2021

Summer Internship report
3 DOF robotic arm

Internship In-charge:

Dr. Abid Imran

Submitted By:

Waqar Ahmad____ 2018506

Abdul Qadeer____ 2018010

Ghulam Ishaq Khan Institute of Engineering Sciences and Technology

Table of Contents

1. Acknowledgment:

In the grace of almighty Allah, we got the chance to complete summer internship at Ghulam Ishaq Khan Institute of Engineering Sciences and Technology(GIKI). It was a great chance for our professional development and learning new things, as this was our first time in a robotics environment or industry. We are very thrilled to meet qualified and wonderful instructor **Dr. Abid Imran**, who help us and guided throughout our internship period.

We would like to take this opportunity to express gratitude to **Dr. Abid Imran**, internship in-charge, who took time out of his busy schedule and duties to teach and guide us at every step of the way, correcting the path so that we could carry out the responsibilities correctly at the labs this educational organization.

We are also thankful to Mr. Umair shafiq, who welcomed us pleasantly at LAbs and helped from deep of his heart all the learning matterial and hardware needed for completing this internship.

2. Short summary:

In the first two week of internship we have study the forward Kinematic, Reverse Kinematics of the robotic arm and path planning and taking mentorship from Dr abid Imran with time.

Find the DH table of three Degree of freedom Robotic arm which is mention below. After that find the transform function of each joint with respect to its local base. After that Apply forward kinematic to find the end effector position with respect to global base, where by applying the reverse kinematics, find the required angle of the servo motor for the target position. In the last Path planning is done for Robotic arm.

Arduino is being used to control the three degree of freedom robotic arm. All the above equation is code in Matlab and then connect with Arduino and the result was fantastic.

3. Introduction:

The advancement of industrial automation has been boosted greatly by computer-aided design and manufacturing. In the field of industrial automation, robotic manipulators have played a significant role. The design of robotic arms is a popular study topic these days. The high-DOF robots are utilized to replace a sequence of processes in a manufacturing process, they come very close to being a direct replacement for a high-DOF human.

Forward kinematics determines end-effector position and orientation from a given set of joint angles, whereas inverse kinematics does the opposite. End-effector motion is specified in Cartesian coordinates in robotic applications. However, because the dynamics of the manipulator are described in terms of these joint parameters, the robotic arm's motion is specified in terms of joint angles. Finding the desired joint space trajectories given the Cartesian trajectories is the task of trajectory generation. Inverse kinematics is used to accomplish this. The Denavit-Hartenberg convention and the screw theory technique are two methodologies that are extensively utilized for kinematic modelling of robotic manipulators.

To compute the forces and torques required by the actuators to complete the intended tasks, a mathematical model of the robotic arm is required. The joint torques were computed using the Euler-Lagrange formulation in this paper, and the calculated torques were compared to the torques acquired using computer simulation programmes.

4. Report:

4.1. General Characteristics of Robot Arm Mechanics:

Kinematics in robotics is the science of motion investigation. Robot arm links can be rotated or offset according to the reference coordinate frame. A systematic and general approach developed by Denavit and Hartenberg establishes the relationship between the robot endpoint and the total displacement of robot arm links. Angular and linear displacements between limbs are called joint coordinates and are defined by limb variables. In order to determine the amount of rotation and displacement according to the reference coordinate system of the endpoint, the matrices A which represent the amounts of each limb rotation and displacement are multiplied in turn. If the coordinates of the end point are given, limb variables can be obtained by going backward. These operations are called forward and inverse kinematics. The next section will explain how to determine forward and reverse kinematics. The general transformation matrix can be quite complex even for simple robots.

4.2. Denavit–Hartenberg:

The Denavit–Hartenberg parameters (also known as DH parameters) are four parameters related with a specific convention for attaching reference frames to the links of a spatial kinematic chain, or robot manipulator, in mechanical engineering.

Here is DH table for our 3D of freedom robotic arm.

#	θ(radian)	a(cm)	d(cm)	α
1	t1	-2.5	10	0
2	t2	0	0	0
3	t3 - $\frac{\pi}{2}$	0	10	0
4	0	11	0	0

4.3. Transformation vector for link:

Here is transformation vector for individual link.

