| **3-DOF Articulated Manipulator with Pick and Place Operation Based on Inverse Kinematics** |
| --- |

| A Project Report Submitted by |
| --- |
| **Shahriar Ferdous**  Roll No.: 1831002 |
| Completed in  **Special Studies and Project**  (MTE 3200) |

| Supervised by |
| --- |
| **Dr. Asief Javed**  Assistant Professor  Department of Mechatronics Engineering  Khulna University of Engineering & Technology (KUET)  Khulna -9203, Bangladesh. |
|  |
|  |
| 21 December 2022 |



| Department of Mechatronics Engineering |
| --- |
| Khulna University of Engineering & Technology  Khulna-9203 |

# Introduction

The robotic manipulator is a type of robot that is used for various industrial processes. The robot can move the position from one point to another by determining the position of the coordinates. This robot control configuration can be operated manually or automatically [1]. Inverse kinematics is an algorithm that is often used in the operation of a robot arm. This method allows the conversion of coordinates into the angular motion of each joint robot. In solving the inverse kinematic problem, several methods can be used such as Denavit-Hartenberg (D-H) [2], screw theory, and iterative method [3]. Besides, soft computing methods can also be used to solve inverse kinematic equations [4], [5]. In essence, some of these methods are used to find the best solution for calculating joint angles based on target coordinates and robot configuration.

Several researchers have discussed the application of inverse kinematics in the robot arm. One of them is presented in [6] which uses this method to control the position of the robot arm movement when carrying out a pick-and-place mission. Besides, other researchers also use this method for conveyor systems [7], sorting objects by shape [8], integration with computer vision [9], and rapid prototyping [10]. However, to facilitate the development of robot arms, especially in studying the kinematics of their movements, a small-scale prototype of a robot arm is needed. Several researchers have also developed a low-cost and small-scale prototype robot arm. Research [11] discussed the fabrication of light robot arm parts which was carried out for 6 to 10 hours. Each part of the robot is designed using 3D simulation.

The purpose of this project is to design a small-scale 3-DOF articulated manipulator for learning needs in the laboratory. The main contribution of this project is to provide a simple design for a robot arm that can be used to study the behavior and tasks of an articulated manipulator on a small scale. The robot arm was open source so Solidworks design can be found and then procured the aluminum body to avoid the hassle of 3D printing. The robot is designed using low-cost parts, such as MG servo motors and Arduino Uno microcontroller [12]. In this paper, the inverse kinematics method is applied to the 3-DOF articulated manipulator robot to accomplish and test an automatic pick-and-place mission.

# Objectives

1. To design and build a low-cost 3-DOF articulated manipulator.
2. To test and complete a pick and place operation.
3. To use the inverse kinematics to complete the pick and place mission.

# Methodology

## System Design

The overall system of this manipulator is simple. It is an Arduino-based 3-DOF manipulator that uses 4 metal servos along with a servo drive unit. The inverse kinematics mathematical algorithm is implemented in the Arduino IDE code where the user can define two points, the first one for pick operation and the second one for place operation. Every operation is well-defined and moduled in the code so that further operations can be programmed easily.

Table 1. Robot Specification

| **Specification** | **Type** |
| --- | --- |
| Microcontroller | Arduino Uno R3  ATMega 328P, operating voltage 5V,14 digital I/O pins, 6 analog input pins, clock speed 16 MHz |
| Servo Motor | MG996R  -Torque 9.4 Kgf.cm (4.8V)  -Torque 11 Kgf.cm (6V)  -Operating voltage 4.8-7.2V |
| Joint | 4 (base, shoulder, elbow, gripper) |
| Height (vertical position) | 35cm |
| Reachability | 25 cm, 0-180 degrees |

## Mathematical Modelling

Inverse kinematics is a method used to find corner variables for joint robots based on the robot's position coordinates. The inverse kinematics method will look for the parameter values that must be given to each actuator to achieve the final destination coordinates. The block diagram of inverse kinematics is shown in Fig. 1. In determining the final coordinates of the end-effector, inverse kinematics must be adjusted to the workspace boundary of the robot's reach. Solving the inverse kinematics can be done using Pythagoras's law and trigonometric rules. Inverse kinematics can be solved by looking at the two sides of the robot, namely the top view and the side view. The top side is used to find the degree angle θ1 of the base joint and the joint with the vertical rotating axis. The sides are used to find the degrees of angle θ2 of the joint shoulder and the degrees of angle θ3 of the joint elbow. For the degree of angle on the gripper, it is enough to use the ON-OFF algorithm. The illustration of the top and side diagram of the robot can be seen in Fig. 2 and Fig. 3.

