iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** XI **Month of publication:** November 2023

DOI: https://doi.org/10.22214/ijraset.2023.56469

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XI Nov 2023- Available at www.ijraset.com

Programming 2DoF Robot Arm Based on Forward Kinematics using Arduino

T. Srilatha[1], N. Snehalatha[2], CH. Prathibha[3], R. Seema[4], Y. Vasavi[5]
ECE Dept, G Narayanamma Institute of Technology and Science, Hyderabad, Ts, India

***Abstract:** This project's main goal is to use Arduino to program a 2-degree-of-freedom (2Dof) robot arm based on forward kinematics. The suggested system is made to precisely regulate the robot arm's motion for a variety of activities, such as pick-and-place duties and assembly procedures. The robot arm can be precisely positioned and controlled thanks to the forward kinematics model, which determines the end effector's position based on the joint angles. The system is put into practice and put through its paces in a real-world setting, and the results demonstrate its ability to conduct precise movements with decent accuracy. The 2DoF Robot Arm can be utilized to automate complex processes and boost productivity in a variety of industries, including manufacturing, healthcare, and robotics research.*

I. INTRODUCTION

The idea of forward kinematics, a cornerstone of robotics, is at the heart of this endeavor. By using joint angle input, forward kinematics determines how a robotic arm's parts are arranged in space. This notion is translated into actual movement by a 2-DoF robotic arm since it has two rotating degrees of freedom for each joint. Because of this, the arm can imitate a variety of spatial configurations similar to those of the human arm, such as picking up and placing objects and performing complex movements.

1. Determining End Effector Position: By taking joint angles into account, forward kinematics enables you to determine the end effector's 3D position (x, y, and z). Trigonometric functions and matrix transformations are used in this calculation.
2. Joint Angles to End Effector Mapping: The mapping of joint angles (1, 2) to the position and orientation of the end effector is made easier by forward kinematics. The forward kinematics computations manipulate the joint angles you supply to the software to determine the end effector's position and orientation.

II. FIGURES AND BLOCK DIAGRAM

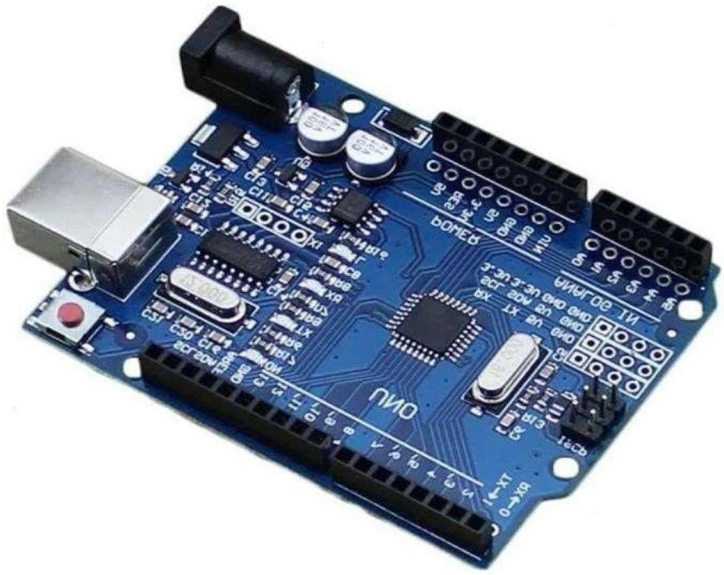


Fig. 1. Arduino board

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XI Nov 2023- Available at www.ijraset.com

