International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 3, pp 1-9, March-2023

https://doi.org/10.47001/IRJIET/2023.703001

Prototype of 3-DOF Pick and Place Arm Robot

[1]Moayed Almobaied, [2]Ahmed Abu Metair, [3]Bashar M. Thabet

[1,2,3]Electrical Engineering Department, Islamic University of Gaza, Palestine

***Abstract* - What we are witnessing today in terms of acceleration in manufacturing and production processes in factories would not have taken place without relying on robotic arms to accomplish routine work with better accuracy, fewer errors and a higher speed compared to the human element and because conducting experiments and modifying ready-made robots that operate on production lines and testing new software on them is expensive and not available in many cases, the idea stemmed from here in creating a prototype for a robotic arm that enables developers and researchers to test and develop their own software before adopting it, and this robot was characterized by ease of use and low cost compared to what is in the market with the possibility of controlling it in more than one way, as this research shows the method of designing and manufacturing a robotic arm of the type 3- DOF (3-Revolute joints) used to capture Various parts. Electromagnets of end effect or were used to pick up the pieces. This robotic arm was tested in an integrated environment using the MATLAB - Simulink program under different conditions, the first of which depends on the forward kinematics and the second mode of operation depends on the inverse kinematics, while the third mode was using the two types of Trajectory Planning (LSPB – Conic), Practical experiments of this robotic arm have shown high accuracy in performance, quick response in executing orders, and avoiding falling into Singularities due to setting mechanical and software restrictions for the robot.**

Keywords: 3-DOF arm robot; kinematic design; manipulator implementation; Simulink.

I. INTRODUCTION

Robotics technology has seen significant advancements and applications in recent years, with robotic arms being one of the most popular applications in industrial settings. The design and assembly of a robotic arm is a complex process that requires expertise in various fields such as mechanical, electrical, computer science.

George Charles Devol first described industrial robots in 1954. According to Devol, an industrial robot is a mechanical device with a number of degrees of freedom (DOF) that can be automatically controlled, programmed to carry out a variety of tasks, and capable of using an arm with various tooling [1], Since the robotic arm may help the industry create high production rates compared to the output production from human labor, many industrial organizations are utilizing it to boost productivity [2]. A pick-and-place robotic arm and simulation for industrial application have been created through numerous investigations. An industrial 6-DOF robotic arm was created in a study to pick up goods moving along a conveyor belt and then drop them with the help of vision sensors into a designated container[3], According to a report, a 3-DOF manipulator robot with servo motors linked to its joints is controlled by an Arduino Mega 2560 [4]. To enable faster computing and processing, their algorithm for controlling the robot needs to be modified. Another study attached motors to each joint of their robotic arm and used the Arduino Mega 2560 as the controller to enable communication between a computer and the robotic arm [2]. By adjusting the motor's power width modulation (PWM) settings, each motor's rotation can be managed. But the technology requires the user to manually set the desired angle of the robotic arm. In other study [5]. To control the rotation of the motors attached to the real 3-DOF robotic arm, they used the rotation (resistance) values of each potentiometer attached to each joint of a small 3-DOF robotic arm. Potentiometer and motor calibration must be done in-depth for this. Based on the previous work, most of them produced an articulated robotic arm using servo or DC motors as the robot's actuator. They used the simple-to-use Arduino Mega 2560 microcontroller for the experimentation.

Previous researchers have faced difficulties related to reducing the cost of manufacturing an easy-to-use robotic arm that is available to everyone within an integrated work environment, which is what distinguishes this current design as is evident in this paper with the possibility of controlling it in more than one mode, forward kinematics mode and inverse kinematics mode and Trajectory Planning mode (LSPB or Conic) The objective of this paper is to designing and implementation.

This robot arm is to assist in the process of picking up pieces and placing them in the designated place on the electronic board on the production line in a factory for assembling electronic boards.

The main contribution in This paper is the design and assembly of a 3-DOF robotic arm using MATLAB's Simulink software, Arduino's servo library, the Robotics Toolbox for handling kinematics, soled work software and use 3-d printer

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 3, pp 1-9, March-2023

https://doi.org/10.47001/IRJIET/2023.703001

to build the arm of robot and the final shape of the links of robot which is shown in figure (1). as low-cost platform that is constantly being improved and provides engineers with hands on experience.

This paper is organized as follows. Section 2 gives the robot description and kinematics. The design and implementation are presented in Section 3. The discussion of the results obtained are described in Section 4. Finally, the conclusion is given in Section 5.

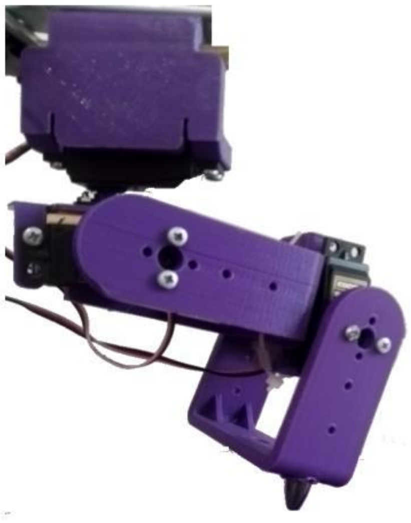

Figure 1: Manipulator Arm

1.1 Elbow arm Model

The elbow arm has many advantages that make it an appealing and popular design, as is clear in the structure in figure (2) and in the workspace. The articulated arm, It enables a comparatively high degree of freedom of movement in a small space and is known as a revolute manipulator; the acronym (RRR) stands for (Revolute - Revolute -Revolute). [6]. The arm's design must start at the base and conclude at end effecter or gripper since each link depends on the link before it. As a result, the design process starts with the trunk or base, then moves on to the shoulder, and so forth [7].