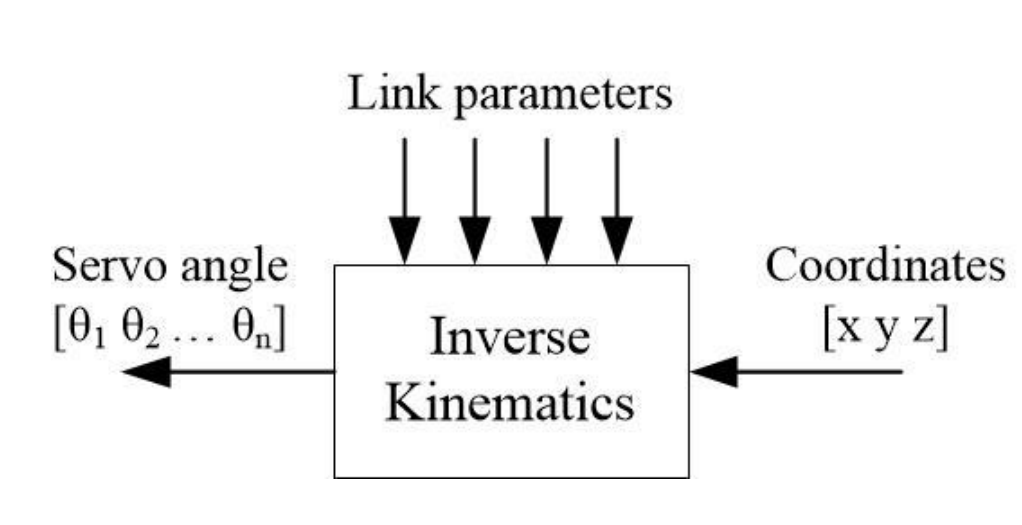


Fig. 1. Inverse kinematics mechanism

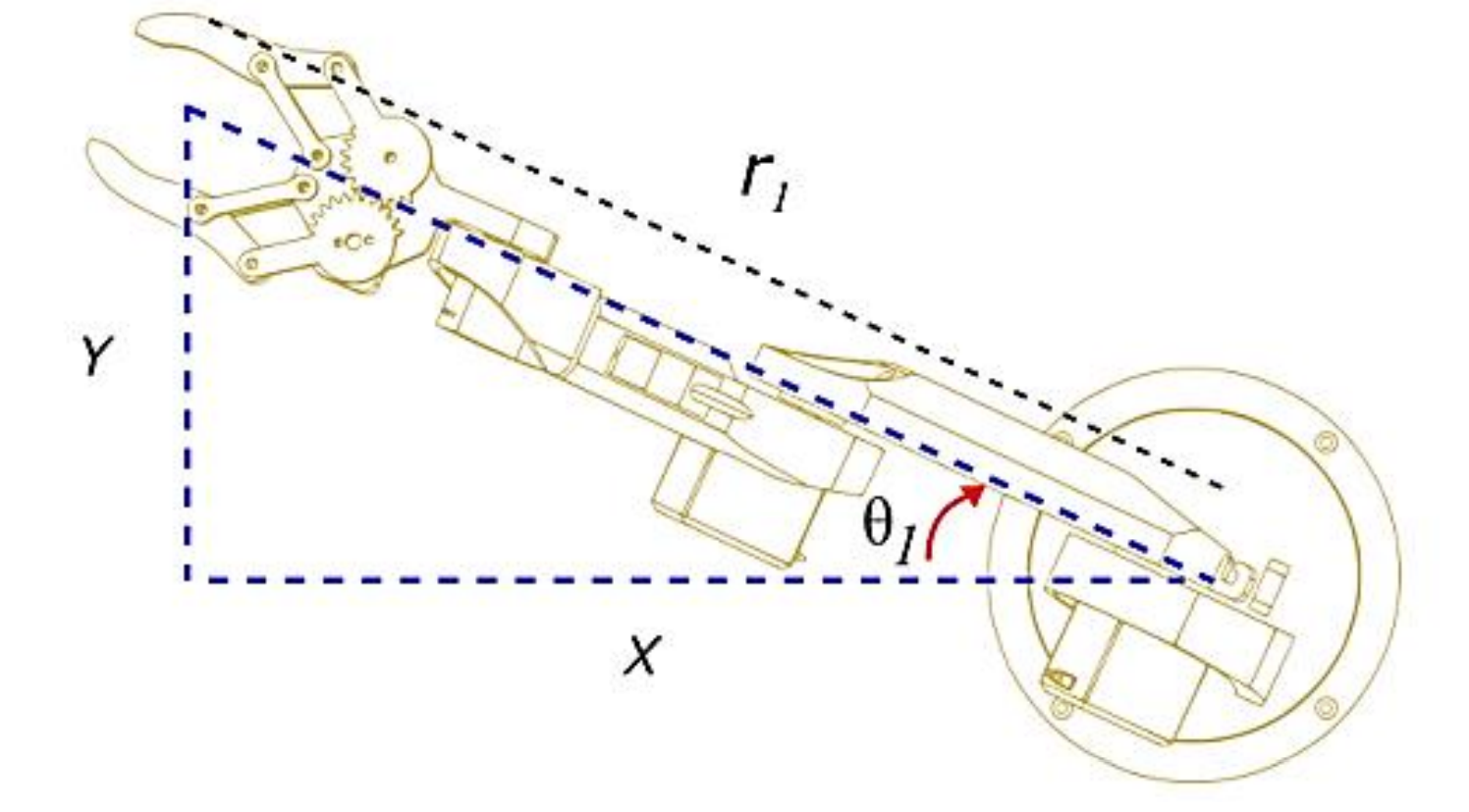


Fig. 2. Robot top view [13]

