e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1580 - 1586

Design and Development of a 4-DOF Robotic Arm for Precision Control

Ayush Daga[1], Aayushi Raj[2], Pavithra C P[3], Sujata R[4]
[1,2,3]Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Chengalpattu, Chennai, Tamil Nadu, India.
[4]Department of Networking and Communications, Faculty of Engineering & Technology, Kattankulathur, Chengalpattu, Chennai, Tamil Nadu, India.
Emails: *dagaayush1205@gmail.com*

Abstract

This project demonstrates the design and development of a robotic arm that is controlled by servo motors and an Arduino microcontroller. The robotic arm has 4 degrees of freedom (DoF), enabling it to make accurate movements and carry out tasks like pick-and-place operations, object handling, and automation. Servo motors are employed to drive the joints of the arm, offering smooth and precise movement. An Arduino board is used as the master controller, interpreting user inputs and carrying out motion commands. The system can be controlled through different interfaces, such as potentiometers, joystick modules, or coded commands. The robotic arm can also be equipped with sensors like ultrasonic or infrared sensors for detecting obstacles, force sensors for feedback control, or a camera module for vision-based object identification. The inclusion of wireless communication modules such as Bluetooth or Wi-Fi facilitates remote control and automation. The project is designed to be cost-effective and scalable for educational, industrial, and research applications in robotics. The modular nature of the arm facilitates future upgrades, including machine learning-based motion planning or AI-based automation. The project illustrates how Arduino-based robotics systems can minimize the gap between low-cost hardware and sophisticated robotic capabilities.

Keywords: *Degrees of Freedom (DoF),*

1. Introduction

Robot arms have transformed industries by improving automation, economy, and accuracy across numerous applications, including manufacturing and surgical robotics. Advancements in technology have sparked increased interest in developing robotic systems that are low-cost and expandable where research, education, and industrial applications would be appropriate. The construction and development of a 4 degree of freedom robotic arm requires the employment of an Arduino microcontroller and servo motors, and all associated mechanical and electronic components needed for their accurate operation. To provide actuation signals, the electronic module may use PLCs, embedded microcontrollers, or more specialized hardware such as FPGAs. Motor drivers act on the signals provided by the electronic module and serve to power the electromechanical actuators, such as servos and stepper motors. These actuators generate torque and are used to move the robot's joints through power transmission mechanisms such as pulleys, timing belts, and gears, and allow for very little backlash. A number of sensors used to monitor the environment [1] and the robot's operating parameters are included in this design and development through the use of position-measuring devices such as encoders, limit switches, and potentiometers [2], which complement the control and adaptability of the robotic arm. Furthermore, power regulators ensure all components operate within the needed voltage range. Robot firmware, implemented on either a PC or embedded microcontroller, is required for processing input, performing motion algorithms, and decision-making in real-time for efficient task performance. The robotic arm is suitable for various automation tasks given its hardware and software contribute to its accurate motion control, environmental interaction, and adaptability. The system is designed to be high-precision, flexible and used primarily for automation, object manipulation, and pick-and-place function. The Arduino board acts as a master controller that receives user input and executes sequences of movements, while having a smooth and precise movement due to the servo-driven joints. The system also includes control interfaces, such as potentiometers and joysticks, and movements can be coded to promote usability. Additional features of the system include the ability to incorporate sensors, such as force sensors for feedback control, ultrasonic and infrared sensors for collision detection, and camera modules for vision-based tasks. Finally, the system is designed to be portable, and wireless modules are available to implement Bluetooth/Wi-Fi functionality for real world and remote control and automated capabilities. The main goal of the project is to develop an open, manageable, flexible and scalable modular robotic system.