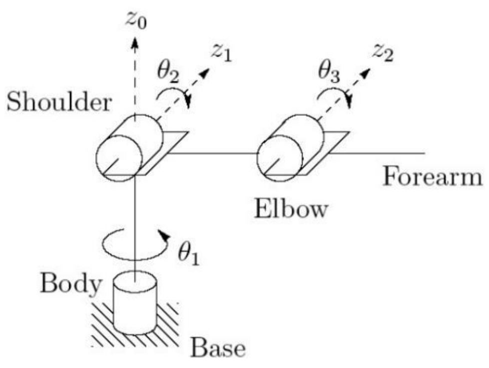

Figure 2: Structure of the Elbow Manipulator

1.2 Forward kinematics

Forward kinematics in robotics refers to the process of finding the end-effector position and orientation relative to the base frame, given the joint angles of a manipulator. The D-H (Denavit - Hartenberg) convention is a method for describing the relative orientation of adjacent coordinate frames in a manipulator using four parameters (ai, di, αi, θi). The forward kinematics problem is solved by computing the homogeneous transformation matrices for each joint, and then composing these matrices to find the transformation from the base frame to the end-effector frame. This transformation represents the position and orientation of the end-effector in the base frame.

To deriving the forward kinematics of a manipulator using the D-H convention.

The steps are [8][9]:

1) Label the joint axes z0 to zn−1.
2) Establish the base frame and origin on z0-axis.
3) Locate the origin Oi for each joint i (1 to n-1).
4) Establish Xi and Yi to form a right-hand frame for each joint i.
5) Establish the end-effector frame (XnYnZn) and its origin.
6) Create a table of link parameters (ai, di, αi, θi).
7) Form the homogeneous transformation matrices (Ai) using the parameters.
8) Compute T 0n = $A_1 ... A_n$ to find the position and orientation of the tool frame in the base coordinates.

1.3 Inverse kinematics

Inverse kinematics in robotics refers to the problem of finding the joint angles of a manipulator that result in a desired end effector position and orientation. It is the reverse process of forward kinematics, where the end-effector position and orientation are known and the joint angles are to be computed.

Inverse kinematics is typically a more complex problem than forward kinematics, as it involves finding a solution to a set of nonlinear equations. There are several methods for solving the inverse kinematics problem, including analytically-based methods, numerical methods, and machine learning-based methods. The choice of method depends on the specific manipulator and the requirements of the application. The inverse kinematic triangulation method is a way to solve the inverse kinematics problem, which is to find the joint angles of a manipulator that result in a desired end-effector position and orientation [10][11]. The steps for this method are:

ISSN (online): 2581-3048
Volume 7, Issue 3, pp 1-9, March-2023
https://doi.org/10.47001/IRJIET/2023.703001

1) Define a target position and orientation for the end effector.
2) Define the location and orientation of the end-effector in the base frame.
3) Using the forward kinematics equation, compute the position of each joint in the base frame.
4) Using the position of the joints and the target end-effector position, solve a set of triangles to find the length of each link.
5) Check if the solution is possible by verifying that the computed link lengths are within the physical constraints of the manipulator.
6) If the solution is possible, use the computed link lengths and the forward kinematics equation to find the joint angles that result in the desired end-effector position and orientation. This method assumes that the manipulator is a tree-structured serial link manipulator and that there is a unique solution for the joint angles.

1.4 Jacobian

The Jacobian is a matrix that relates the velocity of the end- effector to the joint velocities in a manipulator. It is used to determine how changes in the joint angles affect the velocity of the end-effector, and is a key tool in the analysis and control of manipulators.

The steps to find the Jacobian of a manipulator are as Follows [12]:

1) Define the end-effector position and orientation in terms of the joint angles, using forward kinematics.
2) Express the velocity of the end-effector in terms of the joint velocities.
3) Compute the partial derivatives of the end-effector velocity with respect to the joint angles.
4) Assemble the partial derivatives into the Jacobian matrix, which has the same number of rows as the velocity of the end effector and the same number of columns as the joint velocities.
5) The Jacobian can be used to determine the joint velocities that result in a desired end-effector velocity, by solving a linear equation. The structure of the Jacobian can vary depending on the specific manipulator and the choice of reference frame, and that the Jacobian is a function of the joint angles, which can change over time.

1.5 Singularity

Singularities in a manipulator occur when the Jacobian matrix becomes singular or rank-deficient, leading to an indeterminacy in the relationship between the joint velocities and end-effector velocity.

