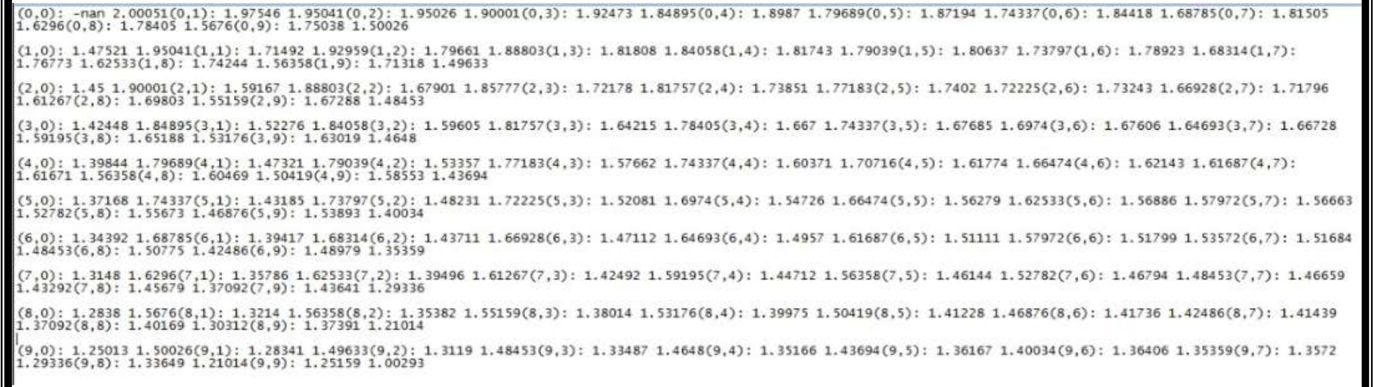
Duty cycle for 2nd servo =[(q1+q2)/18+2.5] [Refer **Figure 12** for further details]

Delay for 1st servo = (Duty cycle for 1st servo*20)/100 ms

Delay for 2nd servo = (Duty cycle for 2nd servo*20)/100 ms

Range of the delay= 2.5ms-12.5ms (0-180 degrees)

Refresh rate – 20ms



```
(0,0): -nan 2.00051(0,1): 1.97546 1.95041(0,2): 1.95026 1.90001(0,3): 1.92473 1.84895(0,4): 1.8987 1.79689(0,5): 1.87194 1.74337(0,6): 1.84418 1.68785(0,7): 1.81505
1.6296(0,8): 1.78405 1.5676(0,9): 1.75038 1.50026

(1,0): 1.47521 1.95041(1,1): 1.71492 1.92959(1,2): 1.79661 1.88803(1,3): 1.81808 1.84058(1,4): 1.81743 1.79039(1,5): 1.80637 1.73797(1,6): 1.78923 1.68314(1,7):
1.76773 1.62533(1,8): 1.74244 1.56358(1,9): 1.71318 1.49633

(2,0): 1.45 1.90001(2,1): 1.59167 1.88803(2,2): 1.67901 1.85777(2,3): 1.72178 1.81757(2,4): 1.73851 1.77183(2,5): 1.7402 1.72225(2,6): 1.73243 1.66928(2,7): 1.71796
1.61267(2,8): 1.69803 1.55159(2,9): 1.67288 1.48453

(3,0): 1.42448 1.84895(3,1): 1.52276 1.84058(3,2): 1.59605 1.81757(3,3): 1.64215 1.78405(3,4): 1.667 1.74337(3,5): 1.67685 1.6974(3,6): 1.67606 1.64693(3,7): 1.66728
1.59195(3,8): 1.65188 1.53176(3,9): 1.63019 1.4648

(4,0): 1.39844 1.79689(4,1): 1.47321 1.79039(4,2): 1.53357 1.77183(4,3): 1.57662 1.74337(4,4): 1.60371 1.70716(4,5): 1.61774 1.66474(4,6): 1.62143 1.61687(4,7):
1.61671 1.56358(4,8): 1.60469 1.50419(4,9): 1.58553 1.43694

(5,0): 1.37168 1.74337(5,1): 1.43185 1.73797(5,2): 1.48231 1.72225(5,3): 1.52081 1.6974(5,4): 1.54726 1.66474(5,5): 1.56279 1.62533(5,6): 1.56886 1.57972(5,7): 1.56663
1.52782(5,8): 1.55673 1.46876(5,9): 1.53893 1.40034

(6,0): 1.34392 1.68785(6,1): 1.39417 1.68314(6,2): 1.43711 1.66928(6,3): 1.47112 1.64693(6,4): 1.4957 1.61687(6,5): 1.51111 1.57972(6,6): 1.51799 1.53572(6,7): 1.51684
1.48453(6,8): 1.50775 1.42486(6,9): 1.48979 1.35359

(7,0): 1.3148 1.6296(7,1): 1.35786 1.62533(7,2): 1.39496 1.61267(7,3): 1.42492 1.59195(7,4): 1.44712 1.56358(7,5): 1.46144 1.52782(7,6): 1.46794 1.48453(7,7): 1.46659
1.43292(7,8): 1.45679 1.37092(7,9): 1.43641 1.29336

(8,0): 1.2838 1.5676(8,1): 1.3214 1.56358(8,2): 1.35382 1.55159(8,3): 1.38014 1.53176(8,4): 1.39975 1.50419(8,5): 1.41228 1.46876(8,6): 1.41736 1.42486(8,7): 1.41439
1.37092(8,8): 1.40169 1.30312(8,9): 1.37391 1.21014

(9,0): 1.25013 1.50026(9,1): 1.28341 1.49633(9,2): 1.3119 1.48453(9,3): 1.33487 1.4648(9,4): 1.35166 1.43694(9,5): 1.36167 1.40034(9,6): 1.36406 1.35359(9,7): 1.3572
1.29336(9,8): 1.33649 1.21014(9,9): 1.25159 1.00293
```