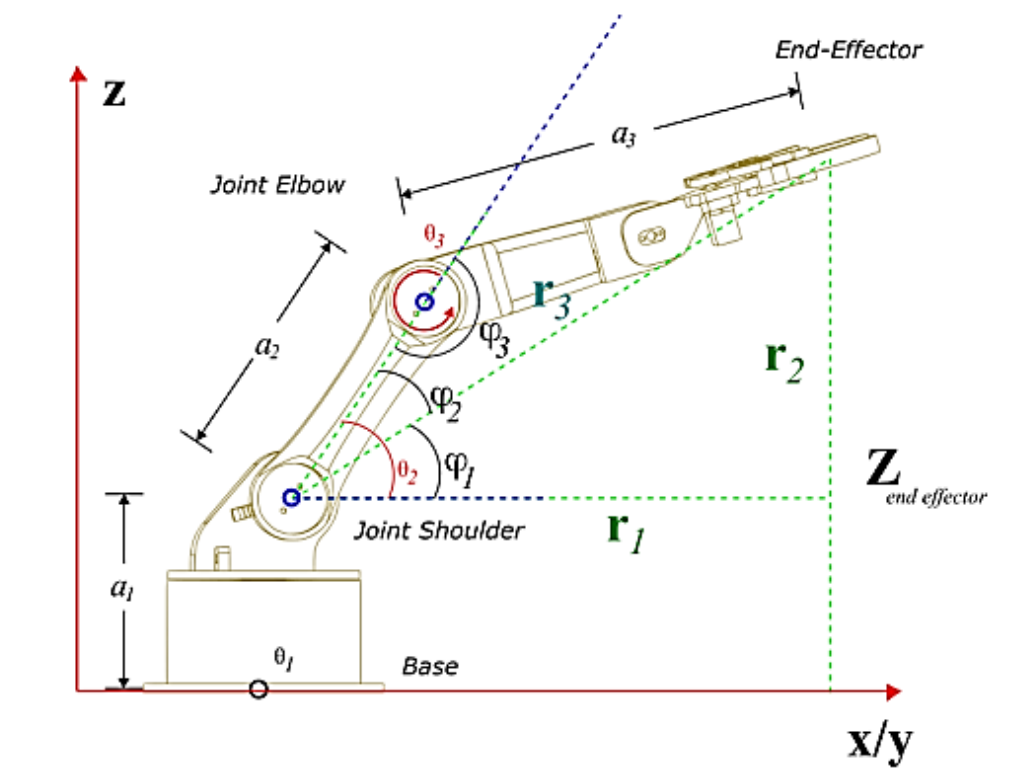
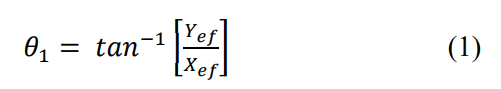


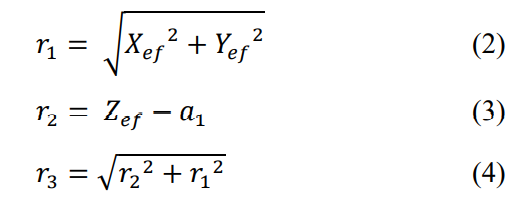
Fig. 3. Robot side view [13]

To determine the degree of the axis of the joint base, the following tangent equation can be used.

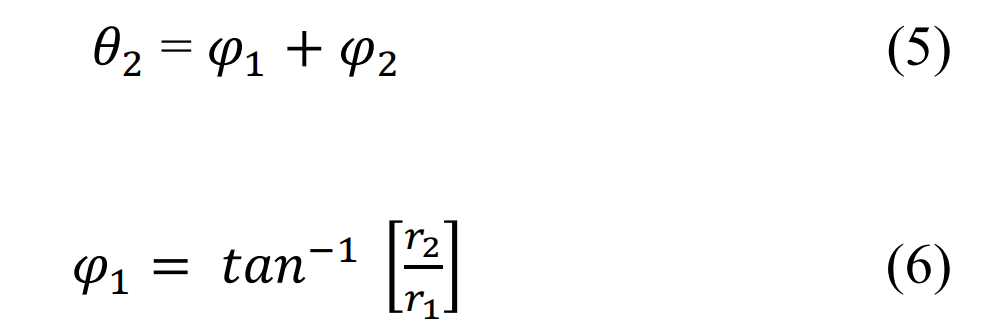


where 𝑋𝑒𝑓 is the position of the end-effector coordinates on the X-axis, 𝑌𝑒𝑓 is the position of the end-effector coordinates on the Y-axis, and 𝜃1 is the degree of the angle of the joint base.