The following are steps to identify and analyze singularities in a manipulator [13][14]:

1) Compute the Jacobian matrix at various configurations of the manipulator.
2) Determine the rank of the Jacobian matrix at each configuration. If the rank is less than the number of joints, the manipulator is in a singular configuration.
3) Identify the singular configurations where the rank of the Jacobian is less than the number of joints. These configurations are known as singularities.
4) Analyze the behavior of the manipulator near the singularities. In some cases, small changes in the joint angles can result in large changes in the end-effector velocity, leading to instability and poor performance.
5) Develop strategies for avoiding or mitigating the effects of singularities in the control of the manipulator, such as using singularity-free control techniques or modifying the design of the manipulator to avoid singular configurations. the presence of singularities can have significant impacts on the performance and stability of the manipulator, and it is important to consider these effects when designing and controlling manipulators.

1.6 Trajectory

Trajectory planning for a manipulator involves defining a path for the end-effector to follow in task space, and then computing the joint angles that result in the desired end effector motion [15][16].

The steps to perform trajectory planning for a manipulator are as follows:

1) Define the initial and final positions and orientations of the end-effector.
2) Choose a reference frame and coordinate system in which to express the end-effector motion.
3) Define the trajectory path in terms of position, velocity, and acceleration profiles. This can be done using a variety of methods, such as cubic splines, quintic polynomials, or differential flatness.
4) Compute the inverse kinematics for each point along the trajectory to find the corresponding joint angles.
5) Optimize the joint angles to ensure smooth motion and minimize joint accelerations, torques, and other performance criteria.
6) Check for feasibility and collision avoidance, and modify the trajectory if necessary.
7) Repeat steps 4-6 as necessary to refine the trajectory.
8) Send the joint angles to the manipulator to execute the trajectory.

ISSN (online): 2581-3048
Volume 7, Issue 3, pp 1-9, March-2023
https://doi.org/10.47001/IRJIET/2023.703001

The computational complexity of trajectory planning can be high, especially for high-dimensional manipulators and complex tasks.

II. DESIGN AND IMPLEMENTATION

This paper talks about each of these areas separately.

2.1 Hardware Environment

According to figure (3), hardware environment for our project comprises of an Arduino microcontroller [17], a PC, and a data link that connects PC to the arm. Microcontroller is a component that performs embedded computing and interfaces with robot actuators. Simulink is used on the PC to program the user-defined embedded software that will run on the microcontroller, which is needed to control the robot. When developing prototypes for microcontroller-based goods and procedures, it also functions as a debugging environment. It enables the user to get particular info. For the microcontroller and PC to communicate, a data link is required. In this project, the microcontroller and PC are connected through a serial communication link (wire).

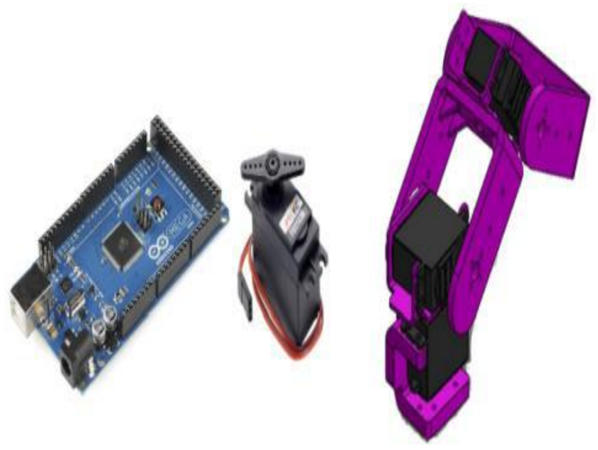

Figure 3: Hardware Environment

The components are as follows:

1) **Servo Motors:** A servo motor is a rotary actuator used in applications that demand a high level of rotation angle precision [18]. This is because servo motors include feedback encoders that monitor the variation between the target and desired angles and guarantee a more accurate final position. Two issues arise when utilizing a typical DC motor (angle and speed controlling). If the motor is pushed to maximum voltage in order to produce large torques, the motor's rotation speed will be too high to accurately adjust the angle of the rotation. While a low voltage will result in very little torque and the motor won't be able to move the necessary load, a low speed is necessary to accurately control the rotation angle. The lack of a feedback mechanism when employing stepper motors might result in a lag caused by inertia and a definite loss of accuracy when reaching the final position.[19] In light of this, servo motors were selected as the ideal actuator for this operation.
2) **Arduino Board:** A board with the microcontroller ATmega2560 is called the Arduino Mega 2560. It contains 16 analog inputs, 4 hardware serial ports (UARTs), a 16 MHz crystal oscillator, 54 digital input/output pins (of which 15 may be used as PWM outputs), a USB connector, a reset button, an ICSP header, and a power jack. Comes with all necessary to support the microcontroller; to get started, just plug in a USB cable, an AC-to-DC adapter, or a battery. The majority of shields made for the Uno and earlier boards like the Duemilanove or Diecimila may be used with the Mega2560 board. servo motors can be managed by an Arduino board thanks to a library. Servos feature precisely controllable shafts and integrated gears. The shaft may be positioned at a variety of angles with standard servos, typically from 0 to 180 degrees. The shaft may rotate at different rates thanks to continuous rotation servos [18].