Figure 12. Lookup table for duty cycle (ON period in ms).

3.1.3. INTIALISING TIMER PARAMETERS

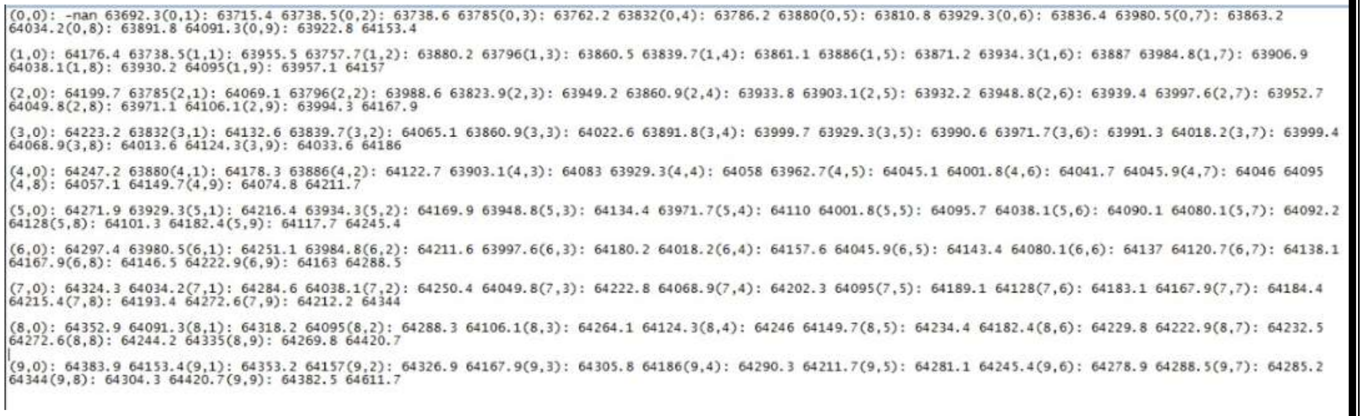
Timer mode -1

Step1: Delay/1.080806us = X

Step2: P=65536-X=Y

Step3: Y converted to hexadecimal =ABCD

Step4: TH1=0xAB; TL1=0xCD; [Refer **Figure 13** for further details]



```
(0,0): -nan 63692.3(0,1): 63715.4 63738.5(0,2): 63738.6 63785(0,3): 63762.2 63832(0,4): 63786.2 63880(0,5): 63810.8 63929.3(0,6): 63836.4 63980.5(0,7): 63863.2
64034.2(0,8): 63891.8 64091.3(0,9): 63922.8 64153.4

(1,0): 64176.4 63738.5(1,1): 63955.5 63757.7(1,2): 63880.2 63796(1,3): 63860.5 63839.7(1,4): 63861.1 63886(1,5): 63871.2 63934.3(1,6): 63887 63984.8(1,7): 63906.9
64038.1(1,8): 63930.2 64095(1,9): 63957.1 64157

(2,0): 64199.7 63785(2,1): 64069.1 63796(2,2): 63988.6 63823.9(2,3): 63949.2 63860.9(2,4): 63933.8 63903.1(2,5): 63932.2 63948.8(2,6): 63939.4 63997.6(2,7): 63952.7
64049.8(2,8): 63971.1 64106.1(2,9): 63994.3 64167.9

(3,0): 64223.2 63832(3,1): 64132.6 63839.7(3,2): 64065.1 63860.9(3,3): 64022.6 63891.8(3,4): 63999.7 63929.3(3,5): 63990.6 63971.7(3,6): 63991.3 64018.2(3,7): 63999.4
64068.9(3,8): 64013.6 64124.3(3,9): 64033.6 64186

(4,0): 64247.2 63880(4,1): 64178.3 63886(4,2): 64122.7 63903.1(4,3): 64083 63929.3(4,4): 64058 63962.7(4,5): 64045.1 64001.8(4,6): 64041.7 64045.9(4,7): 64046 64095
