Autonomous Robots
CSE-5364
Team Final Project

Team No: 1.
Team Members:

NAME	UTA ID
Muhammad Tayyab	1001256129
Srinivas Bavisetti	1001279350
Shivam Bijoria	1001359394
Saif Sayed	1001275259

Problem statement:

The goal of this project is to build and program a robot arm that can draw polygons using a marker. Here you should build a robot arm that can move a marker mounted at its end across a piece of paper to draw arbitrary polygons given to it (at compile time). Given that polygon, the robot should move the marker to the first corner (without drawing a line) and then trace the shape on the paper. Drawing will be limited to a 15cm x 10cm area that can be located based on the kinematic characteristics of the robot constructed.

Working model

You can view the working model of the project on the following link:
https://youtu.be/kWPcrv_SmeI

Tools and Languages used:

We used Matlab to simulate the robotic arm and calculate inverse and forward kinematics of the arm. For coding in the next brick, we initially used C language but the Math library was not working. We then switched to C++ language but Math library still did not work. So we had to write lookup table for **atan** to solve the problem. Moreover created our own functions for square root and atan2.

Robot Design

The main goal when designing robot arm was to attain maximum stability, so we tried 3 different designs. Our first design had 3 degrees of freedom but implementing inverse kinematics on this was hard, therefore we changed it to 2 degrees of freedom. However, this design gave us stability issues as the distribution of mass was not at the center. After making changes to this we ended up getting a robot arm that was stable. Following are the images of three designs:

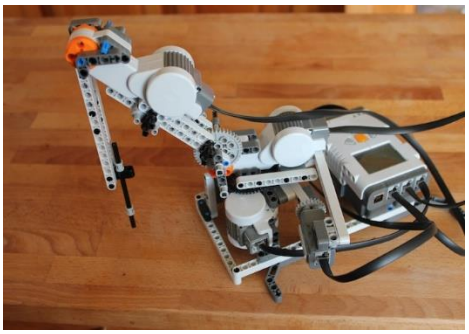


Fig.1 first design

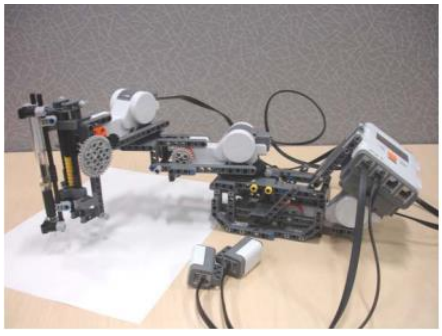


Fig.2 second design

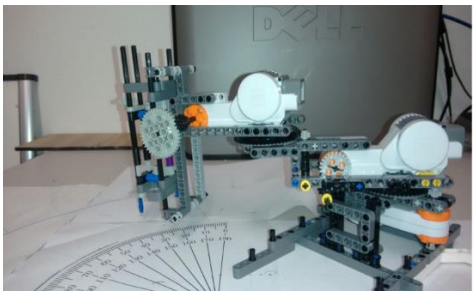


Fig.3 final design

.

Motor Calibration:

There are 2 revolute joints present in the robotic arm which are required to be calibrated. This calibration for robotic arm is done such that we reduce the error between the goal position and the position where our arm moved. It is also done so that we could make the robot arm to move to the designated position with highest precision. Calibration is done one joint a time. For each joint we give a known encoder count and make the motor run until it reaches that given threshold. Once the motor reaches the desired encoder count we then measure the angle of the arm on that particular joint with its previous position and calculate the relative angle. With the given angle and known encoder count we calculate number of encoder counts per degree and use this values that make the each joint to turn to our required angle. We repeat this process multiple times. We also check for reverse motion i.e. if we move by 20 degrees in clockwise direction and then 20 degrees in anti-clockwise direction, we should be at starting position.