2.2 Software Environment

The Arduino program and the MATLAB program are the two main components of the software environment. Arduino program was created to create the interface between the PC and the arm. To calculate forward-inverse kinematics, Jacobians, and singularities, the MATLAB software uses the Serial Communication code and the m. file code [20]. Figure (4) illustrates the Simulink GUI for end user to control the arm robot.

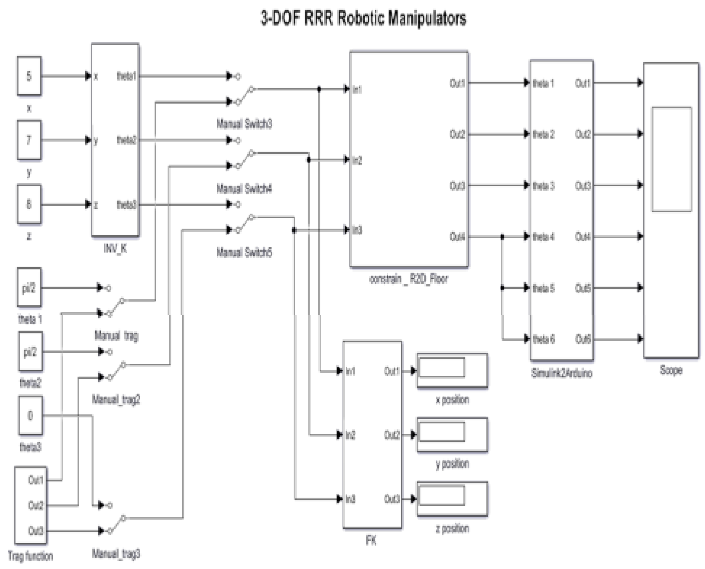

Figure 4: Simulink

ISSN (online): 2581-3048
Volume 7, Issue 3, pp 1-9, March-2023
https://doi.org/10.47001/IRJIET/2023.703001

2.3 Mechanical Part

The robot includes 5 Mechanical Part.

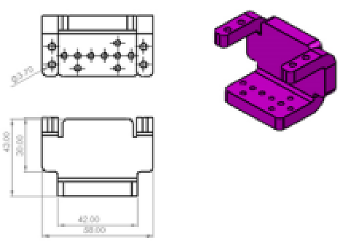

Figure 5: Servo 1 Base Holder

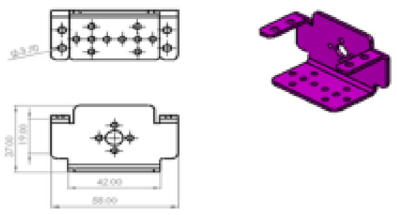

Figure 6: Servo 2 Base Holder

1) Base Servo Holder: Have 3 base holder. Every base has different measurements and weight depend on the place and the load on the part. second base holder connect the first servo with the second servo that move the first link in the robot the final base holds the servo which makes θ3. And connect the first link with the second one. We make it bigger to avoid the touching between the first and the second link. The details shown in figure (5,6,7).

2) Robot Links: Have 2 links. The first link which moves by the second servo to achieve θ2 and carry the third servo. The final link in our robot which moves by the third servo to make θ3 and moves the end effector. The details shown in figure (8,9).

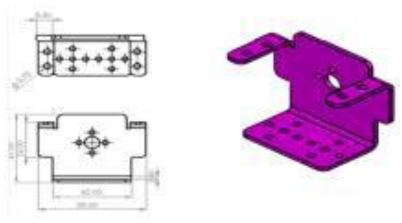

Figure 7: Servo 3 Base Holder

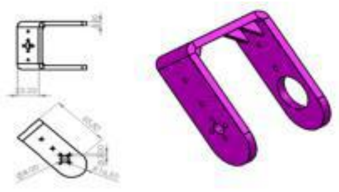

Figure 8: First link

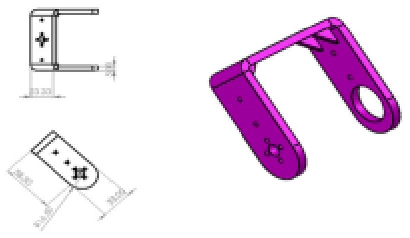

Figure 9: Second link

SolidWorks software was used as 3d Cad program to draw and assemble the robot before starting with the practical manufacturing. After making the needed modification on the measurements and the shape of the robot, the final shape of the links of robot which is shown in figure (10).

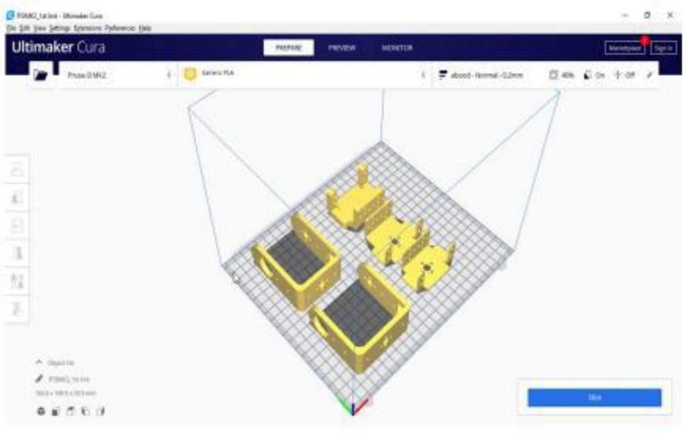

Figure 10: Cura program

III. RESULTS AND DISCUSSIONS

The suggested robotic arm's forward and inverse kinematics is derived in this subsection. The required end effector position in the task space is translated into the desired joint angles in the joint space using the robot's kinematic analysis [21].

