RESEARCH ARTICLE OPEN ACCESS

Robotic Arm Design using Arduino Hardware and Software

S.Surappa*, M. Chandra kireeti**, B.Vishnu Vardhan***
* Assistant Professor, Mechanical engineering, Vignan Institute of Technology and Science, Deshmukhi, Hyderabad.
Email: mosesjayashali@gmail.com
**student, Mechanical engineering, Vignan Institute of Technology and Science, Deshmukhi, Hyderabad.
Email: madupuchandrakireeti04@gmail.com
*** student,Mechanical engineering, Vignan Institute of Technology and Science, Deshmukhi, Hyderabad.
Email: vishnubojja690@gmail.com

--************************----------------------------------

Abstract:

This research paper explores the design, implementation, and functionality of a robotic arm utilizing Arduino hardware and software. Robotic arms find applications in various fields, including manufacturing, healthcare, and education. Arduino, an open-source electronics platform, provides an accessible and cost-effective solution for designing robotic systems. The paper discusses the structural components, hardware requirements, software development, and functionality of a robotic arm controlled by Arduino. Additionally, it explores potential applications and future directions for advancements in this field.

Keywords **—** Robotic Arm, Arduino, Hardware, Software, Control, Applications.

--************************----------------------------------

I. INTRODUCTION

Robotic arms have become integral components in industrial automation, research laboratories, and educational settings. Their versatility in performing tasks such as material handling, assembly, and precise manipulations has led to widespread adoption across various industries. Designing robotic arms traditionally required sophisticated hardware and complex programming. However, with the advent of open-source platforms like Arduino, the barriers to entry have significantly reduced, enabling enthusiasts, students, and professionals to build and experiment with robotic systems.

A robotic arm typically comprises several structural components, including the base, links, joints, and end effectors. The base provides stability and serves as the foundation for the arm. Links connect the joints and determine the arm's reach and maneuverability. Joints, equipped with actuators such as servo motors or stepper motors, facilitate movement along different axes. The end effectors, attached to the arm's terminus, performs specific tasks based on the application requirements.

Banzi et al., provided a comprehensive introduction to programming with Arduino, a popular open-source hardware and software platform. While not specifically focused on robotics, it lays a crucial foundation for understanding Arduino-based projects, including those in robotics. Grimmett et al., delved into the practical aspects of building robots using Arduino. They discussed on motor control, sensors, and navigation algorithms, making it a valuable resource for both beginners and intermediate robotics enthusiasts. Pratt et al., discussed fundamental concepts in robotics, specifically forward and inverse kinematics. Understanding these concepts is essential for designing and controlling robotic manipulators, making this a pertinent resource for robotics engineers. Robinson et al., offered practical guidance on building robots using the LEGO Mindstorms NXT platform. It covers basic principles of robotics and provides hands-on projects, making it relevant for those interested in introductory robotics [1-4].

Zaldivar et al., explored the intersection of Arduino and the Internet of Things (IoT), showcasing projects that integrate sensors, actuators, and communication modules. While not exclusively focused on robotics, it offers insights into connecting Arduino-based devices to broader networks, which can be valuable in robotic applications. Corke's et al., proposed a comprehensive overview of robotics, covering topics such as robot kinematics, dynamics, and control. While MATLAB-centric, it offers valuable theoretical insights and practical implementations applicable to Arduino-based robotics. Siciliano et al., derived a seminal work in the robotics field, it covers a wide range of topics in robotics, from fundamentals to advanced research areas. It serves as a comprehensive reference for roboticists, including those interested in Arduino-based projects. Craig's et al., proposed foundational knowledge essential for understanding robotic systems, providing a thorough introduction to robot mechanics, kinematics, and control. Spong et al., proposed modelling and control techniques for robotic systems, emphasizing both theoretical principles and practical implementation. While not Arduino-focused, it provides valuable insights into control strategies applicable to Arduino-based robots [5-9].

Ogata et al., proposed a comprehensive treatment of modern control theory, covering topics such as PID control, state-space analysis, and robust control techniques. While not Arduino-specific, it provides foundational knowledge applicable to designing control systems for robotic applications. Ljung et al., provides a comprehensive overview of system identification, a crucial aspect of robotics involving the modelling and characterization of dynamic systems. Understanding system identification techniques is essential for designing and controlling robotic systems effectively. Lynch et al., proposed a modern perspective on robotics, covering mechanics, motion planning, and control algorithms. Lynch and Park delve into advanced topics such as kinematics, dynamics, and trajectory optimization, making it a valuable resource for robotics researchers and practitioners [10-12].