(4,8): 64057.1 64149.7(4,9): 64074.8 64211.7

(5,0): 64271.9 63929.3(5,1): 64216.4 63934.3(5,2): 64169.9 63948.8(5,3): 64134.4 63971.7(5,4): 64110 64001.8(5,5): 64095.7 64038.1(5,6): 64090.1 64080.1(5,7): 64092.2
64128(5,8): 64101.3 64182.4(5,9): 64117.7 64245.4

(6,0): 64297.4 63980.5(6,1): 64251.1 63984.8(6,2): 64211.6 63997.6(6,3): 64180.2 64018.2(6,4): 64157.6 64045.9(6,5): 64143.4 64080.1(6,6): 64137 64120.7(6,7): 64138.1
64167.9(6,8): 64146.5 64222.9(6,9): 64163 64288.5

(7,0): 64324.3 64034.2(7,1): 64284.6 64038.1(7,2): 64250.4 64049.8(7,3): 64222.8 64068.9(7,4): 64202.3 64095(7,5): 64189.1 64128(7,6): 64183.1 64167.9(7,7): 64184.4
64215.4(7,8): 64193.4 64272.6(7,9): 64212.2 64344

(8,0): 64352.9 64091.3(8,1): 64318.2 64095(8,2): 64288.3 64106.1(8,3): 64264.1 64124.3(8,4): 64246 64149.7(8,5): 64234.4 64182.4(8,6): 64229.8 64222.9(8,7): 64232.5
64272.6(8,8): 64244.2 64335(8,9): 64269.8 64420.7

(9,0): 64383.9 64153.4(9,1): 64353.2 64157(9,2): 64326.9 64167.9(9,3): 64305.8 64186(9,4): 64290.3 64211.7(9,5): 64281.1 64245.4(9,6): 64278.9 64288.5(9,7): 64285.2
64344(9,8): 64304.3 64420.7(9,9): 64382.5 64611.7
```

Figure 13. Lookup table for timer mode calculation (Y) (ON time in decimal).