2. Literature Survey

Robotic arms have undergone significant evolution over the past few decades, becoming increasingly sophisticated, adaptive, and application-specific. This project focuses on the development and implementation of a **multi-DOF (Degree of Freedom)** robotic arm, emphasizing **4-DOF** articulated kinematics for enhanced workspace and dexterity. Robotic arms can be broadly classified based on their **mechanical structure** (articulated, SCARA, delta, Cartesian), **actuation method** (electric, hydraulic, pneumatic), and **application domain** (industrial automation, surgery, research, etc.). This particular project aligns with **articulated robotic arms**, which use rotary joints to mimic human arm motion, providing flexibility in spatial manipulation. The mechanical design and joint articulation are developed using **SolidWorks** [3], enabling precise CAD modeling and kinematic simulations, while **MATLAB** assists in simulating motion trajectories and solving **inverse kinematics** [4] for accurate end-effector positioning. Control inputs are derived from the **MPU6050** [5], a 6-axis Inertial Measurement Unit (IMU) that combines a **3-axis accelerometer** and a **3-axis gyroscope**. Communication with the control unit is established through the **I²C (Inter-Integrated Circuit)** protocol, which supports multi-device chaining on a single bus, allowing integration of additional IMUs or sensors. An **ATmega328P** [6] microcontroller (as used in **Arduino Uno**) serves as the central processing unit. It processes sensor data, computes control signals, and interfaces with **servo motors**, which are used to actuate the robotic arm joints. Servo motors are particularly advantageous due to their **built-in feedback mechanism** (via potentiometers), **precise angle control**, and compatibility with **PWM (Pulse Width Modulation)** signals. The motion of each joint is regulated based on sensor feedback and pre-programmed trajectories, ensuring smooth and stable operation. Furthermore, **motor drivers** (such as L293D [7] or TB6612FNG) bridge the logic-level control signals from the ATmega328P to the power-hungry motors, ensuring sufficient current and voltage handling. For more intelligent motion, **PID (Proportional–Integral–Derivative)** controllers may be implemented in firmware to stabilize joint movement and reject external disturbances. Future enhancements could involve integrating **servo encoders**, **ROS** [8] **(Robot Operating System)** for higher-level task execution, or even **machine learning models** for adaptive manipulation and autonomous behavior.

3. Implementation

The implementation of the gesture-controlled inverse kinematics (IK) system is executed using Zephyr RTOS [9] on a Blackpill STM32 microcontroller [10]. The system employs a modular, multithreaded architecture to handle concurrent tasks including gesture acquisition, data fusion, IK computation, and servo actuation.

3.1. Inverse Kinematics- Concept Overview

Inverse Kinematics (IK) is a foundational concept in robotics and motion control, concerned with determining the required joint configurations to achieve a specific end-effector position and orientation in space. Unlike **Forward Kinematics (FK)**, which involves computing the position of the end-effector based on known joint parameters, IK operates in reverse — computing joint parameters from a desired spatial position. In robotic arms, IK is essential for achieving precise and goal-directed motion, especially in tasks where the position of the

e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1580 - 1586

end-effector must be controlled to interact with the environment. For arms with revolute joints, IK typically involves calculating joint angles; for prismatic joints, like in the system described in this paper, the goal is to determine the required linear displacements of each joint.

3.1.1. Mathematical Foundation

Let the robotic arm's end-effector position be defined as a vector:

$$\mathbf{P}_{target} = [x, y, z, f]^T$$

where x, y, z represent spatial coordinates and f represents the extension or fine adjustment at the tool-end. The IK function f−1f^{-1}f−1 maps this spatial coordinate to joint values:

$$[q_1, q_2, q_3, q_4] = f^{-1}(\mathbf{P}_{target})$$

where each qiq_iqi is the required displacement of a prismatic joint to reach the corresponding axis-aligned position. Given the linear nature of prismatic joints, the inverse kinematics formulation becomes more straightforward compared to systems with multiple degrees of rotational freedom. Each axis displacement can directly correspond to a joint's linear extension:

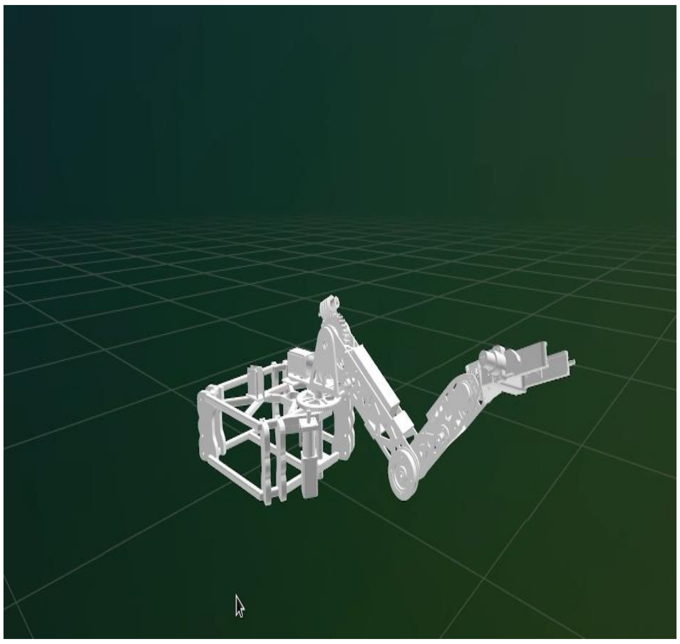


Figure 1 Multiple Degrees of Rotational Freedom

- $q_1 = x$ (base sliding along X-axis),
- $q_2 = y$ (vertical translation),
- $q_3 = z$ (depth movement),
- $q_4 = f$ (end-effector fine positioning).

3.1.2. Relevance to Gesture-Controlled Systems

In the context of this system, IK acts as a bridge between human intention and robotic execution. Hand gestures captured via the IMU and camera are processed to estimate target end-effector coordinates. These are then passed to the IK model, which computes the appropriate joint displacements required to mimic or follow the user's hand. This real-time mapping enables:

- Natural and intuitive robot control,
- Elimination of traditional joysticks or input devices,
- Human-in-the-loop interaction for robotic arms in assistive or collaborative scenarios.

3.1.3. Advantages in a Prismatic-Only Configuration

The choice of using prismatic joints simplifies IK computation:

- Linear relationships avoid trigonometric complexity.
- Reduced computational overhead suits microcontroller-based implementation (STM32).
- Enhances predictability and determinism in control logic.

By embedding the IK solver within a real-time operating system (Zephyr RTOS), the system ensures rapid responsiveness, deterministic timing, and robust handling of concurrent sensor inputs and motor outputs.

3.2. System Architecture and Task Scheduling

Zephyr RTOS is configured with a preemptive multitasking kernel. Key tasks are separated into individual threads with appropriate priorities:

- **High Priority**: IMU sensor data acquisition and camera gesture tracking.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2025.0256

e ISSN: 2584-2854
Volume: 03
Issue: 05 May 2025
Page No: 1580 - 1586

- **Medium Priority**: Sensor fusion and IK computation.
- **Low Priority**: PWM signal generation for motor control and status telemetry.

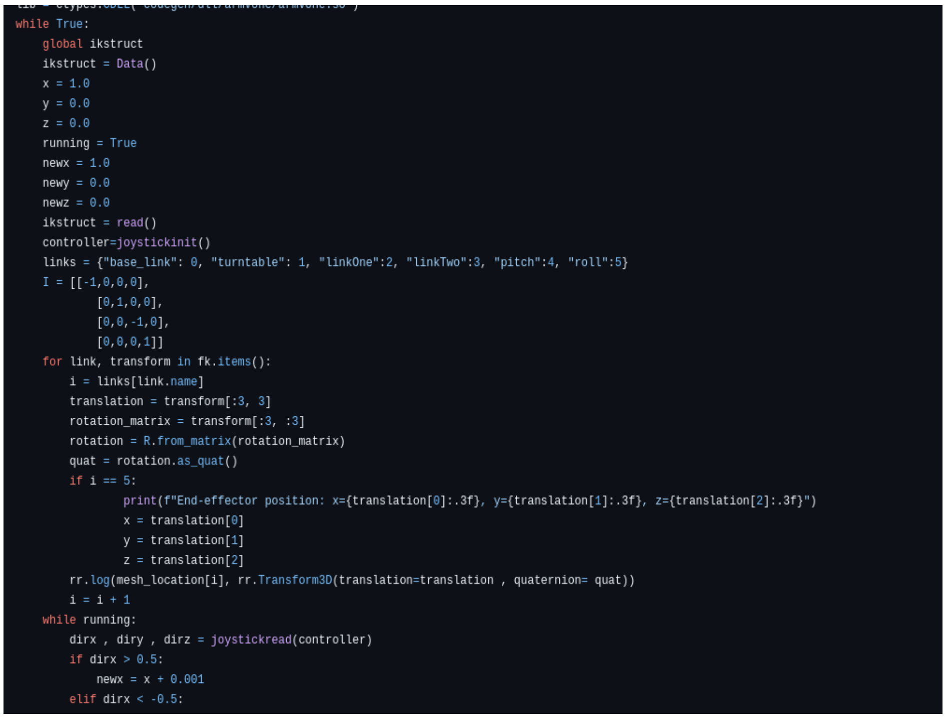
Each component is mapped to Zephyr-compatible device drivers and managed through the RTOS scheduler for deterministic performance.

