IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Intelligent Motion Control Of Servo Motors Using Arduino For Robotic Systems

[1]Dr. Satyabodh M Raichur, [2]Dr Ravi Kumar R, [3]Dr Nagaraj Namdev, [4]Vaishak Kamath B, [5]K Rakesh Krishna , [6]Charan B Shetty

[1]Assistant Professor, Department of Mechanical Engineering, APS College of Engineering, [2]Head, Department of Mechanical Engineering, APS Polytechnic, [3]Research Scholar, Indian Institute of Science, [4]Student , PES University

[5]Student Bangalore Institute of Technology , Bengaluru, India

Abstract: This paper presents a comprehensive approach to automated motion control of servo motors using an Arduino microcontroller, specifically designed for robotic applications. The study focuses on the precise actuation, synchronization, and real-time control of a robotic arm using SG90 servo motors. By integrating Arduino's computational capabilities with sensor feedback mechanisms, the system optimizes motion accuracy, responsiveness, and efficiency. The proposed framework leverages open-source hardware and software, offering a cost-effective, scalable, and flexible solution suitable for industrial automation, robotics research, and educational purposes. Additionally, the system implements optimized control algorithms to enhance smoothness and minimize latency in motor movements. Experimental results validate the system's ability to execute coordinated and precise robotic operations, demonstrating its practical viability for real-world applications in automation and robotics

Index Terms - Servo Motors, Arduino, Motion Control, Robotic Arm, Automation, Microcontroller, Real-Time Control, Sensor Feedback, Precision Actuation, Robotics.

I. INTRODUCTION

The integration of automation and robotics across modern industries has significantly transformed sectors such as manufacturing, healthcare, and logistics, driving the demand for highly precise and efficient motion control systems [1–3]. These systems are fundamental to ensuring accurate, reliable operation in robotic mechanisms, especially in tasks that require meticulous coordination and responsiveness [4–6]. Central to such systems are servo motors, which have become essential components in robotics due to their capability to deliver precise positioning and speed control.

Unlike stepper motors that operate on fixed incremental steps, servo motors function through closed-loop control, enabling real-time adjustments based on feedback. This dynamic adaptability makes servo motors ideal for high- precision applications requiring smooth, repeatable movements [7,9]. However, traditional motion control approaches often encounter challenges such as overshooting, latency, and synchronization inconsistencies when managing multiple actuators, ultimately affecting the overall reliability and efficiency of robotic systems [5,6]. Past research has emphasized the advantages of servo-based control systems in robotics, yet many of these studies rely heavily on expensive industrial controllers or complex hardware configurations . To address these limitations, this study proposes the development of an intelligent motion control system using an Arduino microcontroller—offering a cost-effective, scalable alternative that does not compromise performance. By leveraging Arduino's flexibility and open-source ecosystem, the proposed system introduces advanced control strategies that are accessible to developers, researchers, and educators alike . The system integrates optimized control algorithms, notably proportional-integral-derivative (PID) control, which has been widely adopted in motion regulation for its ability to enhance stability. While PID controllers are not new to

robotic systems, prior implementations have faced issues such as slow response and abrupt motor behavior. In this work, improved PID tuning techniques are employed to address these drawbacks, resulting in more fluid and precise actuation.

Furthermore, the motion control framework described here emphasizes real-time performance without relying heavily on external sensors. While previous studies often incorporate feedback mechanisms like encoders and inertial measurement units (IMUs) to enhance precision, this approach demonstrates how refined algorithmic control within the Arduino environment alone can achieve high accuracy and responsiveness. Nevertheless, the system retains the flexibility to incorporate such sensors when needed, further enhancing adaptability.

Ultimately, the intelligent control system developed in this study demonstrates practical viability across a wide range of robotic applications. Experimental results validate its effectiveness in coordinating multiple servo motors with improved synchronization, reduced latency, and high precision. This work builds upon and extends existing research by offering a low-cost yet high-performance motion control solution, paving the way for more accessible and adaptable robotic automation in both industrial and educational settings.Despite significant advancements in robotic motion control, current systems predominantly rely on costly industrial hardware and complex configurations, limiting their accessibility for educational, experimental, and low-budget applications. Additionally, many existing solutions depend heavily on external sensors such as encoders and inertial measurement units to achieve motion precision, which adds to system complexity and cost. While the use of PID controllers is well-established, prior implementations often struggle with issues like delayed response and poor synchronization in multi-servo setups. Moreover, there is limited research exploring the potential of software-optimized motion control on open-source platforms like Arduino, particularly in achieving high-precision, real-time coordination without the need for extensive sensor integration. This study addresses these gaps by developing an intelligent, cost-effective, and scalable motion control system that leverages optimized control algorithms within the Arduino framework to achieve responsive and accurate servo motor coordination.