3.2 SOURCE CODE

Assembly language code (in keil) :

```
ORG 0
MOV DPTR,#MYCOM
C1: CLR A
    MOVC A,@A+DPTR
      ACALL COMNWRT
      ACALL DELAY
      JZ SEND_DAT
      INC DPTR
      SJMP C1
SEND_DAT:MOV DPTR,#LINE1
D1: CLR A
    MOVC A,@A+DPTR
      ACALL DATAWRT
      ACALL DELAY
      INC DPTR
      JZ AGAIN
      SJMP D1
AGAIN:MOV DPTR,#LINE2
      MOV A,#0C0H
      ACALL COMNWRT
X1:   CLR A
      MOVC A,@A+DPTR
      ACALL DATAWRT
      ACALL DELAY
      INC DPTR
      JZ AGAIN1
      SJMP X1
AGAIN1:
MOV R7,#'X'
ACALL CHECK
MOV A,R7
MOV 60H,A
ACALL DATAWRT
MOV R7,#'X'
ACALL DELAY
MOV A,#','
ACALL DATAWRT
ACALL CHECK
MOV A,R7
MOV 61H,A
ACALL DATAWRT
MOV A,60H
CJNE A,#'6',CN1
MOV A,61H
CJNE A,#'1',CN2
ACALL CASE1    //6,1
CN1:
CN2:
MOV A,60H
CJNE A,#'9',CN3
MOV A,61H
```

```
CJNE A,#'3',CN4
ACALL CASE2  //9,3
CN3:
CN4:
MOV A,60H
CJNE A,#'2',CN5
MOV A,61H
CJNE A,#'8',CN6
ACALL CASE3  //2,8
CN5:
CN6:
MOV DPTR,#MYCOM
C12: CLR A
    MOVC A,@A+DPTR
     ACALL COMNWRT
     ACALL DELAY
     JZ SEND_DAT1
     INC DPTR
     SJMP C12
SEND_DAT1:
MOV DPTR,#FAIL
D12: CLR A
    MOVC A,@A+DPTR
     ACALL DATAWRT
     ACALL DELAY
     INC DPTR
     JZ AGAIN12
     SJMP D12
AGAIN12:SJMP AGAIN12

CASE1:
MOV 10H,#0FAH
MOV 11H,#0FBH
MOV 12H,#0F9H
MOV 13H,#0F1H
MOV 14H,#0C3H
MOV 15H,#014H
ACALL CASE_PRIME

CASE2:
MOV 10H,#0FBH
MOV 11H,#032H
MOV 12H,#0FAH
MOV 13H,#0BAH
MOV 14H,#0C2H
MOV 15H,#014H
ACALL CASE_PRIME

CASE3:
MOV 10H,#0F9H
MOV 11H,#0E3H
MOV 12H,#0FAH
MOV 13H,#06AH
MOV 14H,#0C3H
MOV 15H,#0B3H
ACALL CASE_PRIME
```

```
CHECK:
LOOP1:
ACALL KEY1
ACALL KEY2
ACALL KEY3
ACALL KEY4
CJNE R7,#'X',LOOP
SJMP LOOP1
LOOP:
RET

KEY1:
clr p3.4
MOV A,p3
ANL A,#0FH
MOV r2,A
cjne r2,#14,n1
MOV r7,#'7'