3.3. Gesture Acquisition

3.3.1. IMU-Based Motion Tracking

The IMU (e.g., MPU6050 or BNO055) is interfaced via I2C to the STM32. Data acquisition is performed at 100 Hz (Figure 2).

- Euler angles (pitch, roll, yaw) and acceleration vectors are used to infer:
  - **Pitch** → Y-axis control (up/down movement)
  - **Roll** → X-axis control (lateral movement)
  - **Vertical Acceleration** → Z-axis extension/retraction
- A complementary filter is used to reduce sensor drift and smoothen the signal.



```
while True:
    global ikstruct
    ikstruct = Data()
    x = 1.0
    y = 0.0
    z = 0.0
    running = True
    newx = 1.0
    newy = 0.0
    newz = 0.0
    ikstruct = read()
    controller=joystickinit()
    links = {"base_link": 0, "turntable": 1, "linkOne":2, "linkTwo":3, "pitch":4, "roll":5}
    I = [[-1,0,0,0],
          [0,1,0,0],
          [0,0,-1,0],
          [0,0,0,1]]
    for link, transform in fk.items():
        i = links[link.name]
        translation = transform[:3, 3]
        rotation_matrix = transform[:3, :3]
        rotation = R.from_matrix(rotation_matrix)
        quat = rotation.as_quat()
        if i == 5:
                print(f"End-effector position: x={translation[0]:.3f}, y={translation[1]:.3f}, z={translation[2]:.3f}")
                x = translation[0]
                y = translation[1]
                z = translation[2]
        rr.log(mesh_location[i], rr.Transform3D(translation=translation , quaternion= quat))
        i = i + 1
    while running:
        dirx , diry , dirz = joystickread(controller)
        if dirx > 0.5:
            newx = x + 0.001
        elif dirx < -0.5:
```

Figure 2 Coding

3.3.2. Camera-Based Hand Position Tracking

- A forward-facing camera (e.g., OV7670 or external module via UART/SPI) captures real-time images.
- Simple gesture recognition (e.g., hand position in frame, pinch, swipe) is performed using edge processing or pre-trained models if connected via a companion device.
- X and Y positions from the frame are mapped to a 2D plane, assisting in absolute target localization.

3.3.3. Sensor Fusion and Gesture Interpretation

A dedicated fusion thread normalizes and merges data from both the IMU and camera. The fusion algorithm:

- Uses camera data to estimate hand position in the horizontal plane.
- Uses IMU to refine fine movements and detect gesture orientation.
- Applies a transformation matrix or lookup table to convert these readings into workspace coordinates: Target Position (x, y, z, f).

Capping mechanisms are applied to ensure the output remains within joint movement constraints, preventing overextension or collision.