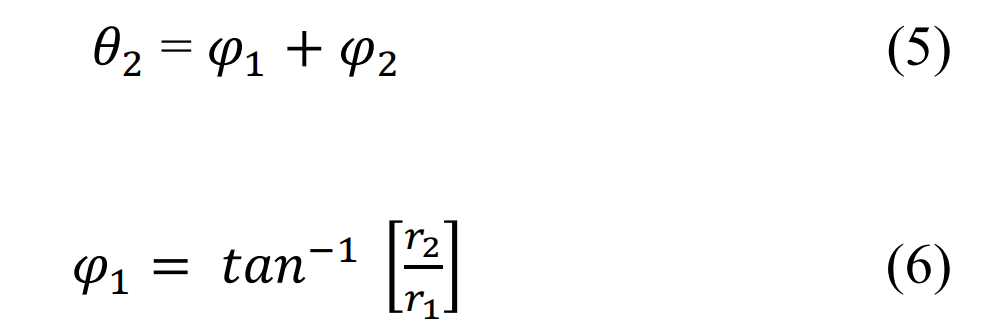
To find the degree of angle 𝜃2, a triangle guideline is needed which describes the lengths 𝑟1, 𝑟2, and 𝑟3 as shown on the dashed green line in Fig. 3. Thus, 𝑟1, 𝑟2, and 𝑟3 can be calculated based on the following equation.

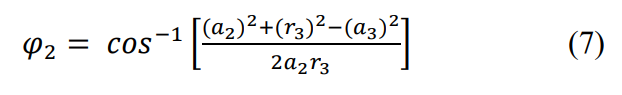


where 𝑍𝑒𝑓 is the position of the end-effector coordinates on the Z-axis and 𝑎1 is the link length between base and shoulder. Furthermore, by looking at the side of the 3-DOF robot arm structure, it can be found the value of the angle degrees 𝜃2 and 𝜃3. From the side, several variables can be defined including 𝑎1, 𝑎2, 𝑎3, 𝑍𝑒𝑓, 𝜃2, and 𝜃3. 𝑎2 is the link length between the shoulder and elbow. 𝑎3 is the link length between the elbow and the end-effector. 𝜃2 is formed from the horizontal line axis of the joint shoulder and the link between the shoulder and elbow. 𝜃3 is formed from the horizontal line axis of the joint shoulder and elbow. Referring to Fig. 3, 𝜃2 can be calculated by the following equation.

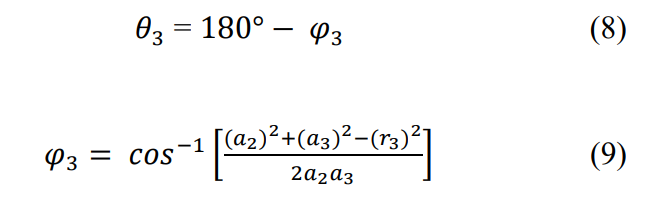


Where

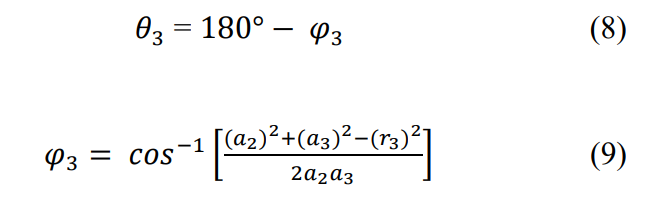




Meanwhile, 𝜃3 can be calculated using the following equation.



Where



## Circuit diagram

The circuit diagram of the system is depicted in Fig. 4. The circuit in the figure is an overall circuit that is being simulated using Tinkercad.