n1: cjne r2,#13,n2
mov r7,#'4'
n2: cjne r2,#11,n3
mov r7,#'1'
n3: cjne r2,#7,n4
mov r7,#'X'
n4: lcall delay
SETB P3.4
RET

; Checking for Key Press on the Second Column of 4x4 Matrix
KEY2:
clr p3.5
MOV A,p3
ANL A,#0FH
MOV r2,A
cjne r2,#14,q1
mov r7,#'8'
q1: cjne r2,#13,q2
mov r7,#'5'
q2: cjne r2,#11,q3
mov r7,#'2'; A=65
q3: cjne r2,#7,q4
mov r7,#'0'
q4: lcall delay
SETB p3.5
RET

; Checking for Key Press On The Third Column of 4x4 Matrix

KEY3:
clr p3.6
MOV A,p3
ANL A,#0FH
MOV r2,A

cjne r2,#14,w1
```

```
mov r7,#'9'
w1: cjne r2,#13,w2
mov r7,#'6'
w2: cjne r2,#11,w3
mov r7,#'3'
w3: cjne r2,#7,w4
mov r7,#'X'
w4: lcall delay
SETB p3.6
RET

; Checking for Key Press on the Fourth Column of 4x4 Matrix
KEY4:
clr p3.7
MOV A,p3
ANL A,#0FH
MOV r2,A
cjne r2,#14,e1
mov r7,#'X'
e1: cjne r2,#13,e2
mov r7,#'X'
e2: cjne r2,#11,e3
mov r7,#'X'
e3: cjne r2,#7,e4
mov r7,#'X'
e4: lcall delay
SETB p3.7
RET

CASE_PRIME:
HERE:
SETB P2.3
ACALL ON1
CLR P2.3
SETB P2.4
ACALL ON2
CLR P2.4
ACALL OFF
SJMP HERE

ON1:
MOV TMOD,#01H
MOV TH0,10H
MOV TL0,11H
SETB TR0
LP: JNB TF0,LP
CLR TR0
CLR TF0
RET

ON2:
MOV TMOD,#01H
MOV TH0,12H
MOV TL0,13H
SETB TR0
LP1: JNB TF0,LP1
```

```
CLR TR0
CLR TF0
RET

OFF:
MOV TMOD,#01H                    //1ST OFF-2ND ON
MOV TH0,14H                    // FFFF-2ND ON   + 1ST OFF
MOV TL0,15H
SETB TR0
LOOP2: JNB TF0,LOOP2
CLR TR0
CLR TF0
RET



COMNWRT:
MOV P1,A
CLR P2.1
CLR P2.2
SETB P2.0
ACALL DELAY
CLR P2.0
RET

DATAWRT:
MOV P1,A
SETB P2.1
CLR P2.2
SETB P2.0
ACALL DELAY
CLR P2.0
RET

DELAY: MOV R3,#250
HERE2: MOV R4,#255
HE:  DJNZ R4,HE
       DJNZ R3,HERE2
RET

ORG 300H
MYCOM: DB 38H,0EH,01,06,80H,0
LINE1: DB "2R MANIPULATOR",0
LINE2: DB "(X,Y)-",0
FAIL:  DB "TRY AGAIN!",0

END
```

