Point to Point Trajectory Control of 3R Planar Robot Arm

Moe Thu Zar, Wai Phyo Ei

***Abstract*— The functions of 3R Planar Robot Arm are described in this journal. It includes three main parts: first is calculation of inverse kinematics of robot arm, second is sending joint angle values to ARDUINO by serial communication and another is ARDUINO drive the arm. In which, three motors are used to move the arm. When the desired input (x, y and ϕ) are entered, inverse kinematic in the MATLAB program in PC is calculated joint angle values and then send it to the ARDUINO. This research used the MATLAB software to calculate the inverse kinematic of robot arm and also uses it as a graphical user interface (GUI) for controlling the movement of this robot. If the ARDUINO get the joint angle values, the motor drives the arm.**

***Index Terms*— Interface, MATLAB, Forward Kinematics, Inverse Kinematics, Trajectory.**

1) Introduction

Robotics education has been based on mobile robotics and manipulator-based robotics. Manipulator-based robotics education requires a large start up investment. This thesis is concerned with the fundamentals of robotics, including kinematics, computer interfacing, and control concepts as applied to industrial robot manipulators. The mathematical modelling of spatial linkages is quite involved. It is useful to start with planar robots because the kinematics of planar mechanisms is generally much simpler to analyze. A link parameter link length and link twist defines the relative location of the two axes in space. A link parameter is link length and link twist. The link offset is the distance between two consecutive links along the axis of the joint. The joint angle is the rotation of one link with respect to the next about the joint axis. These four parameters together are called Denavit-Hartenberg (D-H) parameters. The end effectors position can be controlled using either the joint angle or the link offset.

The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end-effector. The inverse problem of finding the joint variables in terms of the end effector position and orientation is the problem of inverse kinematics and it is in general more difficult than the forward kinematics problem. Deriving both forward and inverse kinematics is an important step in robot modelling based on Denavit -Hartenberg (D-H) representation [1].

One of the most common requirements in robotics is to move the end-effector smoothly from one pose to another pose. The control objective is to make the 3R robot manipulator traces desired trajectory. There are two approaches to generating such trajectories known respectively as joint-space and Cartesian motion. In this research, Cartesian space or operational space trajectory is used. PID tuning software method (eg. MATLAB) are used in this research.

Manuscript received Oct, 2017.

***Moe Thu Zar**, Mechatronic Engineering Department, Mandalay Technological University, Mandalay, Myanmar, +959786060301.*

***Wai Phyo Ei**, Mechatronic Engineering Department, Mandalay Technological University, Mandalay, Myanmar.*

2) System Block Diagram Of Planar Robot Arm

The complete system block diagram shown in Figure 1 consists of many parts like, personal computer with serial communication adapter, ARDUINO, servo, potentiometer and robot arm. Firstly, when the program is run, inverse kinematics in MATLAB is calculating the joint angle value and then sending data to ARDUINO that drives the robot arm. The main function of the ARDUINO is sending joint angle values to the servo motors, feeding the joint angle values from encoders of the servo motors and send back to the ARDUINO.

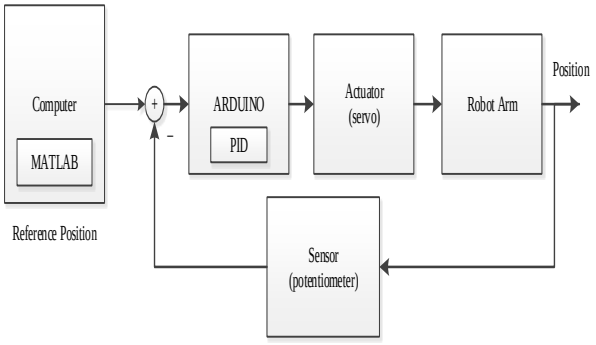


Figure.1. Overall Block Diagram of the System.

3) Robotic Analysis

The analysis of a robot consists of determining D-H parameters, calculating robot kinematics and dynamics, and controlling the robot via a control scheme. The control scheme may be PID, Fuzzy, NN or Visual Control.

i. Determining D-H Parameters

D-H parameters table is a notation developed by Denavit and Hartenberg, is intended for the allocation of orthogonal coordinates for a pair of adjacent links in an open kinematic system. It is used in robotics, where a robot can be modelled as a number of related solids (segments) where the D-H parameters are used to define the relationship between the two adjacent segments. The first step in determining the D-H parameters is locating links and determines the type of movement (rotation or translation) for each link.