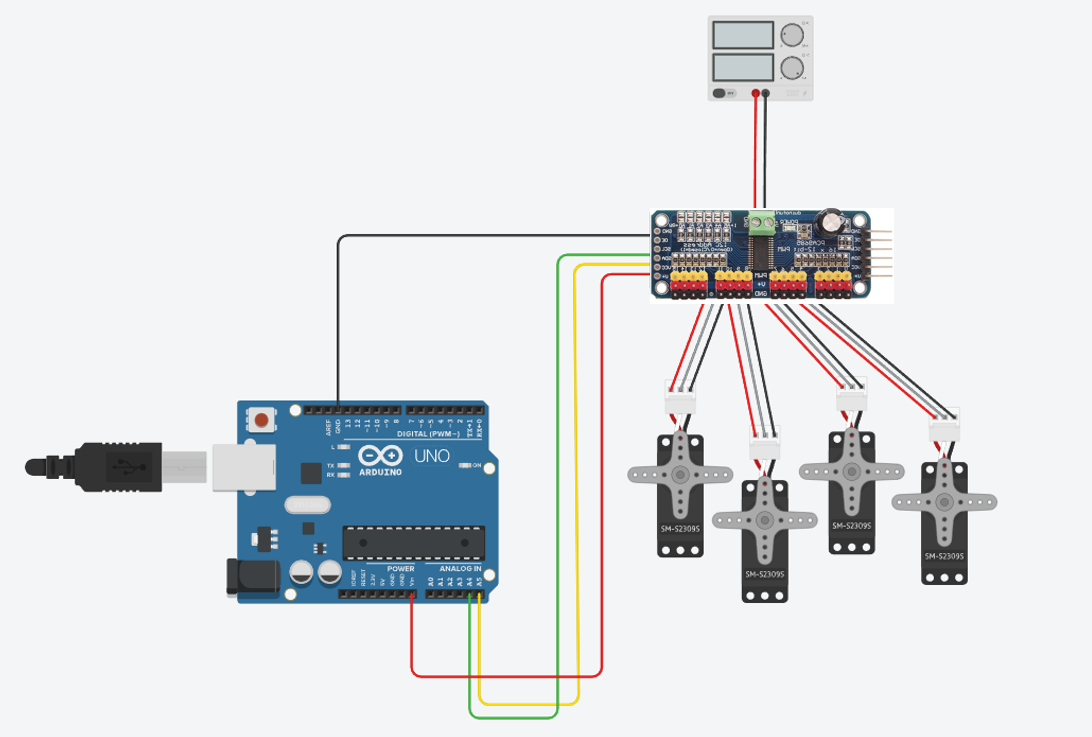
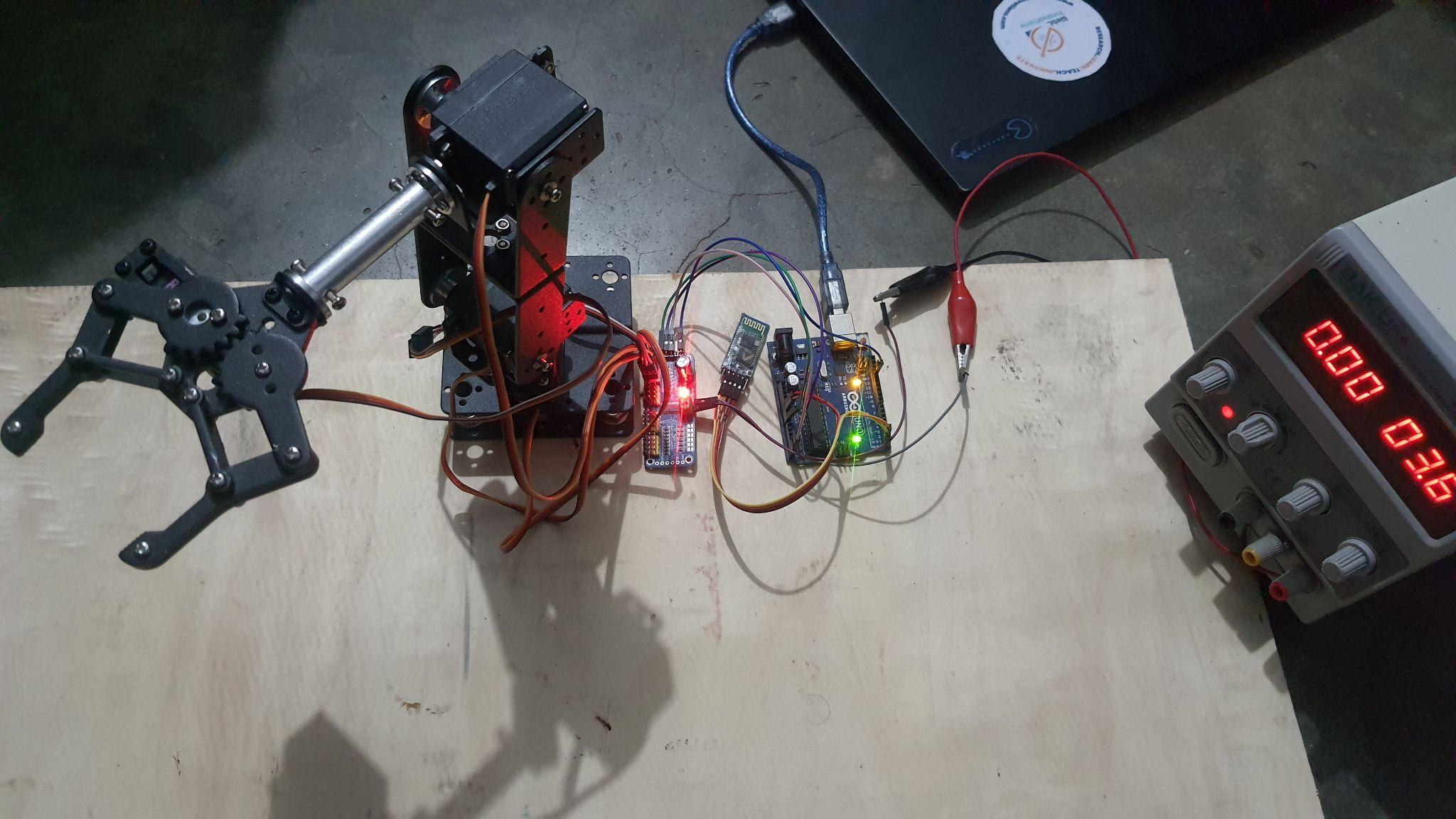
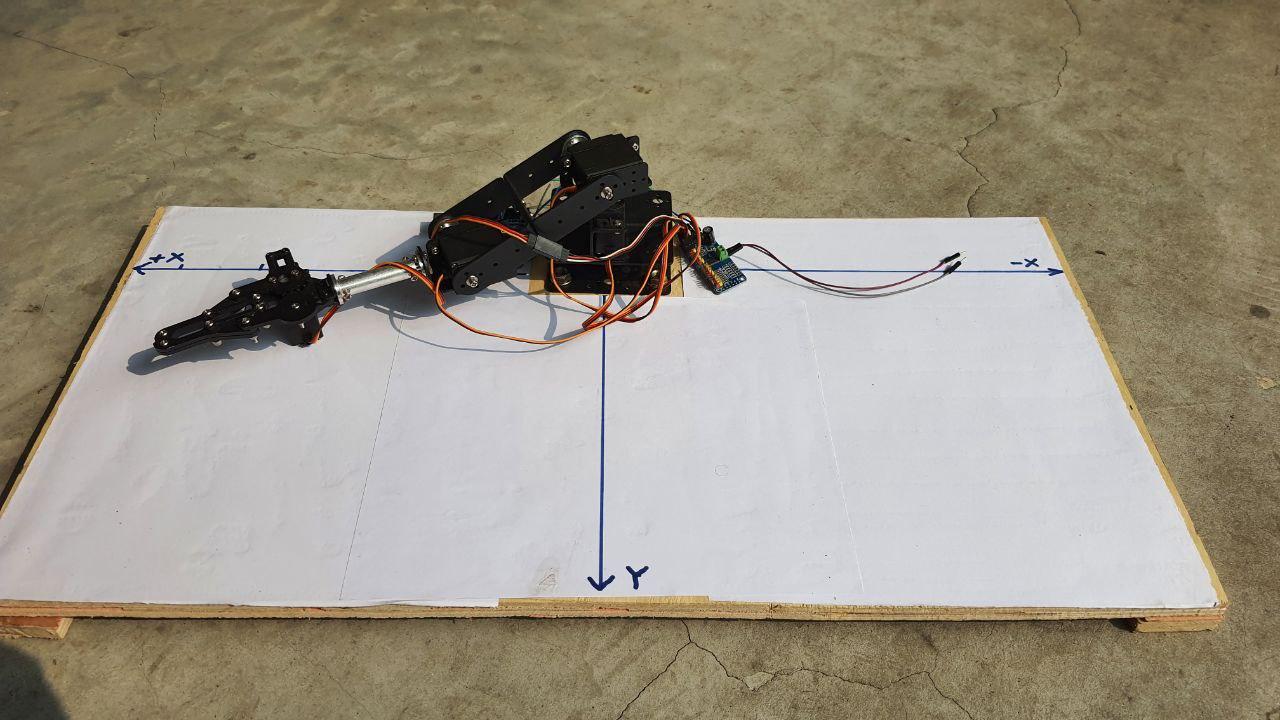
.