II. System Architecture

The proposed system consists of an Arduino microcontroller, SG90 servo motors, and sensor feedback mechanisms. The motion control is achieved through optimized algorithms that ensure smoothness and minimal latency in movement. The system is designed to be cost-effective, scalable, and easily adaptable for various automation applications.

The traditional method of husking rice using a motor and pestle is one of the oldest known agricultural practices, with evidence tracing back thousands of years across Asia, Africa, and parts of South America.

(a) According to historical records (Sharma, 1991; Takeda, 2002), early societies developed simple tools like large wooden or stone mortars paired with long pestles to ease the labour of separating rice grains from their husks.

(b) Researchers (Singh & Yadav, 2008) describe that the process involved rhythmic pounding, which, when skilfully performed, minimized grain breakage while efficiently loosening the husk. The success of this method largely depended on the moisture content of the paddy, the material of the mortar, and the technique of the user.

(c) Anthropological studies (Lansing, 1991) highlight that rice husking was often a communal activity, particularly among women, fostering social cohesion during harvest seasons. In addition, traditional practices emphasized sustainability: the tools were made from locally available resources and had minimal environmental impact.

(d) Technological analyses (Zhao et al., 2010) suggest that while this method was highly labor-intensive, it required little financial investment, making it accessible to rural communities. However, researchers like Fernando (2014) noted the limitations in processing speed and consistency compared to later mechanical or motorized huskers.

(e) Modern agricultural scholars recognize (ILO, 2012) that traditional husking preserved the nutritional value of rice better than some aggressive modern methods, which can damage the bran layer. Today, even though mechanical huskers dominate rice processing, traditional

The time series monthly data is collected on stock prices for sample firmsand relative macroeconomic variables for the period of 5 years. The data collection period is ranging from January 2010 to Dec 2014. Monthly prices of KSE -100 Index is taken from yahoo finance.

2.1 Components Used:

- Arduino Microcontroller (1): Acts as the central processing unit for motion control.
- SG90 Servo Motors (2): Provides precise angular movement for robotic applications.
- Power Supply: Provides necessary voltage and current for motor operation, DC Supply has been preferred on the form of Cell or Battery.
- Communication Interface (Serial/I2C/SPI): Enables interaction with external devices.

Figure 2.1 Arduino Uno Microcontroller

Figure 2.2: DC Servo Motor

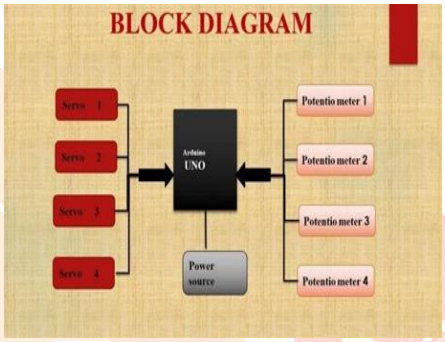


Figure 2.3: Block Diagram of Arduino UNO consists of 4 Servo (SG90) along with Potentiometers and Power Source

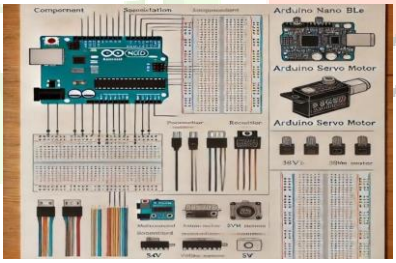


Figure 2.4: Various Components used for the Experimentation setup

III. EXPERIMENTATION:

. 3.1Communication between Arduino Boards and Servo Motors: The Arduino boards communicate with SG90 servo motors and other Arduino units using the I2C (Inter-Integrated Circuit) protocol. This protocol is widely used for communication between multiple microcontrollers and peripherals due to its simplicity and efficiency. The setup consists of an Arduino Nano 33 BLE, a compact microcontroller board responsible for controlling the servo motors. Two SG90 servo motors are used to achieve precise motion control. The system is powered by an external

7.4V power supply, which provides adequate power to the servos. A 5V voltage regulator is incorporated to step down the 7.4V input, ensuring safe operation of the components. A breadboard and connecting wires facilitate the electrical connections between the components.