The Atmel ATmega328/P is a microcontroller commonly used in Arduino boards. The datasheet provides detailed technical specifications and programming information, essential for understanding and utilizing the capabilities of the ATmega328/P in robotics projects. The SG90 micro servo is a compact and affordable servo motor commonly used in robotics projects for actuation and control tasks. Understanding its specifications and capabilities is crucial for integrating it effectively into Arduino-based robotic systems. Adafruit's motor/stepper/servo shield is an Arduino add-on board that simplifies the control of motors and servos. This kit provides a convenient solution for driving various types of motors in robotics applications, offering versatility and ease of use. The DRV8835 dual motor driver carrier is a compact module designed for driving two DC motors or one stepper motor. It offers features such as built-in protection circuitry and a small form factor, making it suitable for use in Arduino-based robotic projects requiring motor control. The official Arduino website serves as a central hub for resources related to the Arduino platform, including documentation, tutorials, and community forums. It provides valuable support for individuals interested in learning and experimenting with Arduino-based robotics projects [13-17].

Lewis et al., explored the evolution of robot behavior using a simulated 8-legged walker as a case study. It demonstrates the application of evolutionary algorithms to optimize the gait of a robotic system, showcasing innovative approaches to robotic design and control. Khatib et al., presented a real-time obstacle avoidance algorithm for manipulators and mobile robots based on potential fields. It introduces practical techniques for navigating complex environments, highlighting the importance of robust motion planning strategies in robotics applications. Zinn et al., proposed a novel actuation approach for designing human-friendly robots with improved safety and dexterity. By integrating compliant mechanisms and advanced control strategies, they addressed key challenges in human-robot interaction, paving the way for safer and more versatile robotic systems [18-20]

II. Hardware and Software Development

To build a robotic arm using Arduino, several hardware components are necessary. Arduino Board: Acts as the central processing unit, controlling motor movements and receiving input from sensors. Servo Motors or Stepper Motors: Power the joints and enable controlled movement. Motor Drivers: Interface between the Arduino board and motors, regulating power and direction. Power Supply: Provides electrical power to the Arduino board, motor drivers, and motors. Sensors (optional): Encoders, limit switches, or proximity sensors can be integrated for feedback control and collision avoidance.

Arduino's intuitive Integrated Development Environment (IDE) simplifies the software development process for controlling robotic arms. Writing code in Arduino programming language, based on C/C++, to define arm movements and behavior. Utilizing libraries such as Servo.h or Stepper.h to facilitate motor control and simplify coding. Implementing algorithms for forward and inverse kinematics to calculate joint angles for precise positioning. Developing user interfaces for controlling the arm, either through physical inputs (buttons, joysticks) or digital interfaces (smartphone apps, Bluetooth/Wi-Fi communication). The functionality of a robotic arm controlled by Arduino encompasses various aspects.

Movement Control: Arduino code translates user inputs or commands into motor movements, facilitating desired arm motions. End Effector Control: The arm's end effector performs specific tasks such as gripping objects, painting, or performing precise manipulations. Feedback Control (optional): Sensors provide feedback on the arm's position and orientation, enabling adjustments for enhanced precision and accuracy. User Interface: Interfaces allow users to interact with the robotic arm, providing inputs for controlling movements and execution. Figure 1 shows the model developed of robotic arm. Program used for the controlling the movements of arm is shown below.

```
#include <Servo.h>
Servo baseServo;
Servo shoulderServo;
Servo elbowServo;
Servo gripperServo;
int baseAngle = 90;
int shoulderAngle = 90;
int elbowAngle = 90;
int gripperAngle = 0;

void setup() {
  baseServo.attach(BASE_PIN);
  shoulderServo.attach(SHOULDER_PIN);
  elbowServo.attach(ELBOW_PIN);
  gripperServo.attach(GRIPPER_PIN);
}
void loop() {
  // Read input or receive commands
  // Adjust angles based on input
  // Move servos to desired angles
  baseServo.write(baseAngle);
  shoulderServo.write(shoulderAngle);
  elbowServo.write(elbowAngle);
  gripperServo.write(gripperAngle);
}
```