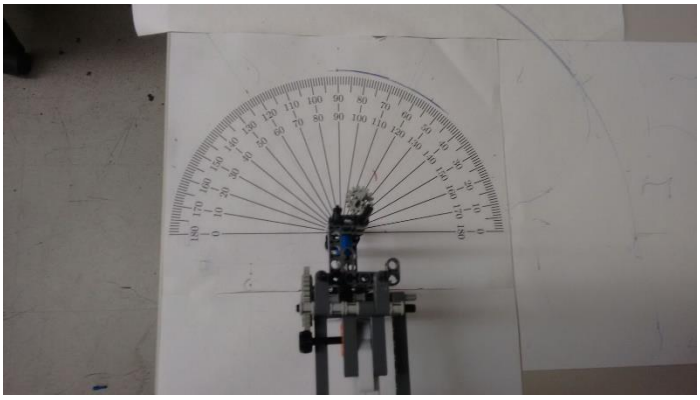


Fig.4 Calibrating the robotic arm

Polygon Creation and Straight line Creation:

We want our robotic arm to create polygon with angles and coordinates we provide.

For this, we input the x and y values of points of the polygon. While moving from a point A to next point B on the polygon we first take the slope of line connecting A and B (which is $(Y_b-Y_a)/(X_b-X_a)$).Then we create n points on this line, moving along slope from A. Moving along these lines we create a straight line. Once our robotic arm knows how to make a straight line, it can develop a polygon with coordinates and angles we give it.

We first simulated the robotic arm polygon formation on Matlab by calculating 1) forward kinematics 2)Inverse kinematics and using these two to form the desired polygon. We designed the robotic arm having 2 degrees of freedom (our final design) and used the same design approach on the Matlab. We start with giving the polygon input coordinates x,y and plot the graph. This gave us the polygon shown in the figure below. We used this as our reference shape and simulated movement of our robotic arm such that it traces the polygon and creates the exact shape. This is shown in the figure below.

The red points in figure 2 gives the original (desired) coordinates of the polygon. The black points are the final points of our robotic arm. As we can see the black points coincides the red points, showing that our robotic arm created the perfect polygon.