Using D-H parameters defined in the previous steps in Table l, robot model was created in MATLAB software using the Robotic Toolbox. Robot model in addition to previously determined D-H parameters contains physical parameters which is using in the calculation of the dynamics movement.

TABLE 1

D-H PARAMETERS

link	α	a	d	θ
1	0	10.5cm	0	0<θ1<180
2	0	9.8cm	0	0<θ2<120
3	0	1cm	0	0<θ3<150

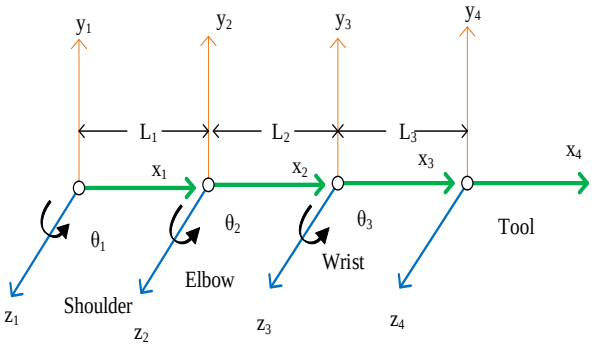


Figure 2. Coordinate Frame Assignment of Robot Arm

Since the robot has three revolute joints, the joint coordinate is

$$q_i = [\theta_1\ \theta_2\ \theta_3]$$

ii. *Direct Geometric Model*

The direct geometric model (DGM) represents the relations calculating the operational coordinates, giving the location of the end-effector, in terms of the joint coordinates. After establishing D-H coordinate system for each link as shown in Table 1, a homogeneous transformation matrix can easily be developed considering frame {i-1} and frame {i} transformation. So, the link transformation matrix between coordinate frames {i-1} and {i} has the following form [2];

$${}^{i-1}T_i = \mathrm{Rot}(z,\theta_i).\mathrm{Trans}(z,d_i).\mathrm{Trans}(x,a_i).\mathrm{Rot}(x,\alpha_i) \quad (1)$$

$${}^{i}T_{i+1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Where the four quantities, αi, ai, di, and θi are the parameters of link i and joint i.

The various parameters in previous equation are given the following names:

- a_i (Length) is the distance from z_i to z_{i+1}, measured along z_i;
- α_i (Twist), is the angle between z_i and z_{i+1} measured about x_i;
- d_i (Offset), is the distance from x_i to x_{i+1} measured along z_i; and
- θ_i (Angle), is the angle between x_i and x_{i+1} measured about z_i.

$${}^{0}T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$${}^{1}T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$${}^{2}T_3 = \begin{bmatrix} c_3 & -s_3 & 0 & l_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$${}^{3}T_{tool} = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$${}^{0}T_3 = \begin{bmatrix} c_{123} & -s_{123} & 0 & l_1c_1 + l_2c_{12} \\ s_{123} & c_{123} & 0 & l_1s_1 + l_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

Finally, to obtain ${}^{0}T_{tool}$, multiply ${}^{0}T_3$ and ${}^{3}T_{tool}$ then get,

$${}^{0}T_{tool} = \begin{bmatrix} c_{123} & -s_{123} & 0 & l_1c_1 + l_2c_{12} + l_3c_{123} \\ s_{123} & c_{123} & 0 & l_1s_1 + l_2s_{12} + l_3s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

where, $c_1 = \cos(\theta_1)$

$s_1 = \sin(\theta_1)$

$c_{12} = \cos(\theta_1 + \theta_2)$

$s_{12} = \sin(\theta_1 + \theta_2)$

$c_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$

$s_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$

Another generalized symbolic transformation matrix,

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

where, $n_x = c_{123}$

$n_y = s_{123}$

$n_z = 0$

$o_x = -s_{123}$

$o_y = c_{123}$

$o_z = 0$

$$a_x = 0$$
$$a_y = 0$$
$$a_z = 1$$
$$p_x = l_1c_1 + l_2c_{12} + l_3c_{123}$$
$$p_y = l_1s_1 + l_2s_{12} + l_3s_{123}$$
$$p_z = 0$$

iii. *Inverse Geometric Model*

Manipulator solution strategies into two broad classes: closed-form solutions and numerical solutions. Because of their iterative nature, numerical solutions generally are much slower than the corresponding closed-form solution. For most uses, are not interested in the numerical approach to solution of kinematics [3]. Within the class of closed-form solutions, distinguish two methods of obtaining the solution: algebraic and geometric. Inverse kinematic need to find the joint angle θ_1 , θ_2 and θ_3 ,corresponding end effector's position and orientation. For a planar motion, the position and orientation of the end effector can be specified by the origin of the frame 4, i.e (px, py) and the orientation of the frame attached to the end effector with respect to X axis, i.e. angle ϕ.

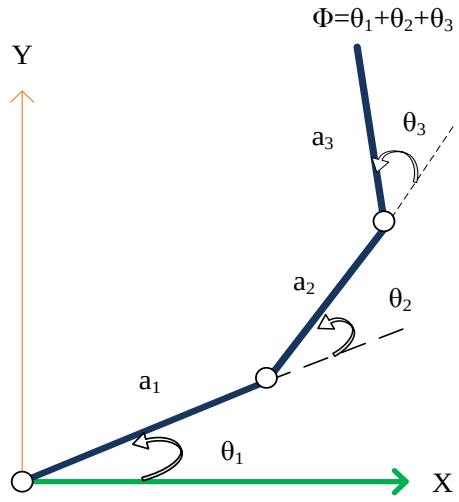


Figure 3. 3-DOF (3R) Planar Robot Arm

For the simulation of the inverse kinematics for 3R planar robot arm, these equations can be simplified as follows:

$$p_x = a_1 c_1 + a_2 c_{12} + a_3 c_{123} \quad (10)$$
$$p_y = a_1 s_1 + a_2 s_{12} + a_3 s_{123} \quad (11)$$
$$\phi = \theta_1 + \theta_2 + \theta_3 \quad (12)$$

The coordinate of wrist is w_x and w_y :

$$w_x = p_x - a_3\cos(\phi) = a_1 c_1 + a_2 c_{12} \quad (13)$$
$$w_y = p_y - a_3\sin(\phi) = a_1 s_1 + a_2 s_{12} \quad (14)$$

Squaring the two sides of equation (13) and (14),

$$w_x^2 + w_y^2 = a_1^2 + a_2^2 + 2a_1a_2\cos(\theta_2)$$
$$\cos(\theta_2) = \frac{w_x^2 + w_y^2 - a_1^2 + a_2^2}{2a_1a_2} \quad (15)$$

$$s_2 = \pm\sqrt{1 - c_2^2} \quad (16)$$
$$\theta_2 = a\tan 2(s_2, c_2) \quad (17)$$

By expanding $\cos(\theta_1 + \theta_2)$ and $\sin(\theta_1 + \theta_2)$ of equation (13) and (14), and rearranging them as:

$$w_x = (a_1 + a_2 c_2)c_1 - a_2 s_1 s_2 \quad (18)$$
$$w_y = (a_1 + a_2 c_2)s_1 + a_2 c_1 s_2 \quad (19)$$
$$s_1 = \frac{(a_1 + a_2\cos(\theta_2))w_y - a_2 s_2 w_x}{w_x^2 + w_y^2} \quad (20)$$
$$c_1 = \frac{(a_1 + a_2\cos(\theta_2))w_x + a_2 s_2 w_y}{w_x^2 + w_y^2} \quad (21)$$
$$\theta_1 = a\tan 2(s_1, c_1) \quad (22)$$
$$\theta_3 = \phi - \theta_1 - \theta_2 \quad (23)$$

4) Robot Hardware And Software

To achieve a control of a robot arm by using personal computer, it must make the connection between the robot and PC. This connection is called interface connection and it is done by using a ARDUINO. ARDUINO is designed to interface to and interact with electrical/electronic devices, sensors and actuators, and high-tech gadgets to automate systems. The complete control process can be divided into two categories, hardware and software.

A. *Hardware Environment*

The hardware environment for this thesis consists of a ARDUINO, a PC, and a data link between the PC and robot arm. The ARDUINO is a device that interfaces to sensors and robot actuators and performs embedded computing. The PC is used to control the robot by GUI; it is used to write the user defined embedded program which is to be run on the ARDUINO. In this thesis, use a serial communication link(wire) between the ARDUINO and PC.