The setup consists of an Arduino Nano 33 BLE, a compact microcontroller board responsible for controlling the servo motors. Two SG90 servo motors are used to achieve precise motion control. The system is powered by an external 7.4V power supply, which provides adequate power to the servos. A 5V voltage regulator is incorporated to step down the 7.4V input, ensuring safe operation of the components. A breadboard and connecting wires facilitate the electrical connections between the components.

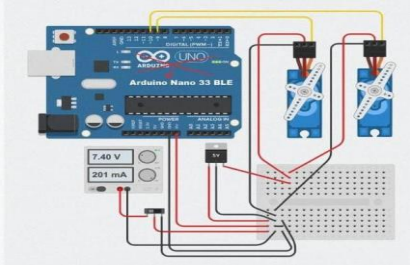


Figure 3.1 Circuit diagram of an experimental setup

In terms of wiring, the **power connections** include red and black wires, where the external 7.4V power supply is regulated down to 5V to safely power the servos. The black wire serves as the common ground for all components, ensuring proper circuit operation. For **control signals**, the yellow wires connect the servo motors' signal pins to the **PWM (Pulse Width Modulation) pins** on the Arduino, allowing precise movement control through programmed instructions.

3.2. Servo Motor Control: Servo motors like the SG90 are controlled using Pulse Width Modulation (PWM) signals, which are generated using the Arduino's built-in Servo library. Each servo motor is assigned to a dedicated PWM pin on the master Arduino. By adjusting the characteristics of the PWM signal, the position of the servo motor shaft can be precisely controlled.

How PWM Controls Servo Motor Movement?: PWM (Pulse Width Modulation) plays a crucial role in controlling the movement of a servo motor by sending a series of repeating pulses, where the duty cycle determines the servo's position. In the case of SG90 servo motors, the pulse width directly corresponds to specific angular positions. A 1 ms pulse width moves the servo to 0°, which is its minimum position. A 1.5 ms pulse width positions the servo at 90°, marking the neutral or midpoint position. A 2 ms pulse width moves the servo to 180°, which is its maximum position.

The Arduino microcontroller continuously updates these PWM signals based on user inputs or pre-programmed instructions, allowing precise and dynamic control of the servo motor's movement.

User Input and Servo Control System:

- A joystick, push buttons, or predefined motion sequences can be used to control servo movement.
- The master Arduino reads user input and adjusts the PWM values sent to the servo motors.

3.3 Flexibility of the Servo Control System: The system enables real-time speed adjustments by varying the rate at which the PWM values are updated, allowing for smooth and controlled motion transitions. Additionally, the range of motion can be fine-tuned by setting specific limits on the PWM signals, ensuring that the servo does not exceed its intended rotation range and preventing mechanical strain. When controlling multiple servo motors, their movements can be synchronized, allowing for coordinated motion sequences. This feature is particularly useful in robotic applications, where precise and simultaneous servo actuation is required to achieve complex and dynamic movements which are shown in the figure below (3.3 and 3.4 respectively).

Figure 3.3 & 3.4 depicts the servo motors connected to Aurdino and Power supply

IV. RESULTS AND DISCUSSIONS:

4.1 Software Implementation: The software implementation consists of two parts: Master AURDUINO Program: This program handles user input and sends control signals to the slave AURDUINO. It also monitors the overall system's performance and handles communication with external devices like a computer or remote control.

Code Snippet on Aurduino 1 saved on .c

```
#include <Wire.h> #include <Servo.h>
Servo servo1, servo2, servo3; // Define the servo motors void setup() {
Wire.begin(); // Start I2C communication servo1.attach(9); // Attach servos to pins servo2.attach(10);
servo3.attach(11);
}
void loop() {
int angle1 = analogRead(A0); // Read input from a potentiometer or joystick int angle2 = analogRead(A1);
// Map the input values to PWM signals
servo1.write(map(angle1, 0, 1023, 0, 180));
servo2.write(map(angle2, 0, 1023, 0, 180)); delay(20);
```

 Slave AURDUINO Program: The slave receives control commands via I2C and adjusts the servo positions accordingly.

```
#include <Wire.h> #include <Servo.h> Servo servo1, servo2; void setup() {
Wire.begin(8); // Slave address 8 servo1.attach(9); // Attach servos to pins servo2.attach(10);
Wire.onRequest(requestEvent); // Function to handle data requests
}
void loop() { delay(100);
}
void requestEvent() {
// Send motor position data back to master (example) Wire.write(servo1.read()); Wire.write(servo2.read());
}
```