3.4. Inverse Kinematics and Mapping

As the robotic arm consists entirely of prismatic joints, the IK problem reduces to direct linear mapping:

- Joint 1 (Base - X) ← Normalized roll from IMU
- Joint 2 (Vertical - Y) ← Normalized pitch from IMU
- Joint 3 (Extension - Z) ← Forward/back hand motion from camera depth estimate
- Joint 4 (End-effector - F) ← Gesture-defined fine control (e.g., finger pinch or acceleration tap)

Each desired position is linearly scaled to the servo range, using calibrated limits from the startup phase.

3.5. PWM Control and Actuation

- Zephyr's native PWM driver APIs are used to control SG90 and MGR servos.
- Pulse widths are mapped linearly to millimeter displacements of each prismatic joint using calibrated lookup tables.
- SG90 servos are driven directly using 3.3V logic-level PWM signals.
- MGR servos are interfaced via MOSFET drivers and powered from a dedicated 5V/12V rail due to higher torque requirements.

3.6. Calibration and Initialization

On power-up, a calibration routine captures: IMU minimum and maximum angles during a predefined motion sweep. Camera's field of view boundary to correlate screen space to arm space. These bounds are used to normalize gesture input into control signals dynamically.

4. Hardware Requirements

4.1. Robotic Arm & Actuators

- SG90 Servo Motors (x2)
  - ➢ Used for light-load prismatic joints
- MGR High Torque Servo Motors (x2)
  - ➢ Used where stronger actuation is required (e.g., base or vertical axis)

4.2. Mechanical Components

- Linear Prismatic Joint Mechanism
- 3D Printed design and model for arm 3D printed or CNC-based slider mechanisms with servo coupling for linear motion
- Linkages with rods and sliders

4.3. Control & Processing

- STM32F411 Blackpill Development Board 32-bit ARM Cortex-M4 Runs Zephyr RTOS
- Voltage Regulator (e.g., AMS1117 or LM2596) Converts input power to 5V for servos and logic

4.4. Sensor Modules

- MPU6050 or BNO055 IMU Sensor Captures 3D orientation and acceleration of the hand
- OV7670 / ESP32-CAM / OpenMV Camera for hand gesture and position tracking

4.5. Power & Connectivity

- 12V Power Adapter / Li-ion Battery Pack Drives MGR and SG90 servos
- Level Shifter / Logic Buffer If camera/IMU use 3.3V while servos need 5V logic
- ESP8266/ESP32 or NRF24L01 Modules for wireless communication if the gesture detection is processed remotely

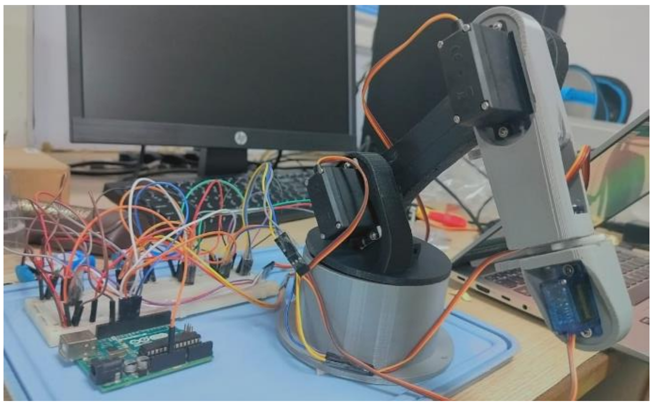


Figure 3 **Hardware Requirements**

4.6. Safety and Error Handling

- Timeout watchdogs prevent lock-ups from stalled sensors.
- Range limiters stop the servos if input exceeds physical boundaries.
- Manual override via Flysky transmitter is available for emergency control.