3.3 PROTEUS IMPLEMENTATION

Given Cases: (9,3), (2,8),(6,1)

A specific case (2,8) is taken from the user input and is implemented through Proteus 8 Professional as shown in **Figure 14, Figure 15, Figure 16** and a failed case is shown in **Figure 17.**

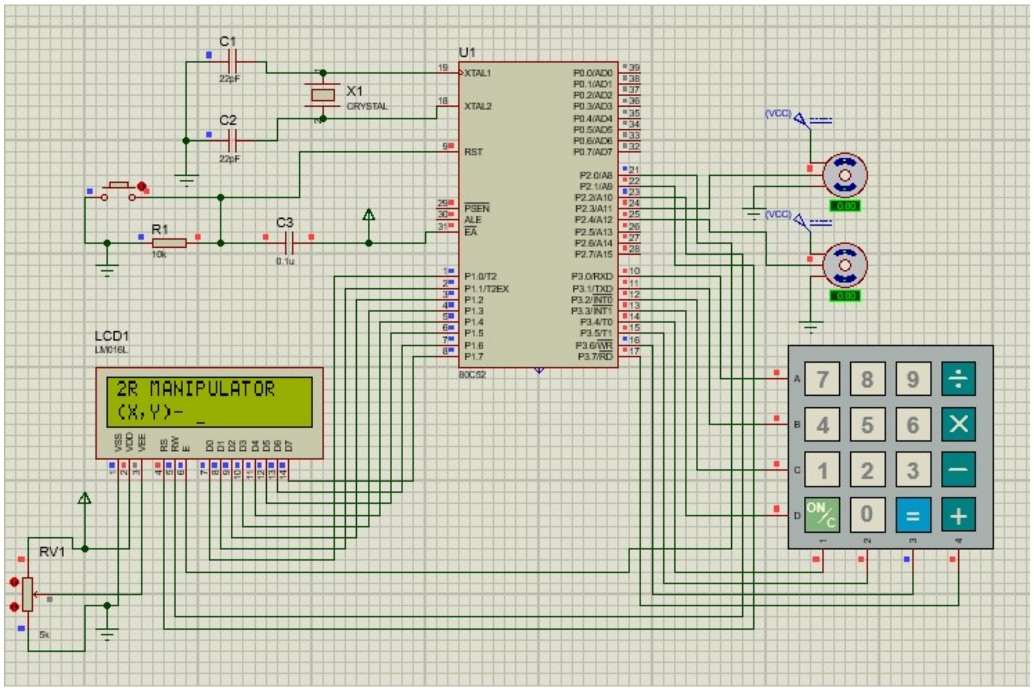


Figure 14. Proteus setup of 2R Manipulator.

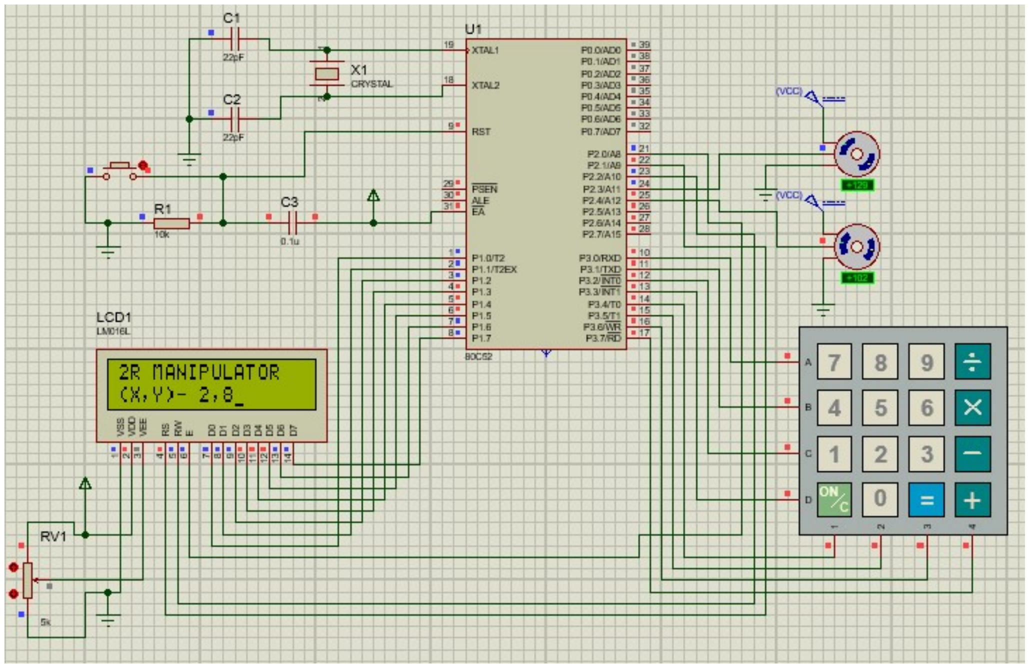


Figure 15. Output of correct case (Servo actuated).

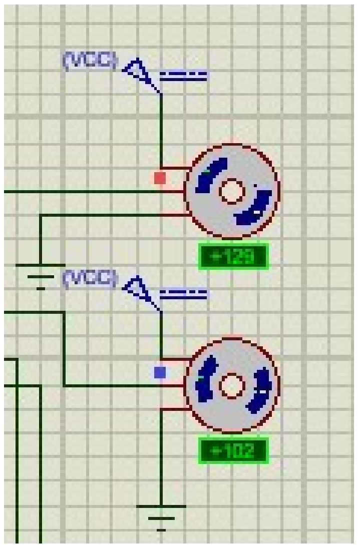


Figure 16. Servo Readings.