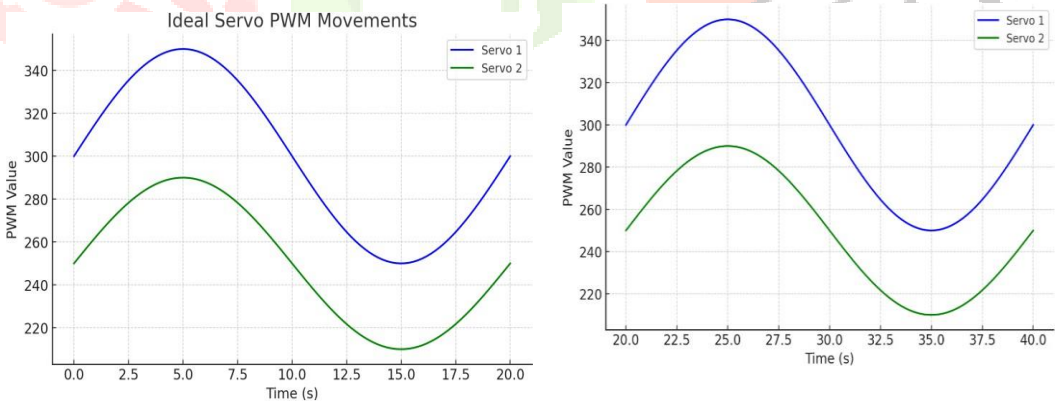


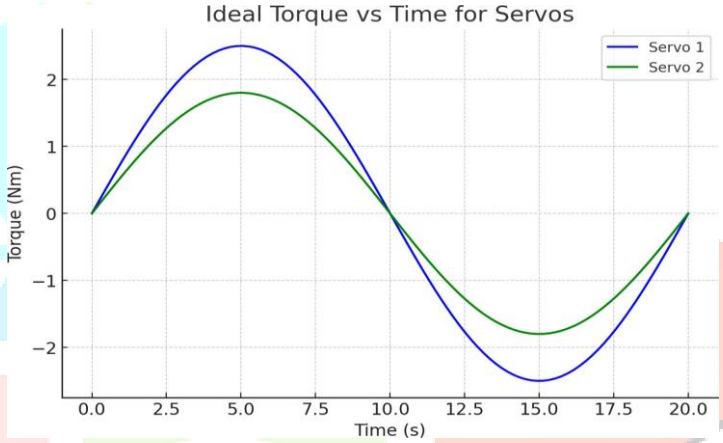
Figure 4.1: Ideal Servo PWM Movements of 2 servos for 20 seconds Interval 40 seconds Intervals

4.3 Analysis of Servo Motion Based on PWM Signal Patterns: Servo 1 (Blue Curve) :Servo 1 that is depicted in the figure 5.1 & 5.2 exhibits a smooth sinusoidal motion, indicating that its PWM values oscillate predictably over time. The PWM signal fluctuates between 250 and 350, with a central value of 300, creating a periodic movement pattern. This ensures gradual acceleration and deceleration, leading to natural and fluid motion—ideal for applications requiring smooth transitions.

Servo 2 (Green Curve): Similar to Servo 1, Servo 2 in the figure no 5.1 & 5.2 also follows a sinusoidal trajectory, but with a slightly different range. The PWM signal varies between 210 and 290, centred at 250, meaning its overall movement amplitude is smaller compared to Servo 1. This suggests a more restricted yet controlled motion, which may be beneficial for applications needing precise but less exaggerated movements.

4.4 Outcomes and Significance: Both servos exhibit continuous and wave-like movements, which ensure smooth transitions between different positions. The peaks in the waveform represent the maximum extension, while the troughs indicate the minimum position of the servo arms. This type of controlled actuation is highly effective for robotic applications, including robotic arms, automated systems, and motion-controlled mechanisms, where predictable and fluid movement is essential for precision and stability.