3.1 Robotic Arm Forward Kinematics

Obtaining a homogeneous transformation matrix of robot arm using Danevit-Hartenberg approach [22]. TABLE I shows the Denavit-Hartenberg (DH) notation where αi−1, ai−1, di and θi are the twist angle, link length, link offset and joint variable, respectively [23].

Table I: D-H Parameters

link	θ_i	d_i	a_{i-1}	α_{i-1}
1	θ_1	7.5	0	pi
2	θ_2	0	7.5	0
3	θ_3	0	5.5	0

ISSN (online): 2581-3048
Volume 7, Issue 3, pp 1-9, March-2023
https://doi.org/10.47001/IRJIET/2023.703001

$$A_i = \begin{bmatrix} c(\theta) & -c(\alpha)s(\theta) & s(\alpha)s(\theta) & a*c(\theta) \\ s(\theta) & c(\alpha)c(\theta) & -s(\alpha)c(\theta) & a*s(\theta) \\ 0 & s(\alpha) & c(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$A_1 = \begin{bmatrix} c(\theta_1) & 0 & s(\theta_1) & 0 \\ s(\theta_1) & 0 & -c(\theta_1) & 0 \\ 0 & 1 & 0 & 7.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$A_2 = \begin{bmatrix} c(\theta_2) & -s(\theta_2) & 0 & 7.5c(\theta_2) \\ s(\theta_2) & c(\theta_2) & 0 & 7.5s(\theta_2) \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$A_3 = \begin{bmatrix} c(\theta_3) & -s(\theta_3) & 0 & 5.5c(\theta_3) \\ s(\theta_3) & c(\theta_3) & 0 & 5.5s(\theta_3) \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$T_1 = \begin{bmatrix} c(\theta_1) & 0 & s(\theta_1) & 0 \\ s(\theta_1) & 0 & -c(\theta_1) & 0 \\ 0 & 1 & 0 & 7.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$T_2 = \begin{bmatrix} c(\theta_1)c(\theta_2) & -c(\theta_1)s(\theta_2) & s(\theta_1) & 7.5c(\theta_1)c(\theta_2) \\ s(\theta_1)c(\theta_2) & -s(\theta_1)s(\theta_2) & -c(\theta_1) & 7.5s(\theta_1)c(\theta_2) \\ s(\theta_2) & c(\theta_2) & 0 & 7.5s(\theta_2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$T_3 = \begin{bmatrix} c(\theta_2+\theta_3)c(\theta_1) & -s(\theta_2+\theta_3)c(\theta_1) & s(\theta_1) & 5.5c(\theta_1)*c(\theta_2+\theta_3) + 7.5c(\theta_2) \\ c(\theta_2+\theta_3)s(\theta_1) & -s(\theta_2+\theta_3)s(\theta_1) & -c(\theta_1) & 5.5s(\theta_1)*c(\theta_2+\theta_3) + 7.5c(\theta_2) \\ s(\theta_2+\theta_3) & c(\theta_2+\theta_3) & 0 & 5.5s(\theta_2+\theta_3)+7.5s(\theta_2)+5.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

The end effectors' orientation is represented by the first three columns of the matrix, while their location is represented by the final column [24]. Joint angles can be used to calculate the direction and placement of the end effectors Fig. 11 simulation.

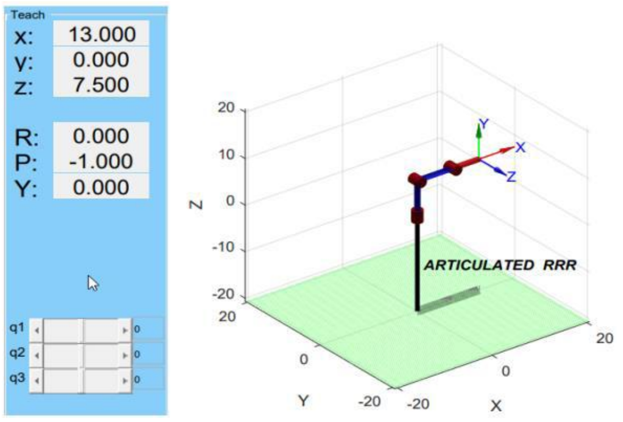
Teach
X: 13.000
Y: 0.000
Z: 7.500
R: 0.000
P: -1.000
Y: 0.000
q1 0
q2 0
q3 0

Figure 11: Simulation

3.2 Robotic Arm Inverse Kinematics

To solve inverse kinematic, triangulation method was implemented .The method is based on cosine rule [21]. And we can test the result of inverse and forward by simulation in MATLAB as shown in Fig. 11 and mathematically. If a triangle was provided, in Fig. 12 from top view and side view the (θ1)(θ2)(θ3) for inverse kinematic can be calculated.