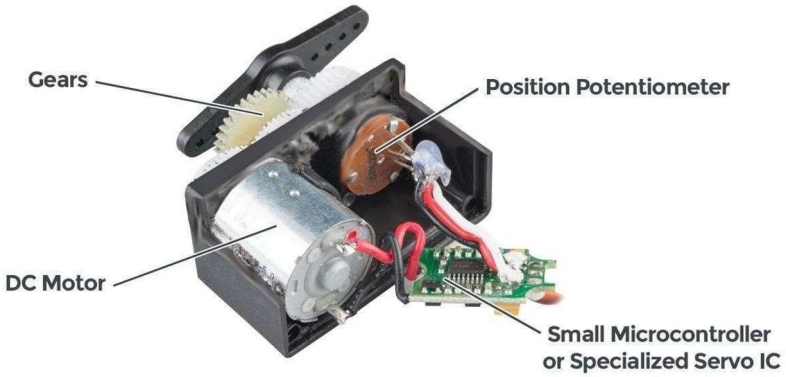


Fig. 2. Servo motor

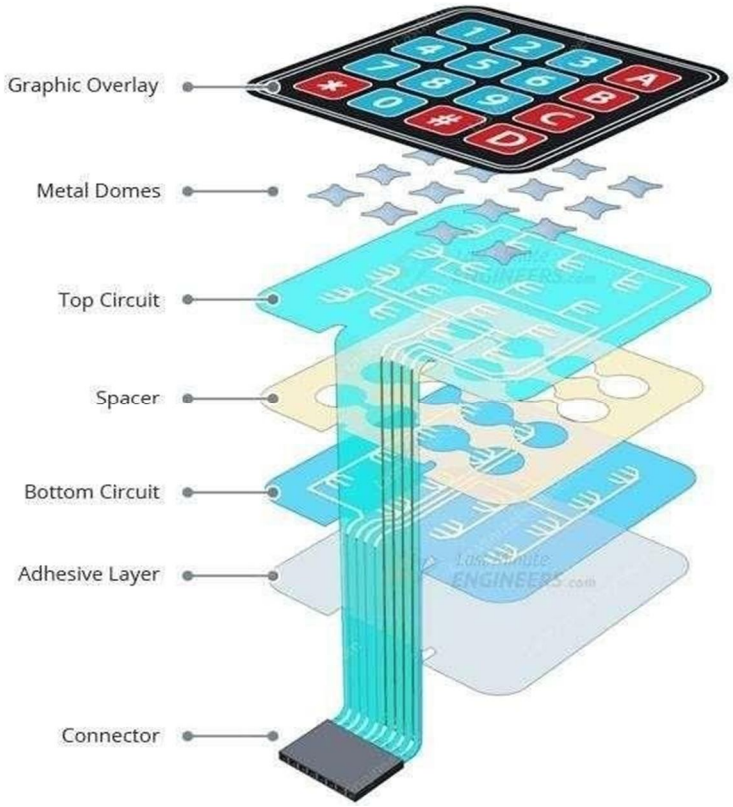


Fig. 3. Keypad internal structure

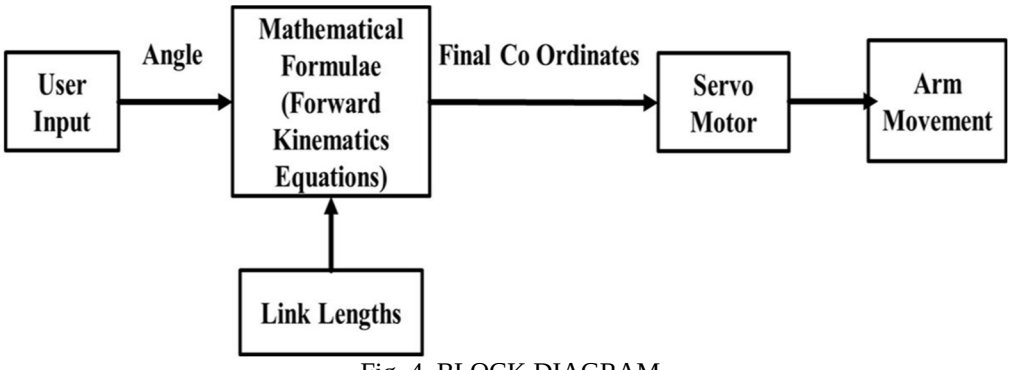


Fig. 4. BLOCK DIAGRAM

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue XI Nov 2023- Available at www.ijraset.com