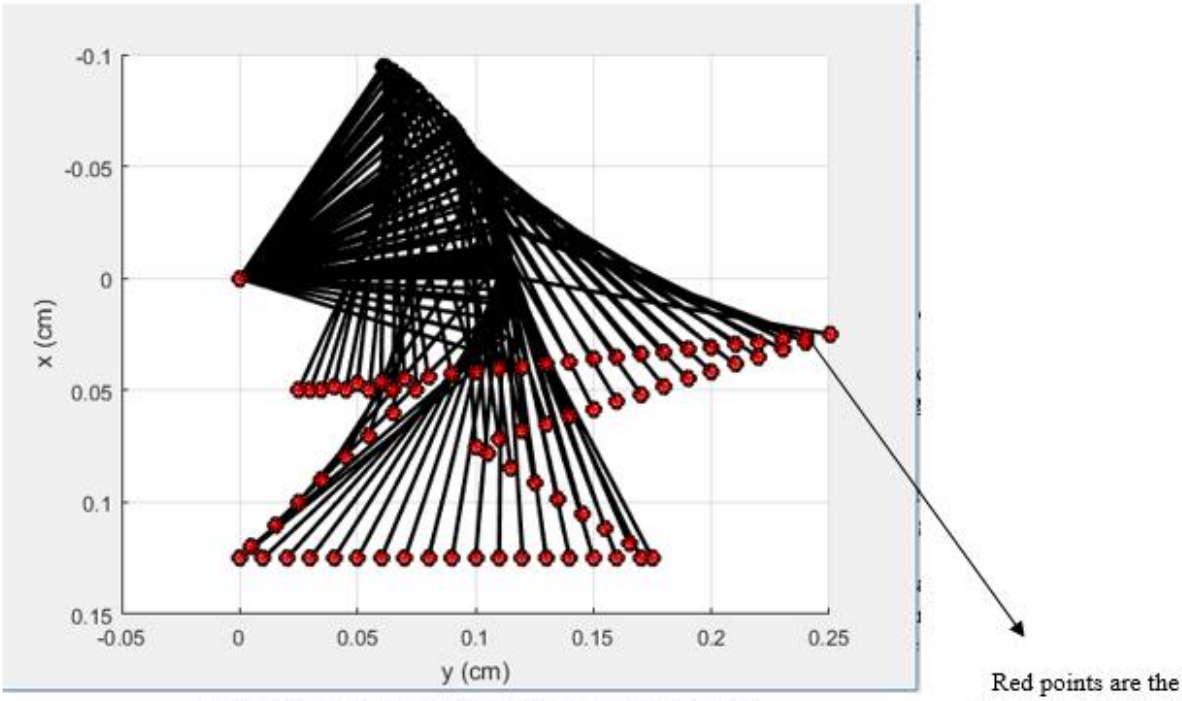


Fig. 5 traced polygon and desired polygon coordinates

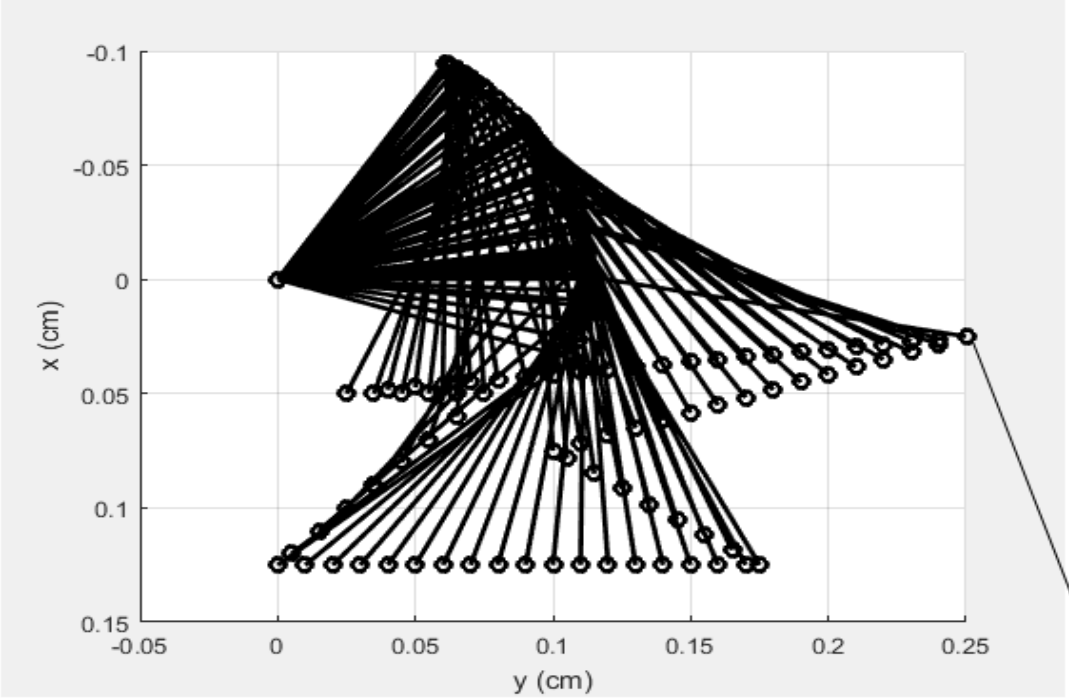
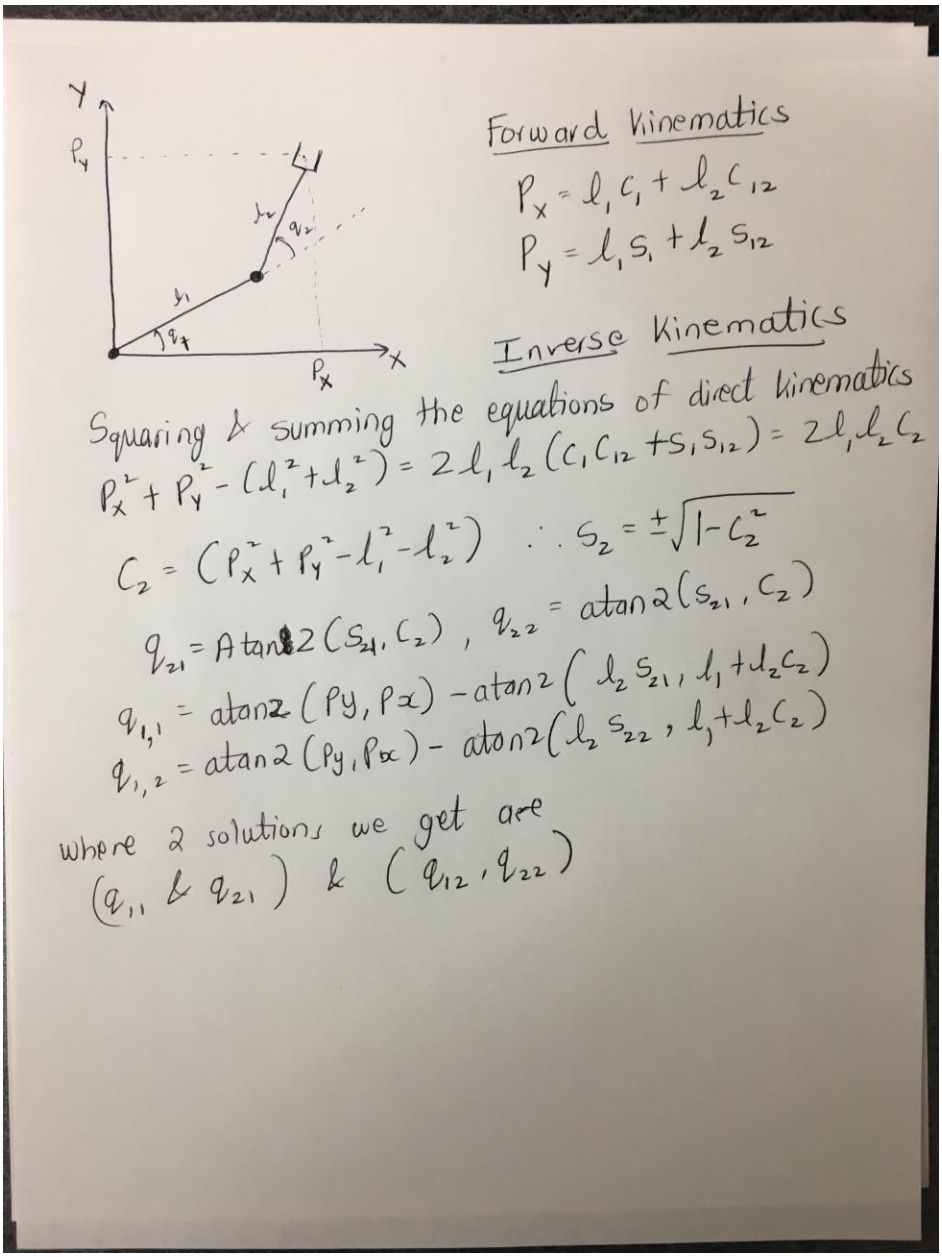