$$\theta_1 = atan(X, Y) \quad \textbf{(8)}$$

$$\theta_2 = atan(\sqrt{X^2+Y^2}, Z-d) - atan(a_2 + c_3 * a_3, s_3 * a_3) \quad \textbf{(9)}$$

$$\theta_3 = atan(D, \sqrt{1-D^2}) \quad \textbf{(10)}$$

$$D = \frac{X^2+Y^2+(Z-d)^2 - a_2^2 - a_3^2}{2a_2a_3} \quad \textbf{(11)}$$

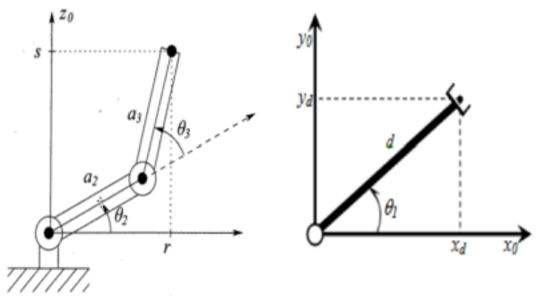

Figure 12: Top and side view

3.3 Jacobian

One of the most crucial parameters in the analysis and management of robot motion is the Jacobian. It is employed in the development of the dynamic equation for smooth trajectory and execution [21].

$$J\omega = \begin{bmatrix} -5.5s(\theta_1)c(\theta_2+\theta_3) \text{ -}7.5s(\theta_1)c(\theta_2) & -5.5c(\theta_1)s(\theta_2+\theta_3) \text{ -}7.5c(\theta_1)s(\theta_2) & -5.5c(\theta_1)s(\theta_2+\theta_3) \\ 5.5c(\theta_1)c(\theta_2+\theta_3) + 7.5c(\theta_1)c(\theta_2) & -5.5s(\theta_1)s(\theta_2+\theta_3) \text{ -}7.5s(\theta_1)s(\theta_2) & -5.5c(\theta_1)s(\theta_2+\theta_3) \\ 0 & 5.5c(\theta_2+\theta_3)\text{-}7.5c(\theta_2) & -5.5c(\theta_2+\theta_3) \\ 0 & -s(\theta_1) & -s(\theta_1) \\ 0 & c(\theta_1) & c(\theta_1) \\ 1 & 0 & 0) \end{bmatrix} \quad \textbf{(12)}$$

$$J_{11} = \begin{bmatrix} -5.5s(\theta_1)c(\theta_2+\theta_3) \text{ -}7.5s(\theta_1)c(\theta_2) & -5.5c(\theta_1)s(\theta_2+\theta_3) \text{ -}7.5c(\theta_1)s(\theta_2) & -5.5c(\theta_1)s(\theta_2+\theta_3) \\ 5.5c(\theta_1)c(\theta_2+\theta_3) + 7.5c(\theta_1)c(\theta_2) & -5.5s(\theta_1)s(\theta_2+\theta_3) \text{ -}7.5s(\theta_1)s(\theta_2) & -5.5c(\theta_1)s(\theta_2+\theta_3) \\ 0 & 5.5c(\theta_2+\theta_3)\text{-}7.5c(\theta_2) & -5.5c(\theta_2+\theta_3) \end{bmatrix} \quad \textbf{(13)}$$

$$\det_{J11} = 1.36c(\theta_2)s(\theta_3)\text{-}s(\theta_2) + s(\theta_2)(c(\theta_3)^2) + c(\theta_2)c(\theta_3)s(\theta_3) \quad \textbf{(14)}$$

3.4 Singularities

In manipulator configurations known as singularities, one or more degrees of freedom are eliminated. Indicating potential configurations at which singularities exist using the Jacobian and can be calculated the singularity of our arm manipulator if the right side of the equation (14) is equal to zero to find the value of angel that achievement this singularity.

1) The value of θ that satisfying singularity: if determined of Jacobian is equal zero at any configuration of theta ,the

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 7, Issue 3, pp 1-9, March-2023

https://doi.org/10.47001/IRJIET/2023.703001

singularity case is occurred. This is occurs at θ2= [0 or 180] degrees θ3=[0 or 180]degrees in the same time.

2) System Limitations: Table 2 displays the raw values for the robot arm's joint limits (servo control input values are integers [0,...., 254]) and the corresponding degrees.

Table II: Limitations

Joint	Joint limits [Deg]
1	$0 \leq \theta_1 \leq +140$
2	$0 \leq \theta_2 \leq +90$
3	$0 \leq \theta_3 \leq +100$

3) Length of links: in the our arm robot the length of links as illustrated in the table 3 this length move in the workspace without near from the singularity configuration.

Table III: Length of Links

link	length
base	75 mm
shoulder	75 mm
elbow	55 mm

3.5 Trajectory Planning

Assume we want to build a trajectory with two configurations and we want to define the start-end velocities for trajectory [25]. The planning of the robot's motion trajectory focuses on how to direct the robot's motion along the desired path. And the figure (13) is illustrated the result of Conic Trajectory of our project. And the figure (14) is illustrated the result of LSPB Trajectory of our project. And this is compare between trajectory types:

First type: Cubic polynomial trajectory:

1) Have four constraints.
2) Require a Third order polynomial.
3) Accelerations without constraints may excite vibration mode in the arm and reducing accuracy of tracking.