Fig 4: Overall circuit diagram

## Photograph of the project



(a)



(b)

Fig 5: Final Project (a & b)

# Result and Discussion

The overall system design is then implemented according to the system block diagram as shown in Fig. 5 (a & b). The system that has been created is then tested to determine its performance. Tests were carried out on servo motors and pick-and-place missions based on inverse kinematics.

## Pick and Place Coordinate Testing:

This test is carried out to determine the coordinates of the end-effector of the robot arm to reach the destination coordinates. The test is done by dividing the workspace into quadrant 1 for the negative end-effector X*ef* position and quadrant 2 for the positive end-effector X*ef* position. Measurements are made using a coordinate board and the aid of a ruler to find out the difference between the targets and the results achieved.

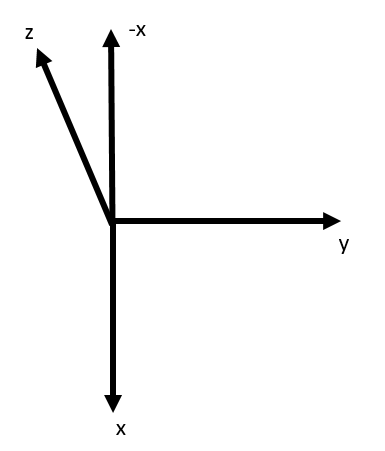


Fig 7: Coordinate Axis system

**Table 2. Cost of the components used in 3-DOF articulated manipulator**

| **SL** | **x-axis (cm)** | | **y-axis (cm)** | | **z-axis (cm)** | | **Error** | **Average**  **Accuracy** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Input** | **Output** | **Input** | **Output** | **Input** | **Output** | 97.01% |
| 2 | 10 | 9.7 | 10 | 10.15 | 10 | 9.4 | 2.83% |
| 3 | 0 | 0 | 25 | 25.7 | 8 | 7.6 | 2.6% |
| 4 | 9 | 8.8 | 15 | 15.5 | 12 | 11.5 | 3.24% |
| 5 | 12 | 11.5 | 20 | 20.6 | 15 | 14.6 | 3.28% |

The result of the coordinate testing depicts that there is less than 0.7 cm error in the x and y-axis. But the error of the z-axis is greater than 0.6 cm because of the weight of the robot's shoulder, elbow, and grippe. Also from the testing, it can be seen that the lowest point on the y-axis is 15cm and the highest point is 25 cm as per the defined workspace. In the case of the X-axis, the robot end effector can reach 25 cm in both directions which means -25 cm to 25 cm. Also in the case of the Z-axis, there is no minus value and it can reach up to 25 cm as well. So the error rate is more or less 2 cm on the X-axis and a maximum of 3 cm on the Z-axis. In table 2, the error has been calculated by averaging the error for the x, y, and z-axis for any particular set of values. Then for 5 sets of values, average accuracy has been calculated using the value of average error. The calculated average accuracy is 97.01% which is acceptable for this project and can be improved by precise motor and link calibration. Also controller like PID can be used which will increase the accuricy.