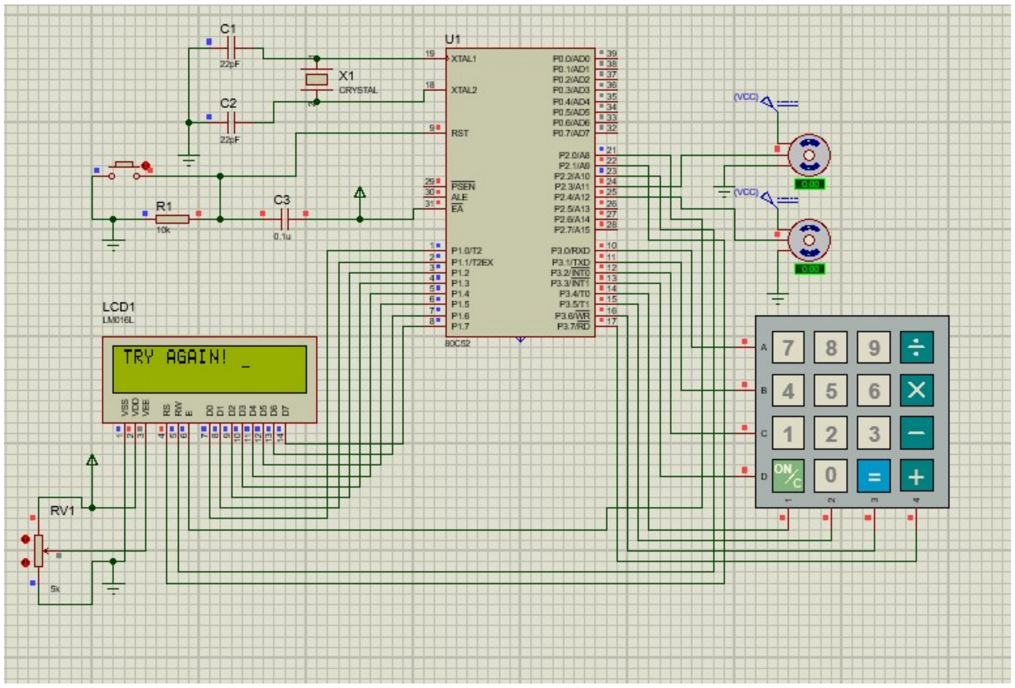


Figure 17. Output of incorrect case (Servo NOT actuated).

CODE ANALYSIS

- **Number of Lines: 266**

ERROR ANALYSIS

- **Error between the observed angle and the calculated angle in the proteus – 2.36%**

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

Therefore for a given input from the keypad (X,Y),the specified point is displayed in the LCD panel and the corresponding duty cycle is calculated and hence the servo is actuated using Pulse Width Modulation, with the required delays being assigned to the timer parameters.

Thereby the servos are actuated and as the end effect, the rod being pointed to the specified input point.

4.2 FUTURE WORK

This project will be further improved by directly implementing the inverse kinematic equations in High level language thereby reducing the error significantly. We can even use high processing power microcontrollers or microprocessor to reduce the error.

This idea of the project can also be implemented for high-end manipulating devices as robotic arms (2 link).

REFERENCES

[1] Dr. M. JAGANNATH, Associate professor, VIT Chennai, 2020

[2] http://www.sml.ee.upatras.gr/UploadedFiles/InverseKinematics.pdf

[3] https://link.springer.com/article/10.1186/s12938-017-0383-2

[4] https://www.researchgate.net/publication/224244534_A_new_geometrical_approach_to_solve_inverse_kinematics_of_hyper_redundant_robots_with_variable_link_length

[5] https://components101.com/motors/mg995-servo-motor

[6] https://www.electronicwings.com/sensors-modules/4x4-keypad-module

[7] https://en.wikipedia.org/wiki/Inverse_kinematics - :~:text=In%20computer%20animation%20and%20robotics,the%20start%20of%20the%20chain.

[8] https://www.indiamart.com/proddetail/8051-microcontroller-18928716988.html

[9] http://firmcode.blogspot.com/2014/12/introduction-of-8051.html

[10]https://components101.com/misc/4x4-keypad-module-pinout-configuration-features-datasheet

[11] https://components101.com/16x2-lcd-pinout-datasheet

[12] https://www.elprocus.com/8051-microcontroller-architecture-and-applications/