Second type: Quantic Polynomial Trajectories:

1) Have six constraints.
2) Require a fifth order polynomial.
3) Constraints on the acceleration as well as on the position with velocity to avoid excite vibration modes in the arm and increasing accuracy of tracking.

Third type: Linear Segments with Parabolic Blends (LSPB):

1) Appropriate when a constant velocity is desired along a portion of the path.
2) Three-part specification of the intended trajectory.
3) A quadratic polynomial represents the first portion from time t0 to time tb. A linear "ramp" velocity is the outcome.
4) The trajectory changes to a linear function in the second section at time tb, often known as the blend time. This is equivalent to a constant speed.
5) The trajectory then changes once more at time tf- tb, this time to a quadratic polynomial that makes the velocity linear.

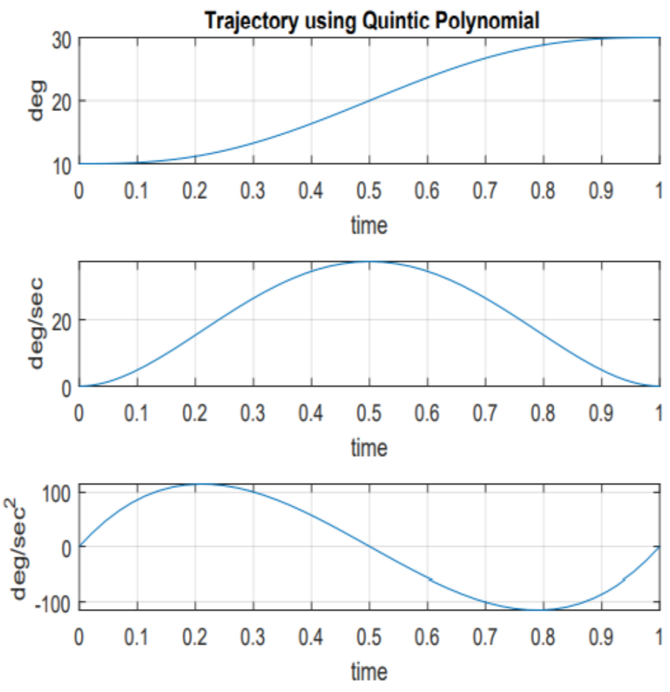

Figure 13: Conic Trajectory

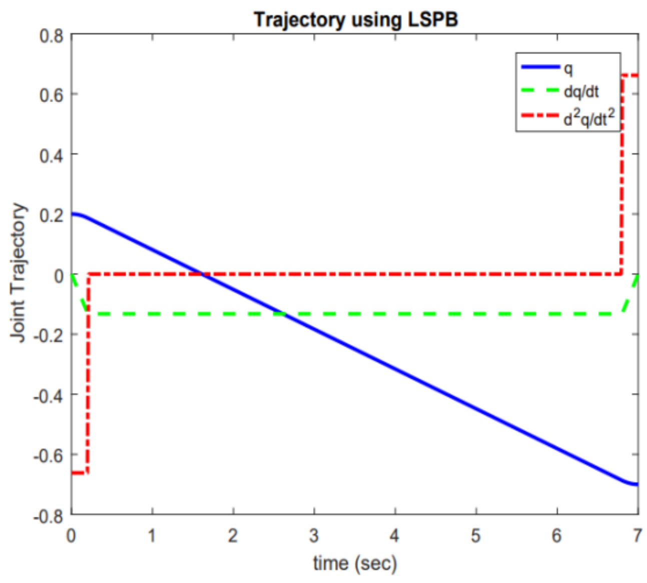

Figure 14: LSPB Trajectory

IV. CONCLUSION

The goal was accomplished when a low-cost, three degrees-of-freedom manipulator robot arm was conceived and constructed. The most latest open hardware and software technologies were used to achieve this. In advanced and intermediate robotics courses, it can be used to didactically illustrate the connection between theoretical and practical aspects of robot motions. Practical mechanical building laboratory exercises, the electrical and electronic components, and the software created for its control were used to describe the most important specifics in the paper.

As a result, SolidWorks was used to design the linkages. These were fabricated using a 3-D printer that was Machin code-programmed [26]. Along with wiring, power supply, and controller for actual robot, final assembly was completed. Robotics Toolbox was used to represent the robot mathematically. Direct and inverse kinematics problems were solved, and a visually appealing and user-friendly Simulink was produced in MATLAB.

The Arduino Servo library and serial communication were loaded into the controller software. In order to assess the performance of the prototype, a number of tasks (trajectories) were completed, and this led to an adequate working.

REFERENCES

[1] S. C. Suhaimin, N. L. Azmi, M. M. Rahman, and H. M. Yusof, "Analysis of point-to-point robotic arm control using pid controller," in *2019 7th International Conference on Mechatronics Engineering (ICOM). IEEE*, 2019, pp. 1–6.