4.5 : For Ideal Torque vs Time: Both Servo 1 (blue) and Servo 2 (green) exhibit a smooth and periodic oscillation, indicating a sinusoidal torque pattern over time. This oscillation ensures that the servos generate torque in a controlled and predictable manner, allowing for gradual force application rather than abrupt changes. The torque variation in Servo 1 follows a broader range, suggesting that it experiences higher fluctuations in force application, whereas Servo 2 maintains a more restricted oscillation, indicating a smoother and more controlled torque output. The peaks in the graph represent the maximum torque output, while the troughs correspond to the minimum torque levels, showcasing a continuous and stable torque transition. This behaviour is essential for robotic systems, automated actuators, and precision mechanisms,



where maintaining an optimal balance between torque and motion stability is crucial. The sinusoidal nature of the torque distribution ensures smooth mechanical operation, reduced wear on components, and efficient energy utilization, making it ideal for applications requiring

Figure 4.3: Plot for Ideal Torque vs Time

consistent and controlled force application. This suggests a controlled and predictable motion, possibly a cyclic movement like a robotic arm or a motor-driven mechanism.Peak and Minimum Torque Values: For Servo 1, the torque reaches a peak of +2.5 Nm and drops to a minimum of -2.5 Nm, indicating a balanced oscillation between positive and negative torque. Similarly, Servo 2 achieves a peak of +1.8 Nm and a minimum of -1.8 Nm, demonstrating a slightly lower force output compared to Servo 1. The presence of negative torque values signifies that the servos apply force in opposite directions during different phases of their movement..Both Servo 1 and Servo 2 reach their peak and minimum torque values almost simultaneously, highlighting their synchronized operation despite generating different force outputs. This synchronization plays a crucial role in ensuring coordinated movement, particularly in robotic and automation systems where multiple actuators must function together while maintaining individual torque characteristics. The ability to distribute torque dynamically across multiple actuators enhances motion stability, prevents mechanical imbalances, and reduces wear and tear on components. This type of torque behaviour is especially advantageous for applications requiring smooth and oscillatory motion, such as robotic joints, servo-controlled arms, and actuators used in industrial automation. In these systems, maintaining precise and predictable torque variations is essential for achieving controlled and fluid movements. The synchronization of servo motors ensures that different mechanical parts interact harmoniously, reducing sudden jerks or misalignments that could otherwise lead to inefficiencies or mechanical failures.

Moreover, the controlled variation in torque contributes to overall system efficiency by optimizing energy usage and minimizing unnecessary strain on actuators. By reducing mechanical stress, the system extends the operational lifespan of components and ensures consistent performance over time. This makes synchronized

servo control a valuable approach for precision-driven mechanical systems, where accuracy, stability, and durability are critical factors in achieving optimal functionality

V. CONCLUSION:

This paper demonstrates the use of servo motors for controlling a robotic arm with a dual Arduino setup, where task separation between two Arduino boards enhances performance, enabling smoother and more reliable multi-servo control. The modular approach ensures scalability, allowing easy integration of additional servos and sensors, making it ideal for educational, prototyping, and cost-effective automation applications. The study successfully implemented precise control of multiple SG90 servo motors using the Arduino Uno microcontroller, achieving desired angular motions of 90° and 180° through pulse-width modulation (PWM) signals generated via the Arduino Servo library. The system demonstrated smooth transitions, swift response times, and stable angular positioning, as confirmed by real-time data visualization using the Serial Plotter .The PWM vs. Time and Torque vs. Time graphs confirmed the smooth interpolated sinusoidal patterns of PWM values, ensuring stable and predictable movements of the servo motors. Torque variations followed a similar sinusoidal trend, indicating controlled oscillatory motion, essential for robotic arms, automation, and precision motor control. The system effectively maintained consistent torque output, minimizing erratic movements, ensuring smooth transitions, and reducing mechanical stress. Additionally, the presence of negative torque values suggested bidirectional control, allowing the servos to reverse motion efficiently when required.

A robust hardware setup, including an external power supply and shared grounding, was implemented to mitigate power limitations and signal inconsistencies. The simplicity of the Arduino programming environment enhanced system accessibility, enabling precise and synchronized control of multiple servo motors. The study highlights the system's cost-effectiveness, ease of implementation, and scalability, making it a versatile solution for motion control applications in robotics, automation, and educational tools. For future improvements, integrating real-time sensor feedback, enhancing inverse kinematics algorithms, implementing wireless communication for remote control, and expanding the system for multi-axis servo control could significantly enhance the system’s capabilities. This research establishes a strong foundation for advanced servo motor control systems, contributing to the development of more sophisticated robotic and automation applications.

REFERENCES

[1] Navin Kumar Agrawal, Vinay Kumar Singh, Vinay Singh Parmar, Vijay Kumar Sharma, Dipti Singh, Muskan Agrawal, "Design and Development of IoT based Robotic Arm by using Arduino," Proceedings of the Fourth International Conference on Computing Methodologies and Communication (ICCMC 2020), IEEE Xplore, ISBN: 978-1-72814889-2.

