Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar, Gujarat

IE-410

Introduction to Robotics

(Prof. Tapas Kumar Maiti)

Group-8:

Meet Joshi → 202201065

Het Gandhi → 202201167

Harsh Bosamiya → 202201243

Dishant Patel → 202201260

Assignment 1: Inverse Kinematics in the Robotic Arm

1) **Aim:** To implement **Inverse Kinematics** in the Robotic Arm.

2) **Material used:**
   i) Arduino Compatible Shield
   ii) Arduino Tinkerkit Braccio Robotic Arm
   iii) Arduino Uno Microcontroller with USB cable
   iv) Power adapter

3) **Code:**

```
#include <Braccio.h>
#include <Servo.h>

Servo base;
Servo shoulder;
Servo elbow;
Servo wrist_rot;
Servo wrist_ver;
Servo gripper;

// Lengths for the link (in mm) for the Braccio Robotic Arm
const float L1 = 125; // Length of link 1 (shoulder to elbow)
const float L2 = 125; // Length of link 2 (elbow to wrist)
const float L3 = 71.5; // Length of link 3 (wrist to end-effector)

void setup()
{
    Serial.begin(9600); // Initializes the serial monitor
    delay(1000);
    Braccio.begin(); // Initializes the Braccio Robotic Arm position
}

void loop()
{
    // These values are changeable according to different coordinates
    float x=0, y=314, theta=90;
    // Calls inverse kinematics function to calculate the link angles
    inverseKinematics(x, y, theta);

    delay(1000); // Waits for 1 second
}

void inverseKinematics(float x, float y, float theta)
{
    // Calculates position of J3
    float a = x - (L3 * cos(radians(theta)));
    float b = y - (L3 * sin(radians(theta)));
    float C = sqrt(pow(a, 2) + pow(b, 2));

    // Calculates angles Alpha and Beta
    float alpha = degrees(acos((pow(L1, 2) + pow(C, 2) - pow(L2, 2)) /
(2 * L1 * C)));
    float Beta = degrees(acos((pow(L1, 2) + pow(L2, 2) - pow(C, 2)) /
(2 * L1 * L2)));

    // Calculates angle theta1 for shoulder-elbow link
    float theta1 = degrees(atan2(b, a)) - alpha;

    // Calculates angle theta2 for elbow-wrist link
    float theta2 = 180 - Beta;

    // Calculates angle theta1 for wrist-end-effector link
    float theta3 = theta - theta1 - theta2;

    // Sets servo angles and move the Braccio Robotic Arm according to
the given position
    Braccio.ServoMovement(20, 0, int(theta1), 90+int(theta2),
90+int(theta3), 0, 73);

    // Prints the values of angles in the serial monitor
    Serial.print("Theta 1: ");
    Serial.println(theta1);
    Serial.print("Theta 2: ");
    Serial.println(theta2);
    Serial.print("Theta 3: ");
    Serial.println(theta3);


}
```

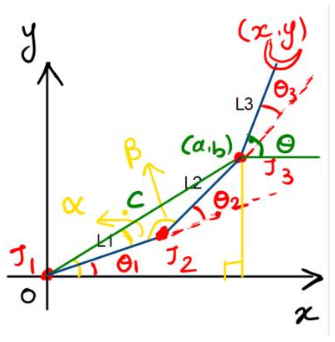
4) Procedure:

➔ Basically, in Inverse kinematics, we are given the end-effector coordinates (x, y) in the case of 2-D. Then we have to find the angle values of the individual links Theta1, Theta2, etc.

➔ In the case of 3-link **Inverse Kinematics**, we also require an angle of the 3rd link respective to the horizontal.

➔ Firstly, we connected the Robot's microcontroller with the Arduino Uno microcontroller.

➔ Then we derived the calculation part of **Inverse Kinematics** for a 3-link Robotic Arm shown in the video, which is also attached below.



We are given the coordinates of the end-effector (x, y) and an angle of the 3rd link θ.

Now, with the help of (x, y) we will find the coordinates of the joint J3 (a, b).