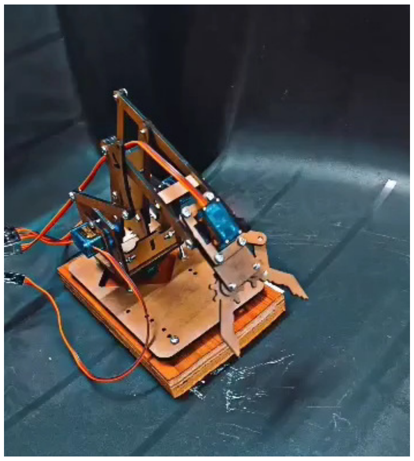

Figure 1 Model of Robotic arm using arduino

III. Applications and Future Directions

Robotic arms designed with Arduino hardware and software find applications across diverse domains. Education: Arduino-based robotic arms serve as educational tools for teaching robotics, programming, and mechatronics. Prototyping: Rapid prototyping capabilities of Arduino enable quick iteration and experimentation in developing robotic solutions. Research: Researchers utilize Arduino-based robotic arms for exploring advanced control algorithms, human-robot interaction, and autonomous systems. Home Automation: Arduino-powered robotic arms can automate household tasks, such as picking up objects or assisting individuals with disabilities.

IV. CONCLUSIONS

Designing a robotic arm using Arduino hardware and software offers a cost-effective and accessible approach to developing robotic systems. By leveraging Arduino's simplicity and versatility, enthusiasts, students, and professionals can explore the fundamentals of robotics, experiment with different configurations, and innovate in various application domains. As technology continues to evolve, Arduino-based robotic arms are expected to play an increasingly significant role in shaping the future of automation and robotics.

REFERENCES

[1] Banzi, M. (2011). Arduino Programming in 24 Hours. Sams Publishing.
[2] Grimmett, G. (2015). Arduino Robotics. Packt Publishing.
[3] Pratt, G., & Williamson, M. (1995). Series of six articles on robotics: Part 5, Forward and Inverse Kinematics. Industrial Robot: An International Journal, 22(6), 21-24.
[4] Robinson, R. S. (2009). Basic Robot Building With LEGO Mindstorms NXT 2.0. Que Publishing.
[5] Zaldivar, D., & Gehani, A. (2019). Internet of Things with Arduino Blueprints. Packt Publishing.
[6] Corke, P. (2017). Robotics, Vision and Control: Fundamental Algorithms in MATLAB® Second, Completely Revised, Extended and Updated Edition. Springer.
[7] Siciliano, B., & Khatib, O. (Eds.). (2016). Springer Handbook of Robotics. Springer.
[8] Craig, J. J. (2005). Introduction to Robotics: Mechanics and Control (3rd ed.). Pearson Prentice Hall.
[9] Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2005). Robot Modeling and Control. John Wiley & Sons.
[10] Ogata, K. (2010). Modern Control Engineering (5th ed.). Prentice Hall.
[11] Ljung, L. (1999). System Identification: Theory for the User (2nd ed.). Prentice Hall.
[12] Lynch, K. M., & Park, F. C. (2017). Modern Robotics: Mechanics, Planning, and Control. Cambridge University Press.
[13] Atmel Corporation. (2016). Atmel ATmega328/P Datasheet. Retrieved from https://www.microchip.com/wwwproducts/en/ATmega328
[14] Tower Pro. (2024). SG90 Micro Servo. Retrieved from https://www.towerpro.com.tw/product/sg90-7/
[15] Adafruit Industries. (2024). Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit. Retrieved from https://www.adafruit.com/product/1438
[16] Pololu Corporation. (2024). DRV8835 Dual Motor Driver Carrier. Retrieved from https://www.pololu.com/product/2135
[17] Arduino. (2024). Arduino - Home. Retrieved from https://www.arduino.cc/
[18] Lewis, M. A., & Langton, C. G. (1991). Evolving Robot Behavior in the Real World: The Gait of a Simulated 8-Legged Walker. In Proceedings of Artificial Life II (pp. 501-545). Addison-Wesley.
[19] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research, 5(1), 90-98.
[20] Zinn, M., Khatib, O., Roth, B., & Salisbury, J. K. (2004). A new actuation approach for human friendly robot design. The International Journal of Robotics Research, 23(4-5), 379-398.