[2] S. C. Suhaimin, N. L. Azmi, M. M. Rahman, H. M. Yusof, "Analysis of Point-to-Point Robotic Arm Control using PID Controller," 2019 7th International Conference on Mechatronics Engineering (ICOM).

[3] Rahul, Hifjur Raheman, Vikas Paradkar, "Design and Development of a 5R 2DOF Parallel Robot Arm for Handling Paper Pot Seedlings in a Vegetable Transplanter," Computers and Electronics in Agriculture, vol. 166, 2019, 105014.

[4] Ikuo Mizuuchi, Masayuki Inaba, Hirochika Inoue, "Adaptive Pick-and-Place Behaviors in a Whole-Body Humanoid Robot with an Autonomous Layer Based on Parallel Sensor-Motor Modules," Robotics and Autonomous Systems, vol. 28, 1999, pp. 99-113.

[5] Ms. Puja Dhepekar, Yashwant G. Adhav, "Wireless Robotic Hand for Remote Operations using Flex Sensor," IEEE, 2016.

[6] Shunli Xiao, Yangmin Li, "Visual Servo Feedback Control of a Novel Large Working Range Micro Manipulation System for Microassembly," Journal of Microelectromechanical Systems, vol. 23, no. 1, February 2014.

[7] S. A. Khan, T. Z. Anika, N. Sultana, F. Hossain, M. N. Uddin, "Color Sorting Robotic Arm," 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST).

[8] Shivam Sharma, Shashwat Sahai, Jaisal Joshi, Hema N, "Stage-wise Development of a Remote Controlled Robotic Arm," 5th IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC-2018), 20-22 Dec, 2018, Solan, India.

[9] Pei-Chi Huang, Aloysius K. Mok, "A Case Study of Cyber-Physical System Design: Autonomous Pick- and-Place Robot," IEEE, 2018.

[10] Reduanur Rahman, Md Sajid Rahman, Jillor Rahman Bhuiyan, "Joystick Controlled Industrial Robotic

System with Robotic Arm," IEEE, 2019.

[11] J. Dias, P. U. Lima, A. Almeida, "Multiple Sensorial Data Fusion for Servo Control of a Robotic Arm," IEEE Transactions on Robotics and Automation, vol. 11, no. 4, Aug. 1995, pp. 508-517.

[12] T. S. Saponara, F. Martinelli, "Embedded DSP-Controlled Servo Actuators for Robotic Applications,"IEEE Transactions on Industrial Electronics, vol. 57, no. 3, March 2010, pp. 906-914.

[13] J. B. Jonsson, P. Marti, S. Uribe, "Real-Time Control of a Servo Motor using Arduino and MATLAB," International Conference on Mechatronics Systems, 2017.

[14] F. Cuesta, C. V. Munoz, D. Casanova, "Servo-Based Positioning Control for Robotic Arms using Arduino,"International Journal of Robotics and Automation, 2021.

[15] A. S. Ramirez, J. L. Tellez, "Design of a Vision-Based Servoing System using Arduino," Mechatronics and Automation Journal, vol. 12, 2020.

[16] Y. Zhao, B. Li, Y. Xu, "Trajectory Control of a Servo Motor-based Robot using Fuzzy Logic," IEEE Transactions on Mechatronics, vol. 24, no. 1, Jan. 2019.

[17] M. K. Jha, S. Mishra, "Microcontroller-Based Servo Motor Control for Robotic Applications,"International Journal of Advanced Robotics, vol. 9, 2018.

[18] H. A. Ali, K. M. Hussein, "Arduino-Based Servo Motor Control for Autonomous Vehicles," IEEE International Symposium on Intelligent Control Systems, 2020.

[19] J. P. Silva, R. R. Gomes, "Integration of Servo Motors with Arduino for Industrial Automation,"Automation and Robotics Conference, 2021.

[20] L. K. Wagner, M. S. Johnson, "Servo-Controlled Mechanisms in Humanoid Robotics," IEEE International Robotics and Automation Conference, 2019.

[21] C. L. Wang, T. R. Huang, "Wireless Servo Motor Control for Smart Home Applications," Journal of Embedded Systems and Applications, 2021.

[22] M. S. Alam, K. Das, S. Roy, "Arduino-Based Servo Control for Assistive Robotics," Journal of Biomedical Engineering and Automation, 2020.

[23] T. Nakamura, H. Tanaka, S. Kobayashi, "Dynamic Servo Control for Bipedal Walking Robots,"International Conference on Advanced Mechatronics Systems, 2018.