Fig.6 final positions of the arm

Black points give the final position of our robotic arm

Inverse Kinematics:



Forward Kinematics

$$P_x = l_1 c_1 + l_2 c_{12}$$
$$P_y = l_1 s_1 + l_2 s_{12}$$

Inverse Kinematics

Squaring & summing the equations of direct kinematics

$$P_x^2 + P_y^2 - (l_1^2 + l_2^2) = 2 l_1 l_2 (c_1 c_{12} + s_1 s_{12}) = 2 l_1 l_2 c_2$$

$$c_2 = (P_x^2 + P_y^2 - l_1^2 - l_2^2) \quad \therefore s_2 = \pm\sqrt{1 - c_2^2}$$

$$q_{21} = A\tan2(s_{21}, c_2), \quad q_{22} = \operatorname{atan2}(s_{21}, c_2)$$

$$q_{1,1} = \operatorname{atan2}(Py, Px) - \operatorname{atan2}(l_2 s_{21}, l_1 + l_2 c_2)$$

$$q_{1,2} = \operatorname{atan2}(Py, Px) - \operatorname{atan2}(l_2 s_{22}, l_1 + l_2 c_2)$$

where 2 solutions we get are
$(q_{11}$ & $q_{21})$ & (q_{12}, q_{22})

Results:
We are giving input coordinates of a rectangle to our robotic arm. When we run its Matlab simulation we get the correct rectangle as shown below:

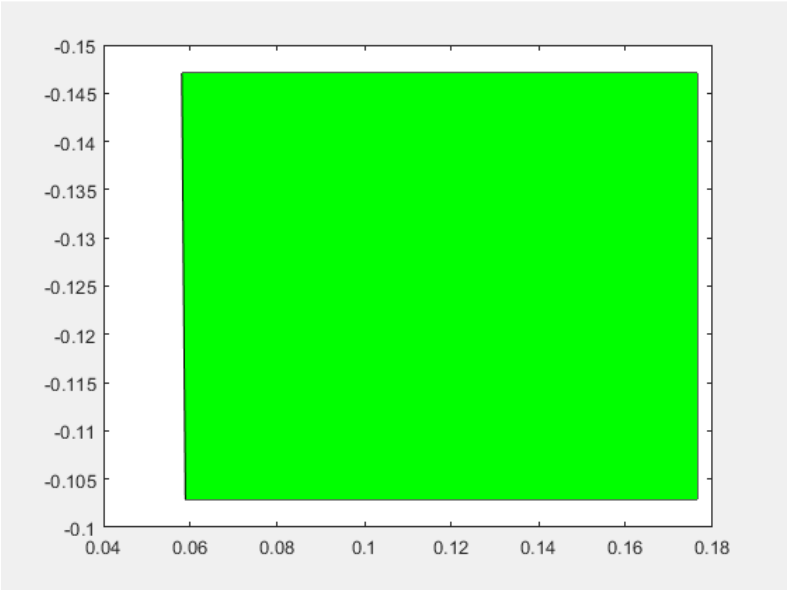


Fig.7 origial rectangle corrdinates

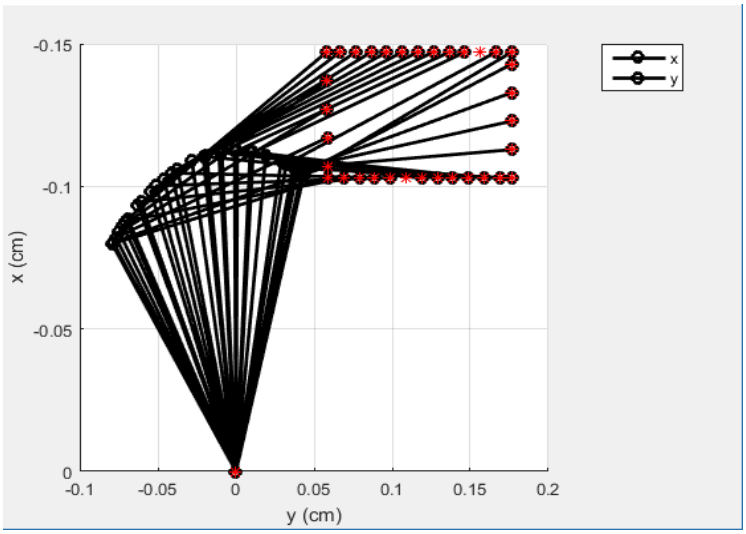


Fig.8 plotted on Matlab simulation

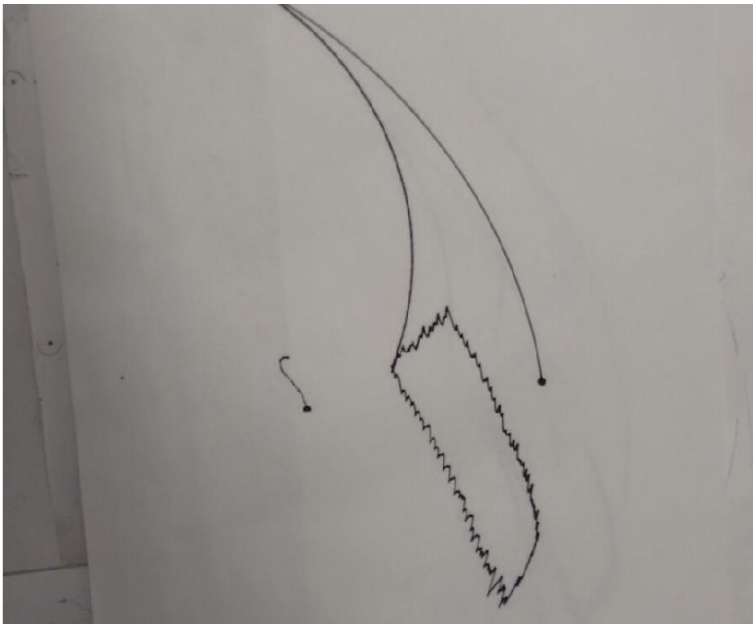


Fig.9 plotted by physical robotic arm

We compared the results of Matlab and c program and the joint angles from the inverse kinematics were identical. However, there is a skew in the plot which is due to error in the starting angle of the robot. This can be improved by conducted multiple trials and finding the best initial coordinates.