Conclusion

This project successfully demonstrates the design and implementation of a 4-degree-of-freedom (DoF) robotic arm, controlled via servo motors and an Arduino microcontroller, with gesture-based inverse kinematics (IK) for intuitive, real-time user interaction. The system leverages both IMU and camera-based gesture recognition to map human hand movements directly to the robotic arm's prismatic joints, enabling natural and precise control without traditional input devices. The modular, scalable architecture—incorporating Zephyr RTOS for multitasking and robust sensor fusion—ensures deterministic, responsive operation suitable for automation, pick-and-place, and object manipulation tasks. The integration of low-cost hardware, open-source software, and flexible interfaces highlights the project's potential for educational, research, and industrial applications, bridging the gap between affordability and advanced robotic capabilities

Future Scope

Several avenues exist for enhancing the system's capabilities and expanding its applicability:

- Integration of Advanced Sensors: Incorporating additional sensors such as force/torque sensors, high-resolution encoders, or advanced vision modules can improve feedback, enable more precise manipulation, and facilitate complex tasks like adaptive grasping or force-controlled assembly.
- Wireless and Cloud Connectivity: Expanding wireless control with Bluetooth, Wi-Fi, or IoT platforms would allow for remote operation, cloud-based monitoring, and collaborative robotics scenarios.
- Machine Learning and AI: Implementing machine learning algorithms for gesture recognition, motion planning, or adaptive control can enable the arm to learn from user behavior, optimize its movements, and perform autonomous or semi-autonomous tasks.
- Integration with ROS: Adopting the Robot Operating System (ROS) framework would facilitate higher-level task execution, interoperability with other robotic platforms, and access to a vast ecosystem of robotics libraries.
- Enhanced User Interfaces: Developing more sophisticated user interfaces, such as haptic feedback gloves or AR/VR-based control panels, could further improve intuitiveness and user engagement.
- Industrial and Assistive Applications: Customizing the system for specific domains—such as industrial automation, medical assistance, or collaborative human-robot environments—could broaden its real-world impact.
- Mechanical Upgrades: Exploring alternative actuation methods (e.g., stepper motors, pneumatic actuators) or hybrid joint configurations (combining prismatic and revolute joints) may increase the arm's dexterity, payload, and workspace.

By pursuing these enhancements, the project can evolve into a more intelligent, adaptive, and widely applicable robotic platform, supporting ongoing advancements in automation, human-robot interaction, and accessible robotics research.

References

[1]. A.N. Joseph Raj, R. Sundaram, V.G. Mahesh, Z. Zhuang, A. Simeone A multi-sensor system for silkworm cocoon gender classification via image processing and support vector machine Sensors., 19 (12) (2019), p. 2656.

[2]. W. Chen, H. Khamis, I. Birznieks, N.F. Lepora, S.J. Redmond Tactile sensors for friction estimation and incipient slip detection - toward dexterous robotic manipulation: a review IEEE Sensors J., 18 (22) (2018), pp. 9049-9064.

[3]. R. Sam, K. Arrifin and N. Buniyamin, "Simulation of pick and place robotics system using Solidworks Softmotion," 2012 International Conference on System Engineering and Technology (ICSET), Bandung, Indonesia, 2012, pp. 1-6, doi: 10.1109/ICSEngT.2012.6339325.

[4]. R. P. Paul and B. Shimano, "Kinematic control equations for simple manipulators,"

1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes, San Diego, CA, USA, 1978, pp. 1398-1406, doi: 10.1109/CDC.1978.268148.

[5]. TDK InvenSense, "MPU-6000 and MPU-6050 Product Specification Revision 3.4," Datasheet, 2013. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

[6]. Microchip Technology Inc., "ATmega328/P: 8-bit AVR Microcontroller with 32KB Flash, 20MHz, 1.8-5.5V," Datasheet, [Online]. Available: https://www.microchip.com/en-us/product/ATmega328P.

[7]. Texas Instruments, "L293D Quadruple Half-H Drivers," Datasheet, [Online]. Available: https://www.ti.com/product/L293D.

[8]. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA) Workshop Open Source Softw., Kobe, Japan, May 2009.

[9]. The Zephyr Project, "The Zephyr Project: Scalable Open Source RTOS for IoT Embedded Devices," The Linux Foundation, White Paper, 2019. [Online]. Available: https://www.zephyrproject.org

[10]. STMicroelectronics, "STM32F103x8/xB: Arm® Cortex®-M3 32-bit MCU+FPU, up to 128KB Flash, 72MHz," Datasheet, 2023. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html.