# Cost Table

Table 3. Cost of the components used in 3-DOF articulated manipulator

| **SL no.** | **Name of the components** | **Quantity** | **Unit price**  **(BDT)** | **Total price**  **(BDT)** |
| --- | --- | --- | --- | --- |
| 1 | 3-DOF Manipulator Frame | 1 | 2500 | 2500 |
| 2 | Arduino Uno R3 | 1 | 1100 | 1100 |
| 3 | Servo Motor MG 996 | 4 | 450 | 1800 |
| 4 | 16 Channel 12-bit PWM/Servo Driver | 1 | 530 | 530 |
| 5 | External Frame | 1 | 400 | 400 |
| Total cost: Six Thousand Seven Hundred Twenty Taka Only | | | | 6,330 |

# Application

The main application of this small-scale 3-DOF articulated manipulator project is to fulfill the learning needs in the laboratory. Due to its small scale and simple design, it can be easily re-fabricated with a low budget which could be very useful for teaching robotics in educational institutions.

# Conclusion

A small-scale 3-DOF manipulator robot arm with a pick-and-place mission based on inverse kinematics has been designed, implemented, and tested successfully. The inverse kinematics method that is applied can make the robot perform a pick-and-place mission for several coordinate targets. The system takes coordinates of x, y, and z as input and moves the joints automatically to reach the end effector to the target coordinate. Then the jaw of the end effector opens and picks an object. After that, the end effector moves to the place coordinates to finally place the object. The pick and place coordinates can be given using Arduino code.

In the further development phase, a more easy-to-use user interface (UI) will be introduced from where multiple waypoints can be provided by the user. Also, the accuracy will be increased by building the manipulator frame with robust 3D printed materials which will reduce the error due to the weight. The robot can be more accurate and precise with complex operations like pick and place with more DoF. So, in further development, the manipulator will be a 4-DoF manipulator with a user-friendly UI and learning features.

# Reference:

[1] R. Gautam, A. Gedam, A. Zade, and A. Mahawadiwar, “Review on development of industrial robotic arm,” International Research Journal of Engineering and Technology (IRJET), vol. 4, no. 03, 2017.

[2] Y. Becerra, M. Arbulu, S. Soto, and F. Martinez, “A comparison among the Denavit-Hartenberg, the screw theory, and the iterative methods to solve inverse kinematics for assistant robot arm,” In International Conference on Swarm Intelligence, Springer, Cham, 2019, pp. 447-457.

[3] A. Patil, M. Kulkarni, and A. Aswale, “Analysis of the inverse kinematics for 5 DOF robot arm using DH parameters,” In 2017 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2017, pp. 688-693.

[4] C. Lopez-Franco, J. Hernandez-Barragan, A. Y. Alanis, and N. Arana-Daniel, “A soft computing approach for inverse kinematics of robot manipulators,” Engineering Applications of Artificial Intelligence, vol. 74, pp. 104-120, 2018.

[5] G. B. Mahanta, B. B. V. L. Deepak, M. Dileep, B. B. Biswal, and S. K. Pattanayak, “Prediction of inverse kinematics for a 6-DOF industrial robot arm using soft computing techniques,” In Soft computing for problem solving, Singapore, 2019, pp. 519-530.

[6] K. M. Myint, Z. M. M. Htun, and H. M. Tun, “Position control method for pick and place robot arm for object sorting system,” International journal of scientific & technology research, vol. 5, no. 06, pp. 57-61, 2016.

[7] S. D. Han, S. W. Feng, and J. Yu, “Toward Fast and Optimal Robotic Pick-and-Place on a Moving Conveyor,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 446-453, 2019.

[8] H. S. Nu'man, Y. Sofyan, and A. R. Al Tahtawi, “Pengendalian Robot Lengan Pemilah Benda Berdasarkan Bentuk Menggunakan Teknologi Computer Vision. In SEMNASTERA (Seminar Nasional Teknologi dan Riset Terapan), vol. 2, 2020, pp. 42-48.

[9] N. F. Zulfardi, D. I. Saputra, A. D. A. Ahkam, “Aplikasi Deteksi Benda Menggunakan Metode Image Substraction Sebagai Masukan Koordinat Pada Robot Lengan 3-DOF,” In Prosiding Seminar Nasional Teknologi dan Riset Terapan (SEMNASTERA), vol. 1, 2019, pp. 30-37.