III. WORKING

A 2DoF robotic arm can move precisely and controllably thanks to an Arduino, servo motors, and keypad. The Arduino serves as the main CPU for the robotic arm, with the servo motor controlling motion, position, speed, and torque. With an input keypad, user input is entered. Because each joint in a 2-DoF robotic arm has two rotational degrees of freedom in its component assembly, it can convert the forward kinematics theory—which describes how a robotic arm's components are arranged in space depending on joint angles—into actual movement. This allows the arm to simulate a large range of spatial configurations that are similar to those of the human arm, from placing and picking up objects to performing complex tasks. The Arduino Uno is a microcontroller board that is available as open-source software, and it was created by Arduino.cc using the Microchip ATmega328P MCU. A range of expansion boards (shields) and other circuits can be interfaced to the microcontroller board's sets of digital and analog input/output (I/O) pins. This board features 6 analog I/O pins, 14 digital I/O pins (six of which can be used to generate PWM), and can be programmed using the Arduino IDE with the use of a type B USB cable. Coders can create and upload code to the Arduino boards using the open-source Arduino IDE software. Operating platforms like Windows, Mac OS X, and Linux can all use the IDE application. Both C and C++ are supported programming languages. Integrated Development Environment is what's meant to be understood here. With the help of an input signal from a controller, an electric motor known as a servo motor can rotate or move to a particular position, speed, or torque. A suitable motor is connected to a position-feedback sensor, and a controller controls the motor's movement based on a preset point that is wanted. Although membrane keypads are available in multiple sizes, the 4×3 and 4×4 keypads (12 keys and 16 keys) are the most popular. They are user-friendly for all users, with a layout identical to a conventional telephone keypad.

A. Configure Hardware

Consider a level, sturdy surface, such as a wooden board. The base servo motor should then be inserted and secured to the board with screws. The foundation of the arm is this servo motor. The first link, about 22.5 cm in length, should be positioned above the base servo motor.

The second servo motor should then be positioned at the end of the first link and secured in place with screws. Attach the second link, measuring 22.5 cm in length, to the top of the second servo motor. The ground, supply, and control pins on each of the servo motors are colored orange, red, and brown, respectively. Attach the base servo motor's supply, ground, and control pins to pin number ten on the Arduino UNO. Then, do the same with the second servo motor's pin number eleven. Connect the pins on the Arduino UNO to the rows (R1, R2, R3, R4) and columns (C1, C2, C3, C4) of the 4X4 Keypad (R1, R2, R3, R4), correspondingly. Attach the ground pin and LED supply pin to the Arduino UNO's 13th and 13th pins, respectively. For a more ordered setup, attach the Arduino UNO to the wooden board and connect it to the AC adapter.

B. Upload Software

1) *Attach the Arduino using a USB Cable:* Use the USB cord to link the Arduino board to your computer. An Arduino and a USB 2.0 Type-A connection on a computer are often connected with a USB Type-B cable. The red power LED (designated PWR) ought to illuminate upon connection.
2) *Get the Arduino IDE Open:* Code is written using the IDE (Integrated Development Environment), which also compiles programs and uploads them to the Arduino so they can run.
3) *Choose the Port and Board for Arduino:* Choose your Board and Port from the Tools option at the top of the IDE. Choose the appropriate port (COMxx) from the list of ports provided.
4) *Put the Software Online:* To ensure that the code is error-free before submitting, select Verify. In the surroundings, click the "Upload" button. After a little while, the RX and TX LEDs on the board ought to start flashing. The status bar will display the message "Done uploading." if the upload is successful.
5) *Variables and Libraries:* The Keypad and Servo libraries, required for keypad input and servo motor control, are included. Establish character variables based on the pressed key (Kpressed).
6) *Configuring the Keypad:* Using a 2D array of keys, specify the arrangement of your 4x3 keypad. The physical arrangement of the keys on your keypad is mapped by this array.Indicate the keypad's row and column pin locations.Set your configuration as the Keypad object's first value.
7) *Setup of Servo Motor:* To control the servo motors, create the S_myservo and E_myservo servo objects.Establish your servos' starting positions (pos_1 and pos_2).

8) *Context:* To begin debugging, initiate serial communication (Serial.begin(9600)).S_myservo.attach(10) and E_myservo.attach(11) are the pins to which the servo motors should be attached.Turn on the LED pin by setting it as an output (digitalWrite(ledPin, HIGH)).Set up variables for keypad events and LED status.
9) *Loop:* Use the keypad to continuously check for a keypress.findKey().When a key is pressed, its value is recorded in Kpressed, printed to serial, and utilized for servo position determination.You can specify specific servo locations based on the key that is pressed. Keys 1 through 9 have cases in the code, each of which specifies a different servo position. The LED will blink if the '*' key is depressed, setting blink to true.Toggle the LED on/off by pressing the '#' key.
10) *Event Handling on Keypads:* Keypad events such key press, release, and hold are managed by the keypadEvent function. The LED status is toggled and the previous state is remembered when the '#' key is pushed. Release of the '*' key puts an end to any blinking that may have occurred during the hold event and returns the LED to its initial state.This code adds an LED control function and a keypad for controlling the position of the servo motors on the 2-DoF robotic arm. The code simulates various movements of the robotic arm by assigning the servos different positions based on the key pushed.