[2] K. Jahnavi and P. Sivraj, "Teaching and learning robotic arm model," in *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). IEEE*, 2017, pp. 1570–1575.

[3] Y. Zhang, L. Li, M. Ripperger, J. Nicho, M. Veeraraghavan, and A. Fumagalli, "Gilbreth: A conveyor-belt based pick-and-sort industrial robotics application," in *2018 Second IEEE International Conference on Robotic Computing (IRC). IEEE*, 2018, pp. 17–24.

[4] I.Q. Kalimullah, A. Desikan, and V. Kalaichelvi, "Analysis of a proposed algorithm for point to point control of a 3 dof robot manipulator," in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR). IEEE*, 2017, pp. 289–292.

[5] R. K. Megalingam, S. Boddupalli, and K. Apuroop, "Robotic arm control through mimicking of miniature robotic arm," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE*, 2017, pp. 1–7.

[6] O. Olukayode, B. Adeboye, and K. Alawode, "Design and fabrication of a manually controlled electro-mechanical manipulator for educational purpose."

[7] B. House, J. Malkin, and J. Bilmes, "The voicebot: a voice controlled robot arm," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 183–192.

[8] J. Ang and K. Tan, "Forward kinematic solutions for a class of serial robots using the d-h parameter method," *IEEE transactions on robotic and automation*, vol. 13, no. 5, pp. 796–805, 1997.

[9] B. Han, K. Kim, and Y. Choe, "Forward kinematics solution of a 3-dof serial robot using the d-h convention," *Journal of Mechanical Science and Technology*, vol. 25, no. 1, pp. 99–105, 2011.

[10] G. S. Chirikjian, "Triangulation method for inverse kinematics," *The International Journal of Robotics Research*, vol. 11, no. 3, pp. 331– 341, 1992.

[11] J. Chen and Y. Wang, "Inverse kinematics solution of a robotic manipulator based on triangulation method," *Journal of Control, Automation and Electrical Systems*, vol. 19, no. 2, pp. 146–152, 2008.

[12] A.Goswami, Robotics: Theory and Industrial Applications. *Springer Science Business Media*, 2010.

[13] H. Zhang and X. Xie, "Identification and analysis of singularities in robot manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 6, pp. 615–623, 2010.

[14] A.Shafiee and M. Pirbazari, "Singularity analysis of redundant robots using polynomial geometry and numerical optimization," *Robotics and Computer-Integrated Manufacturing*, vol. 34, pp. 67–80, 2015.

[15] F. Li and L. Wang, "Trajectory planning for robot manipulators: a review," *Robotics and computer integrated manufacturing*, vol. 26, no. 3, pp. 201–212, 2010.

[16] R. Geraerts, J. De Schutter, and J. Swevers, "A survey on motion planning for robots," *Mechanism and Machine Theory*, vol. 42, no. 5, pp. 547–567, 2007.

[17] J. S. NaserAlanabi and J. Shrivastava, "Performance comparison of robotic arm using arduino and matlab anfis," *International Journal of Scientific & Engineering Research*, vol. 6, no. 1, 2015.

[18] E. B. Mathew, D. Khanduja, B. Sapra, and B. Bhushan, "Robotic arm control through human arm movement detection using potentiometers," in *2015 International Conference on Recent Developments in Control, Automation and Power Engineering (RDCAPE). IEEE*, 2015, pp. 298– 303.

[19] S. Patil and S. Lakshminarayan, "Position control of pick and place robotic arm," 2012.

ISSN (online): 2581-3048

Volume 7, Issue 3, pp 1-9, March-2023

https://doi.org/10.47001/IRJIET/2023.703001

[20] P. I. Corke and O. Khatib, Robotics, vision and control: fundamental algorithms in MATLAB. *Springer*, 2011, vol. 73.

[21] M. A. Qassem, I. Abuhadrous, and H. Elaydi, "Modeling and simulation of 5 dof educational robot arm," *in 2010 2nd International Conference on Advanced Computer Control*, vol. 5. IEEE, 2010, pp. 569–574.

[22] Y. Cui, P. Shi, and J. Hua, "Kinematics analysis and simulation of a 6-dof humanoid robot manipulator," in *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, vol. 2. IEEE, 2010, pp. 246–249.

[23] C. Marcu, G. Lazea, and R. Robotin, "An opengl application for industrial robots simulation," in *2006 IEEE International Conference on Automation, Quality and Testing, Robotics, vol. 2. IEEE*, 2006, pp.254–259.

[24] B. Koyuncu and M. G¨uzel, "Software development for the kinematic analysis of a lynx 6 robot arm," *International Journal of Computer and Information Engineering*, vol. 1, no. 6, pp. 1575–1580, 2007.

[25] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control, *jon wiley & sons,"* Inc, ISBN-100-471-649, 2005.

[26] S. Pei, F. Cui, and F. Chu, "Application of solidworks in mechanical design and drafting courses," in *3rd International Conference on Science.*

Citation of this Article:

Moayed Almobaied, Ahmed Abu Metair, Bashar M. Thabet, "Prototype of 3-DOF Pick and Place Arm Robot" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 7, Issue 3, pp 1-9, March 2023. Article DOI https://doi.org/10.47001/IRJIET/2023.703001

****