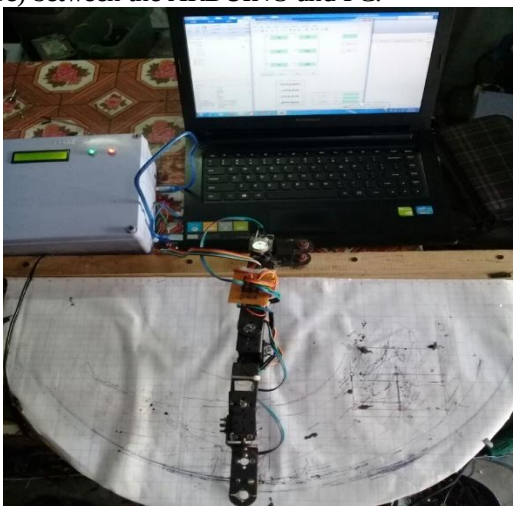


Figure.4. Hardware Environment

i. *Arduino*

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header,

and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

ii. Personal Computer

As previously mentioned, the PC is used to write Basic programs that the ARDUINO executes and to display sensory data processed by the ARDUINO and control the robot using MATLAB GUI. Any PC that supports MATLAB can be used.

iii. Servo Motor

Servo control in general can be broken into two fundamental classes of problems. The first class deals with command tracking. It addresses the question of how well does the actual motion follow what is being commanded. The typical commands in rotary motion control are position, velocity, acceleration and torque. The second general class of servo control addresses the disturbance rejection characteristics of the system. Disturbances can be anything from torque disturbances on the motor shaft to incorrect motor parameter estimations used in the feedforward control.

Figure.5. Servo Motor (MG995_Tower-Pro)

B. *Software Environment*

Software environment can be divided into two parts: the ARDUINO program and MATLAB Program. In ARDUINO program, write a code to make the interfacing between PC and arm. The MATLAB program consists of the Serial Communication code and the graphical user interface (GUI).

5) Operation Of The Control System

MATLAB and ARDUINO program is applied in this research.

When the inverse kinematic of the robot arm is calculated, MATLAB sends joint angle value as serial data to the robot. By entering each corresponding input, the operator can control simultaneously the robot movement. While performing these actions, the operator can check whether the command is correct or not with the help of LCD display.

All the motors are driven by the joint angle values received from the MATLAB in PC. ARDUINO is also used as the main microcontroller to interface with other devices by receiving the data. The robot is moved to the desired position with the help of three servo motors and another one servo motor is used to move the ball-pen up and down to write some numbers.

6) Algorithm Of Trajectory Control System

Straight-line motions are most common in the industrial applications; however, movement on a line is mostly obtained by specifying the discrete time joint- displacements at a constant time rate. The velocity and acceleration of the points can be calculated from the numerical approximation of the time derivatives. Several methods were used to compress the describing data of the trajectories as cubic, quintic and LSPB trajectory.

Cubic Trajectory: Cubic polynomial or third order polynomial approximation describes the path parametrically as a function of time with the position and velocity constraints at initial time t = zero and final time tf. It gives continuous positions and velocities at the start and finish points times but discontinuities in the acceleration. The derivative of acceleration is called the jerk. Higher order polynomials are required to guarantee the smoothness of the joint accelerations.

Quintic Trajectory: A discontinuity in acceleration leads to an impulsive jerk, which may excite vibration modes in the manipulator and reduce tracking accuracy. For this reason, one may wish to specify constraints on the acceleration as well as on the position and velocity. In this case, we have six constraints (one each for initial and final configurations, initial and final velocities, and initial and final accelerations). Therefore, a fifth order polynomial is required.

for distance:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 = q$$

for velocity:

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 = v$$

for acceleration:

$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 = \alpha$$

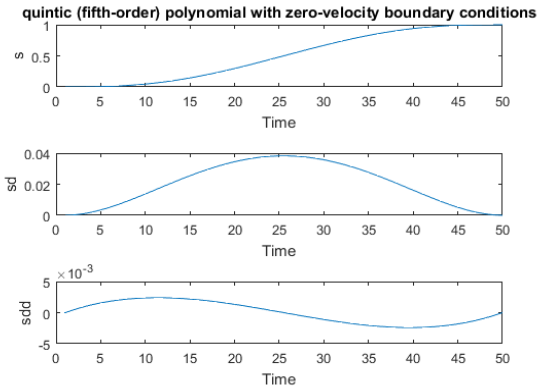


Figure 6. Quintic Polynomial Trajectory with Zero-Velocity Boundary Conditions [t_0=0, t_f=50 second, q_0=0, q_f=1, v=0 and α=0]