[10] I. Ishak, J. Fisher, and P. Larochelle, “Robot arm platform for rapid prototyping: Concept,” In Proceedings in Florida Conference on Recent Advances in Robotics, 2015.

[11] S. C. Gutiérrez, R. Zotovic, M. D. Navarro, and M. D. Meseguer, “Design and manufacturing of a prototype of a lightweight robot arm,” Procedia Manufacturing, vol. 13, pp. 283-290, 2017.

[12] K. Kunal, A. Z. Arfianto, J. E. Poetro, F. Waseel, and R. A. Atmoko, “Accelerometer Implementation as Feedback on 5 Degree of Freedom Arm Robot,” Journal of Robotics and Control (JRC), vol. 1, no. 1, pp. 31-34, 2020.

[13] A. R. A. Tahtawi, M. Agni and T. D. Hendrawati, “Small-scaleRobot Arm Design with Pick and Place Mission Based on InverseKinematics,” Journal of Robotics and Control, vol. 2, pp. 469–475,2021, https://doi.org/10.18196/jrc.26124

# Code:

The inverse kinematics equation is implemented using programming language embedded in the Arduino board microcontroller. Below shows a list of inverse kinematics programs in C language on the Arduino IDE compiler.

#include <Wire.h>

#include <Adafruit\_PWMServoDriver.h>

Adafruit\_PWMServoDriver myServo = Adafruit\_PWMServoDriver();

#define SERVOMIN 150

#define SERVOMAX 600

uint8\_t servonum = 0;

uint8\_t numberOfServos = 4;

float a1 = 6.1;

float a2 = 10.5;

float a3 = 19.0;

float currentPos0 = 375;

float currentPos1 = 375;

float currentPos2 = 375;

float theta1, theta2, theta3, minusx, r1, r2, r3, pi1, pi2, pi3, thet1, thet2, thet3;

void setup() {

//Serial.begin(9600);

myServo.begin();

myServo.setPWMFreq(60);

//delay(10);

}

void loop() {

float x1 = 0.0;

float y1 = 10.0;

float z1 = 10.0;

float x2 = 0.0;

float y2 = 25.0;

float z2 = 10.0;

currentPos0 = 375;

currentPos1 = 375;

currentPos2 = 375;

myServo.setPWM(0, 0, currentPos0);

myServo.setPWM(1, 0, currentPos1);

myServo.setPWM(2, 0, currentPos2);

myServo.setPWM(3, 0, 430);

delay(3000);

inverse(x1, y1, z1, currentPos0, currentPos1, currentPos2);

delay (3000);

myServo.setPWM(3, 0, 250);

delay (3000);

myServo.setPWM(3, 0, 450);

delay(1000);

inverse(x2, y2, z2, currentPos0, currentPos1, currentPos2);

delay (3000);

myServo.setPWM(3, 0, 250);

delay(3000);

myServo.setPWM(3, 0, 430);

while(1){

}

}

//Inverse Kinematics Function

float inverse(float x, float y, float z, float cPos1, float cPos2, float cPos3){

//calculate theta1

if (x>-1 ) {

theta1 = atan(y/x);

theta1 = theta1\*180/PI;

}

else {

minusx = -x;

theta1 = atan(y/minusx);

theta1 = theta1\*180/PI;

theta1 = 180-theta1;

}

//calculate r1,r2,r3

r1 = sqrt((sq(x))+(sq(y)));

r2 = z-a1;

r3 = sqrt((sq(r2))+(sq(r1)));

//calculate pi1,pi2,pi3

pi1 = atan(r2/r1)\*180/PI;

pi2 = acos(((sq(a2))+(sq(r3))-(sq(a3)))

/(2\*a2\*r3))\*180/PI;

pi3 = acos(((sq(a2))+(sq(a3))-(sq(r3)))

/ (2\*a2\*a3))\*180/PI;

//calculate theta2,theta3,theta4

theta2 = pi1+pi2;

theta3 = 180-pi3;

thet1 = (2.5\*theta1)+150;

thet2 = (2.5\*theta2)+150;

thet3 = (2.5\*theta3)+150;

// For Joint 1

if (thet1 >= cPos1){

for (int i = cPos1; i <= thet1; i++){

myServo.setPWM(0, 0, i);

delay(10);

}

}

else {

for (int i = cPos1; i >= thet1; i--){

myServo.setPWM(0, 0, i);

delay(10);

}

}

// For Joint 2

if (thet2 >= cPos2){

for (int i = cPos2; i <= thet2; i++){

myServo.setPWM(1, 0, i);

delay(10);

}

}

else {

for (int i = cPos2; i >= thet2; i--){

myServo.setPWM(1, 0, i);

delay(10);

}

}

// For joint 3

if (thet3 >= cPos3){

for (int i = cPos3; i <= thet3; i++){

myServo.setPWM(2, 0, i);

delay(10);

}

}

else {

for (int i = cPos3; i >= thet3; i--){

myServo.setPWM(2, 0, i);

delay(10);

}

}

currentPos0 = thet1;

currentPos1 = thet2;

currentPos2 = thet3;

}