From the diagram, we can say that

$$a = x - l_3 \cos\theta$$
$$b = y - l_3 \sin\theta$$

Now, we will apply 2-link inverse kinematics to the l_1 and l_2 links.

$$c = \sqrt{a^2 + b^2}$$

We will find $\propto$ and β using "**cosine method**".

$$\propto = \cos^{-1}\frac{{l_1}^2 + c^2 - {l_2}^2}{2l_1c}$$
$$\beta = \cos^{-1}\frac{{l_1}^2 + {l_2}^2 - c^2}{2l_1l_2}$$

Now, from the diagram, angle θ_1 can be written as,

$$\theta_1 = \tan^{-1}\frac{b}{a} - \propto$$

Now, θ_2 can be written as,

$$\theta_2 = \pi - \beta$$

And also θ_3 can be written as,

$$\theta_3 = \theta - \theta_1 - \theta_2$$

➔ After obtaining the above equations, we did the coding part as shown in the code given above.

➔ We typed the above code in Arduino IDE and uploaded it to the microcontroller.

➔ In the code, the inverseKinematics() function takes three values as parameters, (x, y) coordinates and the angle of the 3rd link with the horizontal. It finds out the angle values for link1, link2, and link3 and then this function passes these angle values to **Braccio.ServoMovement(step delay, m1, m2, m3, m4, m5, m6);** system-call and moves the Robotic Arm in that position whose coordinates were (x, y).

➔ We used the in-built library named Braccio.h & Servo.h. It allows one to move each **Braccio** part using simple calls like **Braccio.ServoMovement(step delay, m1, m2, m3, m4, m5, m6)**; etc…

➔ During the execution of the code, we changed the values of the coordinate (x, y) and the angle theta and got the different values for theta1, theta2 and theta3 and also moved the Robotic Arm according to the angle values. In every execution, the Robotic Arm is first set to its initial position and then set to its given position.

➔ We have done a (+90) with theta2 and theta3 to keep them in the first quadrant even if in some cases values of the angles come negative.

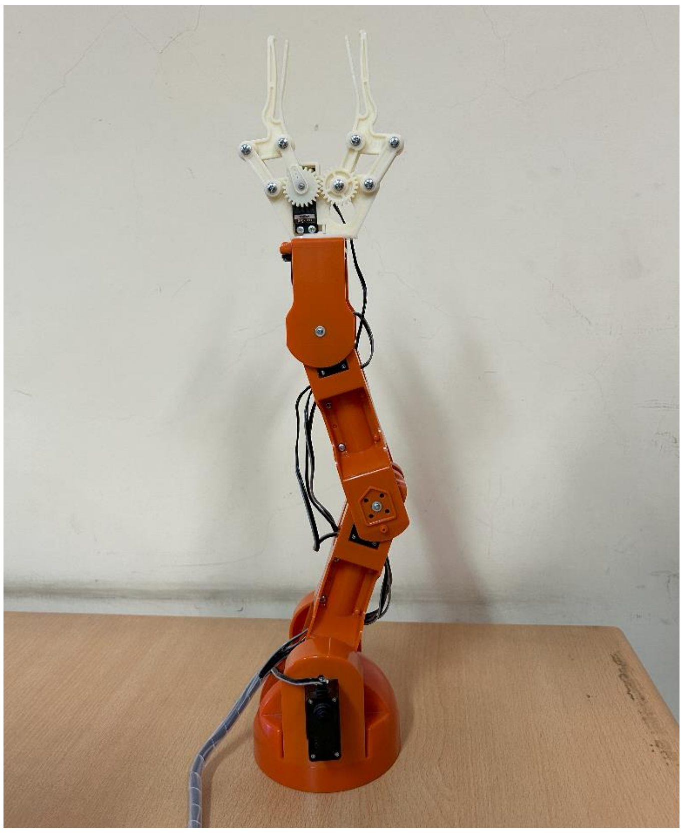
➔ After uploading the code, the Robotic Arm moved to the position that was given as coordinates (x, y) and the angle value of the 3rd link.

5) Observations:

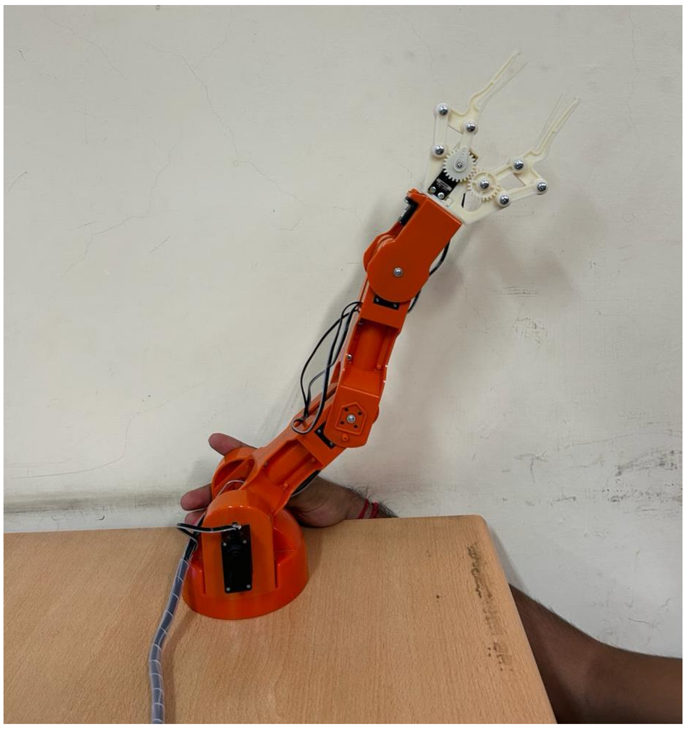
➔ YouTube Link of the Output Video: https://youtu.be/gsnxKZ3Plk0?si=-Y-jB_4tQrPGEVx4

➔ Output Images:

- Task 1: (x, y) = (0, 314), $\theta = 90°$



- Task 2: (x, y) = (157, 271.93), $\theta = 60°$



➔ More Tasks are in the YouTube video.

➔ We have used these values:

- (x, y) = (0, 314), $\theta = 90°$.
- (x, y) = (314, 0), $\theta = 0°$.
- (x, y) = (157, 271.93), $\theta = 60°$.
- (x, y) = (271.93, 157), $\theta = 30°$.
- (x, y) = (222, 222), $\theta = 45°$.

➔ After we uploaded the code, we ran it for different coordinate values also stated in the video to know what happens when we change the coordinate values and the value of θ. So, when we changed the values, the Robotic Arm configured in these positions shown in the video and images.

6) Result:

As a result, we successfully implemented the **Inverse Kinematics** in a 3-link Robotic Arm. The Robotic Arm is set to its initial position first. Then the custom function calculates the **Inverse Kinematics** and passes the appropriate angle values to the Braccio.ServoMovement() system call. Then it is set to the given position in terms of the end-effector position (x, y) and the angle of the 3rd link.

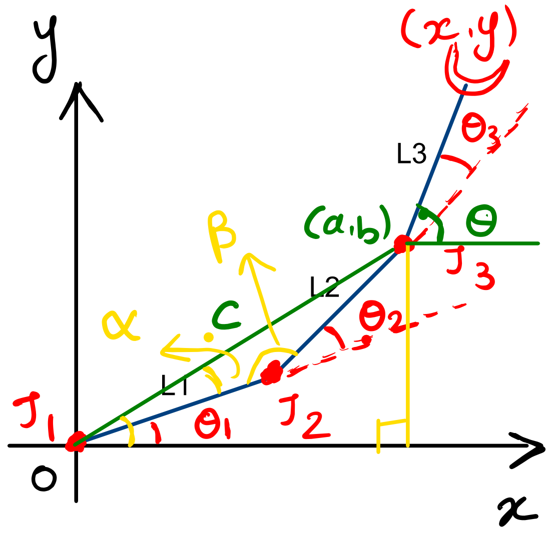
⇒ We are given with the co-ordinates of the end-effector (x,y) and an angle of the 3rd link 'θ'.

⇒ Now, with the help of (x,y) we will find the co-ordinates of the joint J3 (a,b).

⇒ From diagram, we can see that.

$$a = x - l_3 \cos\theta.$$

$$b = y - l_3 \sin\theta.$$



$\Rightarrow$ Now, we will apply 2-link inverse kinematics to the l_1 & l_2 links.

$$c = \sqrt{a^2+b^2}$$

$\Rightarrow$ Firstly, we will find α and β using "cosine method".

$$\therefore \alpha = \cos^{-1}\left(\frac{l_1^2 + c^2 - l_2^2}{2 l_1 c}\right).$$

$$\beta = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - c^2}{2 l_1 l_2}\right)$$

$\Rightarrow$ Now, from the diagram, angle θ_1 can be written as.

$$\boxed{\theta_1 = \tan^{-1}\left(\frac{b}{a}\right) - \alpha}$$

$\Rightarrow$ Now. θ_2 can be written as.

$$\boxed{\theta_2 = \pi - \beta}$$

$\Rightarrow$ And. θ_3 can be written as.

$$\boxed{\theta_3 = \theta - \theta_1 - \theta_2}$$