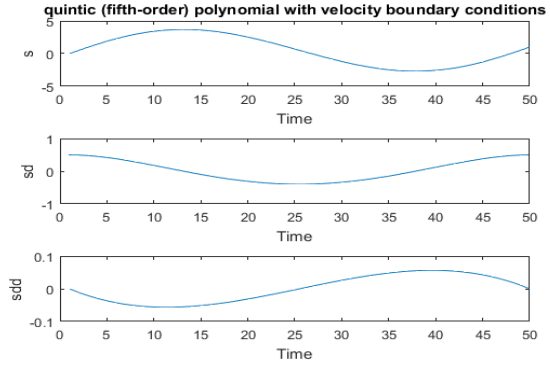


Figure 7. Quintic Polynomial Trajectory with Velocity Boundary Conditions [t_0=0, t_f=50 second, q_0=0, q_f=1, v_0=0.5, v_f=0 and α=0]

LSPB trajectory: Another way to generate suitable joint space trajectories is by so-called Linear Segments with Parabolic Blends or (LSPB) for short. This type of trajectory is appropriate when a constant velocity is desired along a portion of the path. The LSPB trajectory is such that the velocity is initially "ramped up" to its desired value and then "ramped down" when it approaches the goal position. To achieve this, we specify the desired trajectory in three parts. The first part from time t0 to time tb is a quadratic polynomial. This results in a linear "ramp" velocity. At time tb, called the blend time, the trajectory switches to a linear function. This corresponds to a constant velocity. Finally, at time tf − t b the trajectory switches once again, this time to a quadratic polynomial so that the velocity is linear [4][5].

$$q(t) = \begin{cases} q_0 + \frac{a}{2}t^2 & 0 \le t \le t_b \\ \frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \le t_f - t_b \\ q_f - \frac{at_f^2}{2} + at_f t - \frac{a}{2}t^2 & t_f - t_b < t \le t_f \end{cases}$$

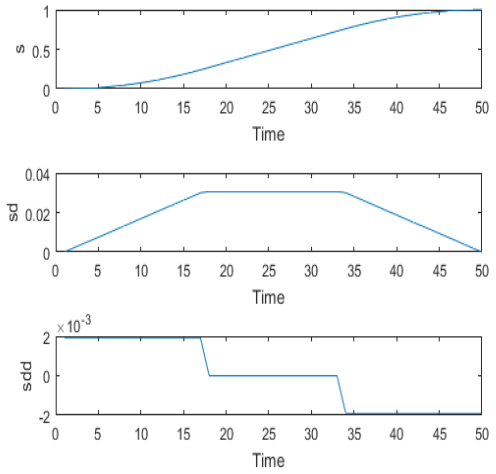


Figure 8. LSPB with Zero-Velocity Boundary Conditions [t_0=0, t_f=50 second, q_0=0, q_f=1, v=0 and α=0]

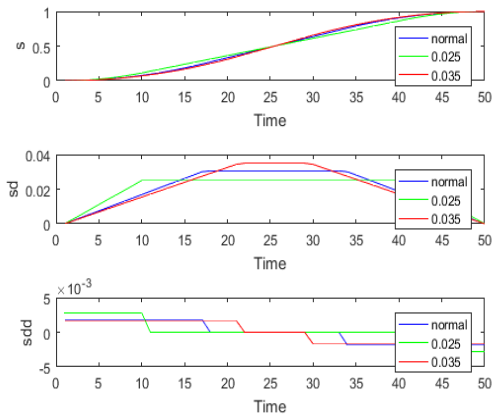


Figure 9. LSPB with Velocity Boundary Conditions [t_0=0, t_f=50 second, q_0=0, q_f=1, v_0[maximum]=0.035, v_0[minimum]=0.025, v_f=0, and α=0]

At home position all angles are zero, put θ1 = 0, θ2 = 0, θ3 = 0.