IV. ADVANTAGES

1) Compared to arms with more degrees of freedom, a 2DoF robot arm is comparatively basic, which makes it a great place for novices to start learning about kinematics and robotics principles.
2) Before scaling up to more complicated robotic systems, it can be used as a platform for quick prototyping to test and validate control algorithms, sensor integration, and other software features.
3) Simple pick-and-place operations, such as moving goods from one area to another, can be carried out by the arm, which is helpful for implementing automation in small-scale settings.
4) The reduced complexity of the system makes it easier to implement control algorithms, like PID control, for the 2DoF arm. This frees you up to concentrate on improving control tactics.
5) It is possible to design and test algorithms and control strategies rather quickly due to the system's simplicity. Iterative development techniques particularly benefit from this.

V. DISADVANTAGES

1) In comparison to more intricate arms, a 2DoF robot arm has less degrees of freedom. It is not appropriate for tasks requiring more mobility and flexibility because of this constraint, which limits the range of tasks it can accomplish.
2) More difficult ideas like inverse kinematics, dynamics, or complex motion planning may not be covered by a 2DoF arm, despite the fact that it is an excellent teaching tool. If you're trying to learn more about complex robotics topics, this may impede your comprehension.
3) For jobs requiring multidimensional motion and precise manipulation, a 2DoF arm is less suitable because to its constrained range of motion. Its suitability for more complex robotics applications is therefore restricted.

VI. RESULTS

Arduino is propelled by continuous technological progress, community cooperation, and the increasing need for accessible and reasonably priced robotic solutions. We may anticipate seeing even more inventive and varied uses of Arduino-based robotic arms in a range of sectors and fields as the field of robotics develops.

Table 5.1 Position Calculations

Key	**ø1**	**ø2**	(x,y)
1	80	60	(-3,14)
2	60	60	(0,14)
3	40	60	(3,14)
4	70	80	(-3,11)
5	40	90	(0,11)
6	20	90	(3,11)
7	75	105	(-3,8)
8	35	115	(0,8)
9	0	115	(3,8)