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

The below function will take link parameter as input and it will be give the transformation vector for that link, which will further use to find the position of end effector according to base.

```
function [T]= Transformation(t,a,l,d)
T= [   cos(t)           -sin(t)          0          l   ;
      sin(t)*cos(a)  cos(t)*cos(a)  -sin(a)  -d*sin(a);
      sin(t)*sin(a)  cos(t)*sin(a)  cos(a)    d*cos(a);
      0                 0              0         1] ;
```

a ➔ length of the common normal

b ➔ offset along previous z to the common normal

α ➔ angle about common normal, from old z axis to new $z\ axis$

θ ➔ angle about previous z, from old x to new x

4.4. Path planning function:

Trajectory planning is moving from point A to point B while avoiding collisions over time. This can be computed in both discrete and continuous methods. Trajectory planning is a major area in robotics as it gives way to autonomous vehicles. For robotic arm our concern is that the robotic arm goes with short possible path.
In general, cubic equation is using for the trajectory planning of robotic arm which is given below:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3,$$

Where the boundary condition is given below:

$$\theta(0) = \theta_0,$$
$$\theta(t_f) = \theta_f.$$

$$\dot{\theta}(0) = 0,$$
$$\dot{\theta}(t_f) = 0.$$

Higher-order polynomials are sometimes used for path segments. For example, if we wish to be able to specify the position, velocity, *and* acceleration at the beginning and end of a path segment, a quintic polynomial is required.

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

The code for cubic equation is given below:

```
function [a]= Trajectory(theta0,theta1,t)
        syms a2 a3 x
        % x represent theta
        theta= theta0+a2*x^2 + a3*x^3;
        theta_dot= diff(theta);

        %a1=0
        %a0=theta0
        E = theta1==subs(theta,x,t);
        E1 = 0==subs(theta_dot,x,t);
                        a=solve(E,E1);
```

Function parameters:

Theta0 ➔ Initial theta or motor angle

Theta1 ➔ Target position angle

t ➔ time of flight

4.5. Main code:

Define variable for code which will already mention in DH table.

```
clc,clear
syms t1 t2  t3 a1 a2  l1 l2 l3  d1
```

Putting the respective values of angles and length in transformation vectors. And the result picture is given for each vector.

```
T0_1 = Transformation (t1,      0     ,-2.5   ,10)
T1_2 = Transformation (  t2  , a2  ,0     ,0)
T2_3 = Transformation (    t3-pi/2  ,0     ,10    ,0)
T3_4 = Transformation (   0   ,0     ,11 ,0)
```

Orientation of end effector from base:

```
T0_4= T0_1*T1_2*T2_3*T3_4
```

same basic operation to visualize or sample the equations:

```
T0_4= subs(T0_4,cos(a2),0);
T0_4= subs(T0_4,sin(a2),1);
T0_4= subs(T0_4,sin(t2 + pi/2),cos(t2));
T0_4= subs(T0_4,cos(t2 + pi/2),-sin(t2));
p = T0_4(1:3,4)
```

Position of end effector from base:

```
Px= T0_4(1,4)
Py= T0_4(2,4)
Pz= T0_4(3,4)
p = T0_4(1:3,4)'

subs(p,[t1 t2 t3],[0.1*pi pi/2 0.5*pi])
```

Intial thetas is zeros:

```
tem1=0
tem2=0
tem3=0
```

The continues loop so that the code run for ever:

```
syms time t
while(1)
        theta0_1= tem1
        theta0_2= tem2
        theta0_3= tem3
```

taking the final position of the end effector as input:

```
        theta1_1= input("enter theta 1   ")
        theta1_2= input("enter theta 2   ")
        theta1_3= input("enter theta 3   ")
        t1  = input("time     ")

        tem1=theta1_1
        tem2=theta1_3
        tem3=theta1_2
```

Pass the final, initial position and the time interval from in the trajectory function for path planning:

```
        a=Trajectory(theta0_1,theta1_1,t1)
        theta=theta0_1+vpa(a.a2)*t^2 +vpa(a.a3)*t^3
```

```
a=Trajectory(theta0_2,theta1_2,t1)
theta2=theta0_1+vpa(a.a2)*t^2 +vpa(a.a3)*t^3
a=Trajectory(theta0_3,theta1_3,t1)
theta3=theta0_1+vpa(a.a2)*t^2 +vpa(a.a3)*t^3
```

code for the cunection of Ardiuno and Matlab:

```
a = arduino('COM4', 'Uno', 'Libraries', 'Servo');
s = servo(a, 'D4')
g = servo(a, 'D5')
e = servo(a, 'D6')
```

Angle input to the servo motor:

```
for ti = 0:0.1:t1
        angle = double((subs(theta,t,ti))/180)
        writePosition(s, angle);
        angle2 = double((subs(theta2,t,ti))/180)
        writePosition(g, angle2);
        angle3 = double((subs(theta3,t,ti))/180)
        writePosition(e, angle3);
        current_pos = readPosition(s);
        current_pos = current_pos*180;
        fprintf('Current motor position is %d degrees\n',
        current_pos);
        pause(0.2)
    end
end
```