So the transformation matrix reduced to:

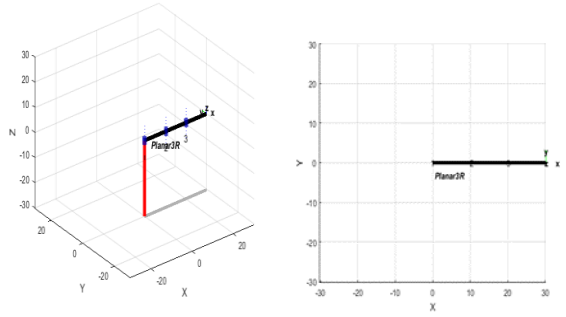
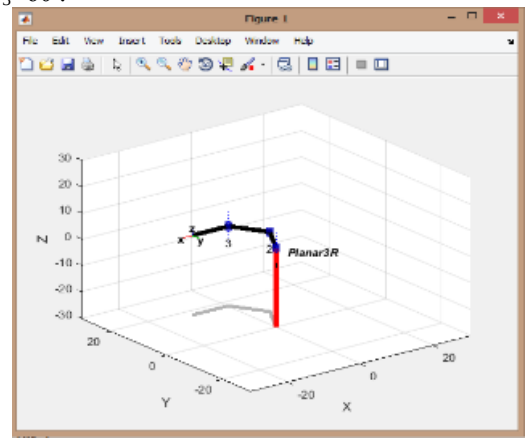
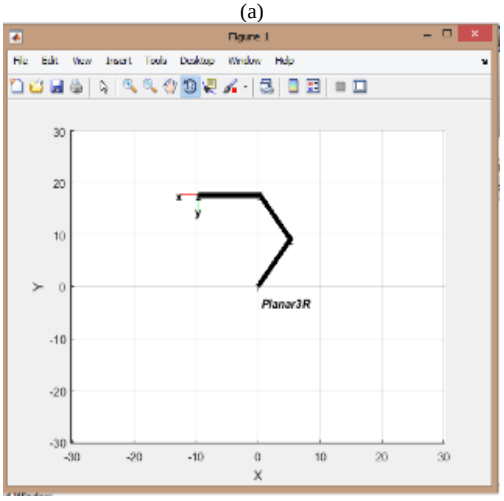


Figure 10. Visualization of robot in all zero joint angles in 3D and 2D

At this position all joint angles are 60˚, put θ_1=60˚, θ_2=60˚ and θ_3=60˚.



(a)



(b)

Figure.11. Visualization of robot in all 60˚joint angles [(a) 3D and (b) 2D view]

7) Results And Discussions

The user enters the input position into the left three edit boxes for moving the base motor, first joint motor and second joint motor. MATLAB program is calculated the joint angle values by using inverse kinematic equation of the robot arm and then send to the microcontroller. Microcontroller receives these signals and then the motors rotate according to the instructed direction. MATLAB simulation result of the robot arm is shown in Figure 12.

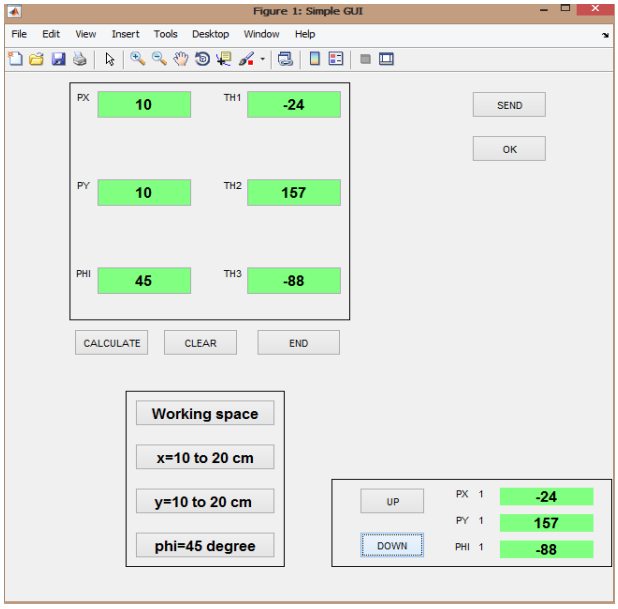


Figure.12. Simulation Result of Serial Interfacing for Robot Arm in MATLAB Form

From home position, (x=30.3cm, y=0 cm and ϕ=0) to goal points is shown in Figure 13, Figure 14 and Figure 15.

Figure.13. Testing Photo of Robot Arm Control System for Home Position (x=30.3cm, y=0 cm and ϕ=0)

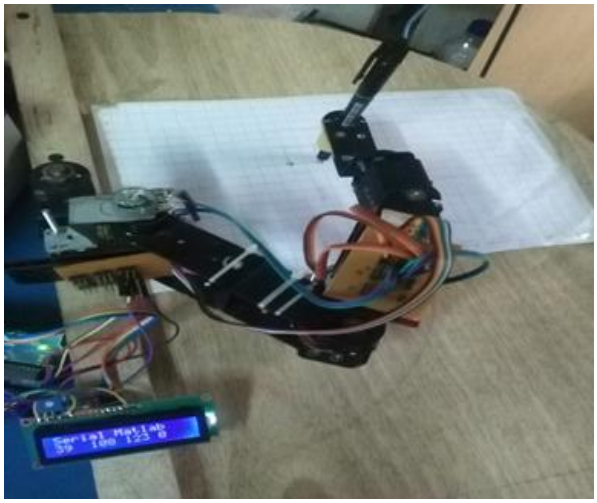


Figure.14. Testing Photo of Robot Arm Control System from Home Position to Goal Position (x=12 cm, y=10 cm and ϕ=90˚)

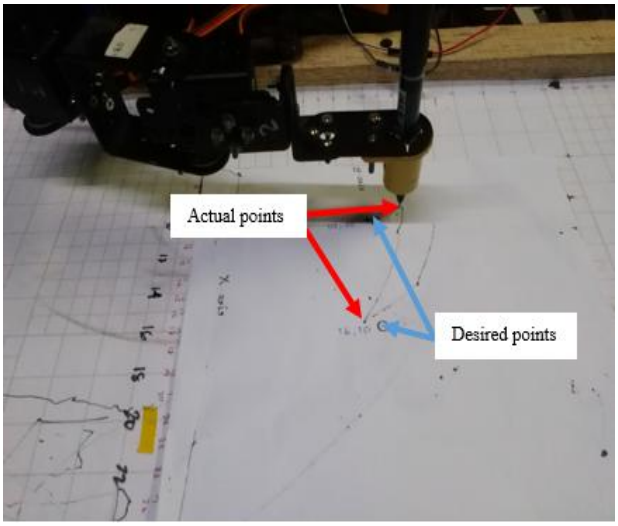


Figure.15. Testing Photo of Robot Arm Control System from Home Position to Goal Positions (x=16 cm, y=10 cm and x=10 cm, y=10 cm with φ=45˚)

The trajectory traced by the end effector with respect to original points. It is seen that at every point some error is occurring. This may be due to inaccuracies in the model setting including the motor clearance as well as due to the integer values of joint angles given to the ARDUINO program. The precision of robot arm is $\pm$ 2 cm.

Since MATLAB is slow in the execution time, recommend using a high-level computer programming language to perform the software part.

CONCLUSIONS

This journal represents a writing robot using PC based point to point trajectory control and it is designed for only as a sample. The Graphical User Interface (GUI) of the software package was developed for testing motional characteristics of the robot arm. A physical interface between the robot arm and the GUI was designed and built. A comparison between kinematics solutions of the virtual arm and the robot's arm physical motional behaviours have been accomplished. The results are displayed in a graphical format and the motion of all joints and end effector can be observed. Many future developments can be carried on this robot arm like other types for robots, these developments include path-planning, dynamics modelling, force control and computer vision.

ACKNOWLEDGEMENT

The author is deeply gratitude to Dr. Sint Soe, Rector of Mandalay Technological University (MTU) for his kind permission to submit this research. The author would like to thank to her supervisor, Daw Wai Phyo Ei, Lecturer of Mechatronic Engineering Department, Mandalay Technological University, for her valuable supervision guidance and comments for preparation of the research. After that, the author would like to thank Dr. Sint Soe, Associate Professor and thanks to all her teachers from Mechatronic Department, Mandalay Technological University.

REFERENCES

[1] MAHMOUD A. ABUALSEBAH, "*Trajectory Tracking Control of 3-Dof Robot Manipulator Using Tsk Fuzzy Controller*", AUGUST 2013.

[2] J J. DENAVIT, R.S. HARTENBERG, "*A Kinematics Notion For Lower-Pair Mechanisms Based On Atrices*", ASME JOURNAL OF APPLIED MECHANICS, VOL. 22, PP. 215–221, 1955.

[3] PAUL R.C.P., "*Robot Manipulators: Mathematics, Programming and Control*", MIT PRESS, CAMBRIDGE, 1981.

[4] MOHAMMED REYAD ABUQASSEM, "*Simulation and Interfacing of 5 Dof Educational Robot Arm*", DEGREE OF MASTER OF SCIENCE IN ELECTRICAL ENGINEERING, JUNE 2010
[5] PETER CORKE, "*Robotics, Vision and Control*".