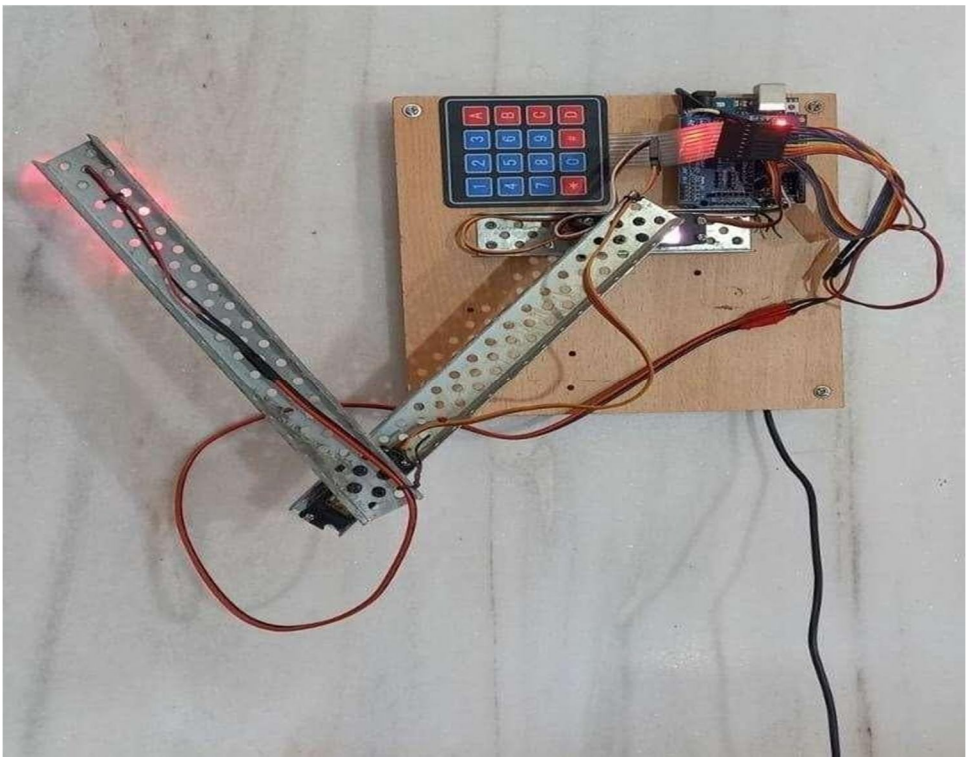


Fig. 5. Output when key 5 is pressed

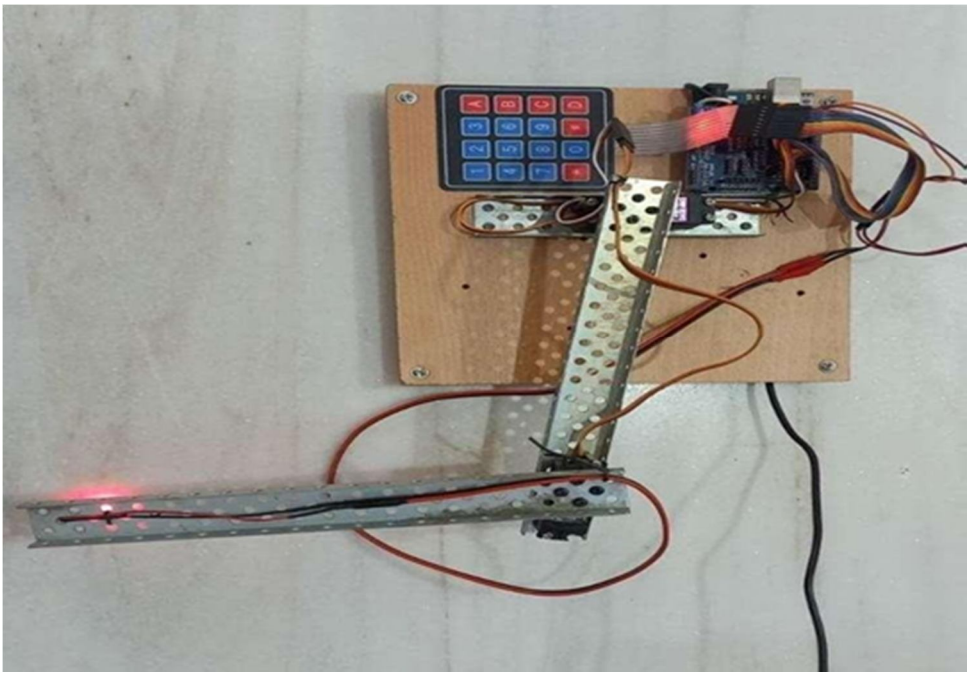


Fig. 6. Output when key 1 is pressed

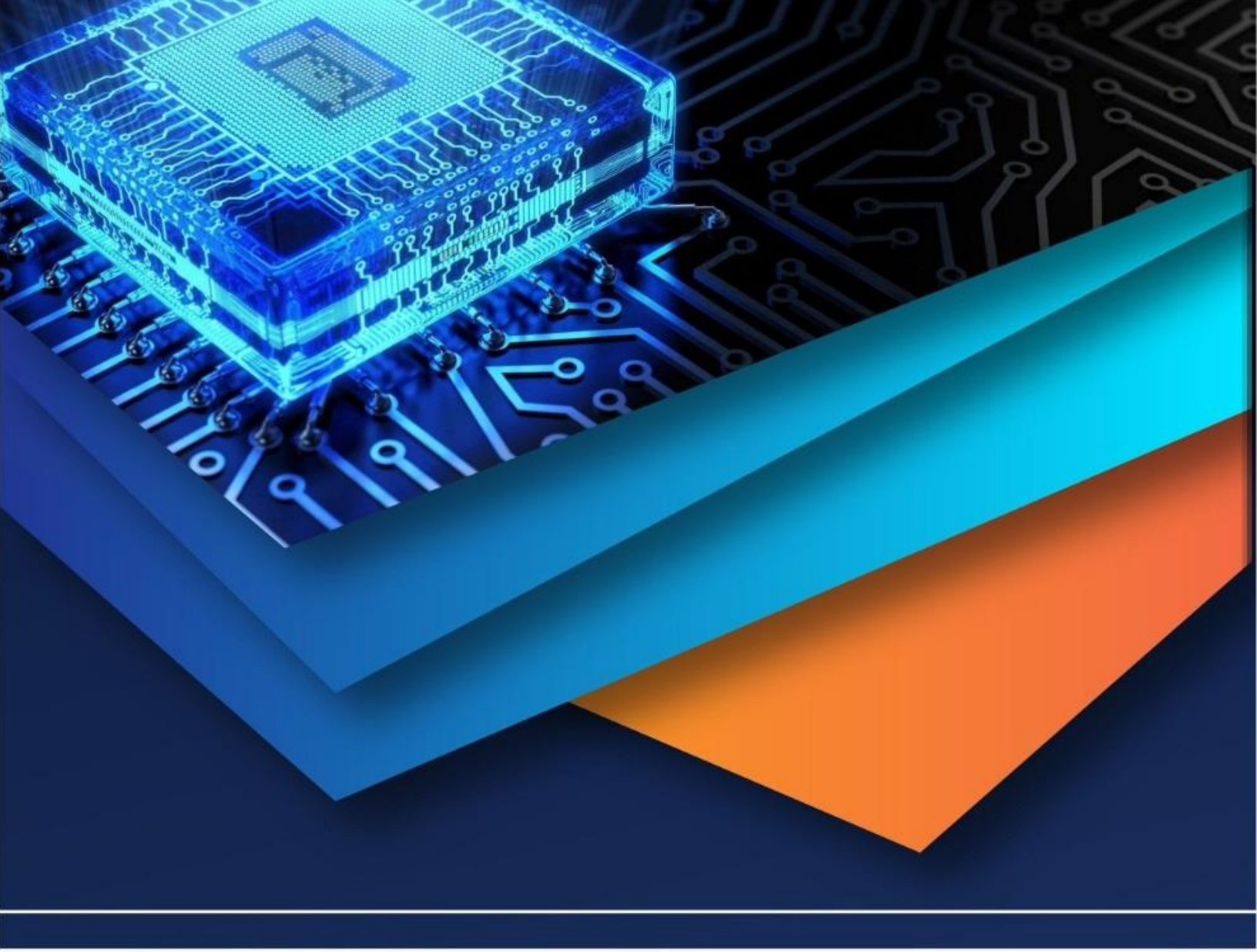
VII. CONCLUSIONS AND FUTURE SCOPE

In conclusion, programming an Arduino board to operate a two-degree-of-freedom (2DoF) robot arm using forward kinematics is an exciting project that enables you to manipulate the position and orientation of the robot's end-effector using code. You most certainly learned a lot about robotics, kinematics, and Arduino programming with this project. We now know how to use kinematic methods to determine the end-effector's position and orientation based on joint angles.

We have shown how forward kinematics may be used practically in robotics, which is essential for robotic arm control, automation, and many other industries. encountered difficulties and had to find solutions for issues with motion planning, accuracy, and calibration—all of which are typical in robotics projects. The future of robotics looks bright with a 2 Degree of Freedom (2DoF) robot arm based on forward kinematics and Arduino. Its potential rests in increasing its functions and uses. Its kinematic complexity may be increased, inverse kinematics can be added for precise control, and machine learning algorithms can be included for tasks like object recognition and path planning. Additionally, this technology can excel in areas like sensor integration, human-robot interaction interfaces, and industrial automation. Additionally, it is a versatile platform with promising futures in the developing field of robotics and automation due to its use in research, prototyping, 3D printing, and teleoperation for remote and dangerous activities. To fully utilize robotics' potential and adapt, it will be essential to stay current with its advances and trends.

REFERENCES

[1] Gasparetto, P. Boscariol, A. Lanzutti, R. Vidoni “Path Planning and Trajectory Planning Algorithms: A General Overview” Motion and Operation Planning of Robotic Systems. Mechanisms and Machine Science, vol 29. Springer, 2015 pp. 3-27.
[2] C. Brown, Kevin M. Passino - Intelligent Control for an Acrobot, Journal of Intelligent and Robotic Systems, Volume 18, Issue 3, pp 209– 248, March 1997
[3] F. Chaumette, Visual servo control, Part I: Basic approaches. IEEE Robotics and Automation Magazine, 13(4):82-90, December 2006.
[4] S. Hutchinson. Visual Servo Control, Part II: Advanced Approaches. IEEE Robotics and Automation Magazine, 14(1):109-118, March 2007



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY