STM JOURNALS

Journal of
Aerospace Engineering & Technology

ISSN: 2231-038X (Online)
ISSN: 2348-7887 (Print)
Volume 15, Issue 2, 2025
May–August
DOI (Journal): 10.37591/JoAET

https://journals.stmjournals.com/joaet

Review JoAET

IoT-Enabled and Hand Gesture Controlled Robotic Arm Using Blynk IoT and OpenCV

Ruchira Pagar[1], Yogesh Bhangale[1], Bhaven Bankar[1,*], T.V. Kafare[2]

Abstract

Robotic arms have become vital in environments that necessitate control, stability, and remote operation. This study presents a low-cost, multifunctional robotic arm that operates in four distinct modes: gesture control using OpenCV, IoT-based control via the Blynk platform and ESP32, joystick-based manual control, and an automatic mode driven by coordinates stored on an SD card. The system integrates multiple control mechanisms through the ESP32 microcontroller and utilizes servo motors for precise actuation. Mode switching is enabled via push buttons, while MQTT protocol ensures reliable and responsive communication across components. This modular and adaptable design makes the system suitable for industrial, assistive applications. It also supports remote-controlled operations, making it suitable for simple remote pick-and-place tasks where high precision is not required.

Keywords: ESP32, Blynk IoT, robotic arm, MQTT, Servo motor, joystick, coordinates, gesture

INTRODUCTION

In recent years, the application of robotic arms has expanded beyond high-precision manufacturing into more cost-effective domains where basic automation can improve efficiency and reduce human involvement. While traditional robotic arms are engineered for precision, stability, and complex motion control, there is a growing demand for simpler systems capable of performing fundamental tasks like picking and placing lightweight objects. Such systems are particularly useful in small-scale industrial settings, educational institutions, and assistive technology, where high accuracy is not critical but functionality and affordability are essential.

This study presents a low-cost, multi-mode robotic arm designed for general-purpose use across a variety of basic automation scenarios. The system supports four modes of operation: automatic mode using preloaded coordinates from an SD card, gesture control using OpenCV for real-time hand movement detection, IoT-based control through the Blynk platform and ESP32 microcontroller, and joystick-based manual control. By integrating these diverse control methods, the robotic arm provides operational flexibility and adaptability while remaining budget-friendly and relatively simple to assemble.

***Author for Correspondence**
Bhaven Bankar
E-mail: bhavenbankar.scoe.entc@gmail.com

[1]Student, Department of Electronics and Telecommunication Engineering, Sinhgad College of Engineering, Vadgaon (BK), Pune, Maharashtra, India
[2]Assistant Professor, Department of Electronics and Telecommunication Engineering, Sinhgad College of Engineering, Vadgaon (BK), Pune, Maharashtra, India

Receiving Date: April 15, 2025
Accepted Date: May 07, 2025
Published Date: May 22, 2025

Citation: Ruchira Pagar, Yogesh Bhangale, Bhaven Bankar, T.V. Kafare. IoT-Enabled and Hand Gesture Controlled Robotic Arm Using Blynk IoT and OpenCV. Journal of Aerospace Engineering & Technology. 2025; 15(2): 36–46p.

In automatic mode, the robotic arm executes a series of predefined movements by reading coordinate data from an SD card, making it suitable for repetitive tasks in structured environments like sorting or assembly lines. Gesture control is enabled through computer vision techniques using OpenCV, where a connected camera detects basic hand gestures and maps them to corresponding arm movements. This mode is particularly advantageous in scenarios requiring contactless control, such as healthcare or sterile environments. The IoT control mode allows users to operate the robotic arm

remotely via a smartphone or web interface using the Blynk platform, which is especially beneficial in remote monitoring or supervision applications. The joystick control mode provides a direct and intuitive form of operation, ideal for real-time control during testing or manual intervention.

The ESP32 microcontroller serves as the core processing unit, coordinating commands from all modes and driving the servo motors (SG90) responsible for the arm's movements. Push buttons are used to switch between modes, and the MQTT protocol is integrated to ensure fast and reliable communication, particularly in the IoT control context. Overall, the proposed system emphasizes modularity, affordability, and ease of integration, making it well-suited for environments that require basic automation without the need for high precision or complex robotics.

Overview

Aiming to simplify basic automation, this project features a low-cost robotic arm with four control modes: automatic (via SD card), gesture-based (using OpenCV), IoT-enabled (through Blynk and ESP32), and manual joystick control. Utilizing SG90 servo motors and MQTT communication, it offers functional versatility for pick-and-place tasks in educational, assistive, and small-scale industrial settings, without focusing on high precision.

This study has been organized as: The first Section describes the introduction and the main purpose of this project; the Section after that consist of previous related work and literature review. The Section next to that consists of methodology, working principle, hardware components used and software tools and algorithms used. This is followed by the Section consisting of result and discussions and the last Section consists of conclusion where all the objectives of the project are satisfied.

RELATED WORK

Several studies have explored the use of hand gestures and sensors to control robotic arms.

- Sihombing *et al.* developed a system where finger and hand movements directly control the robotic arm using wearable sensors [1].
- Similarly, Saleheen *et al*. implemented a gesture-based robotic arm using flex sensors for real-time control [2].
- Paterson and Aldabbagh used OpenCV to detect hand gestures via a camera to control the arm, eliminating the need for wearable devices [3].
- On the IoT side, Singh *et al*. introduced a Wi-Fi-controlled robotic arm on a rover platform using NodeMCU [4].
- Janagiraman *et al*. researched in combined gesture control and IoT to enhance industrial automation [5].
- For home and elderly care, Alshdadi designed a smart home robotic arm system [6].
- In the field of gesture recognition, Chai *et al*. focused on translating sign language using Kinect sensors [7].
- A study by Lu *et al*. improved hand tracking by using multiple visual inputs for better gesture recognition [8].

METHODOLOGY

The development of the robotic arm system followed a modular and multi-mode integration approach. The ESP32 microcontroller served as the central unit, interfacing with various input modules such as a camera for gesture detection, a joystick for manual control, an SD card for automatic sequences, and the Blynk IoT platform for remote operation. Each mode was programmed individually and then integrated into a unified system using a push-button interface for mode switching. Servo motors were calibrated to respond to real-time signals received through PWM outputs [9, 10]. Communication protocols like MQTT were implemented for wireless gesture data transfer, ensuring smooth interaction between hardware and software components. This structured methodology enabled the creation of a flexible, low-cost robotic arm capable of functioning under different control conditions. Furthermore, Figure 1 shows the block diagram of the model.

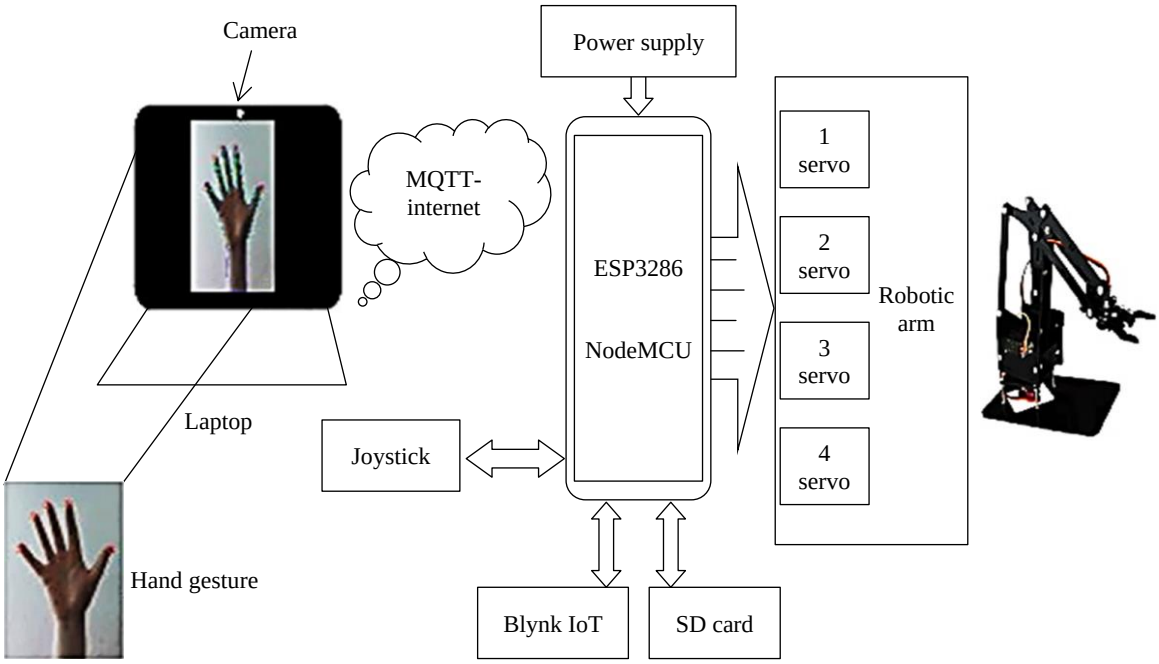

Figure 1. Block diagram.

Working Principle

The working of the system is kept as simple as possible. A push button is used to select the desired mode. In gesture mode, a camera captures hand movements, which are processed by a laptop using OpenCV and MediaPipe, then sent wirelessly to the ESP32 using MQTT. In IoT mode, the user sends commands from a smartphone through the Blynk app, which are received by the ESP32 via Wi-Fi. In joystick mode, the ESP32 reads signals directly from a physical joystick. In automatic mode, the ESP32 reads predefined movement data from an SD card. Based on the selected input, the ESP32 sends signals to servo motors to move the robotic arm accordingly.

Working of Different Modes

Gesture-Based Control Mode (Using OpenCV)

In gesture-based mode, the robotic arm is controlled by interpreting the user's hand gestures via a camera connected to a computer running OpenCV for gesture recognition. This allows for an intuitive and contactless control mechanism, making it suitable for applications such as assistive robotics and interactive automation.

Workflow of Gesture-Based Control Mode

1. *Gesture capture:* A camera captures the live video feed of the user's hand, continuously tracking movements and positions.
2. *Gesture recognition via OpenCV:* The OpenCV library processes the video feed in real time, detecting specific hand gestures such as swipe left, swipe right, open hand, and closed fist using contour detection and machine learning-based classification.
3. *Command generation:* Once a gesture is recognized, the computer interprets it and generates corresponding commands such as rotate arm left, move up, or close gripper to control the robotic arm's movements.
4. *Data transmission via MQTT:* The control commands are then transmitted from the computer to the NodeMCU using the MQTT (Message Queuing Telemetry Transport) protocol, ensuring fast, low-latency communication.
5. *Actuation and execution*: The NodeMCU receives the commands and converts them into appropriate PWM signals to drive the servo motors, enabling the robotic arm to execute precise movements in response to hand gestures.

6. *IoT mode (BLYNK IoT App):* In IoT mode, the user can control the robotic arm remotely via the Blynk IoT application on a smartphone or tablet. This enables real-time control of the robotic arm from anywhere with an internet connection, making it highly flexible for various applications.

Workflow of IoT Mode

1. *User access and initialization*: The user opens the Blynk app, which is connected to the Blynk Cloud to establish a remote communication link.
2. *Command input*: Through the app interface, the user sends movement commands such as move left, right, up, and down using interactive controls like sliders, buttons, and joysticks.
3. *Data transmission via Blynk cloud*: The Blynk Cloud forwards the user's control commands to the NodeMCU over Wi-Fi, ensuring seamless wireless communication.
4. *Processing and actuation*: The NodeMCU microcontroller processes the received commands and generates corresponding PWM signals to servo motors, controlling the robotic arm's movement.
5. *Real-time execution*: The robotic arm executes the instructed movements in real time, providing instant feedback to the user through the app interface.
6. *Automatic mode:* Automatic mode enables the robotic arm to function independently, following a predefined set of movements or instructions without requiring external input. This mode is particularly useful in industrial automation, repetitive tasks, and applications where human intervention is minimal or unnecessary. The robotic arm's movements are pre-programmed into the microcontroller.

Workflow of Automatic Mode

1. *Coordinate preparation*: A sequence of servo movement positions (angles) is first written and saved in a .txt. These coordinates define how the robotic arm should move to perform a task like picking and placing.
2. *Loading to SD card*: The file containing the coordinates is transferred to a microSD card using a computer. Once saved, the SD card is inserted into the SD card adapter, which is connected to the ESP32 microcontroller.
3. *Reading the data:* When automatic mode is activated through the push-button interface, the ESP32 starts reading the file from the SD card. It reads the data line by line using the SPI communication protocol.
4. *Processing the coordinates*: The ESP32 parses each line of coordinates and translates the values into PWM signals required to control the servo motors.
5. *Executing the movements*: The PWM signals are sent to the SG90 servo motors, causing the robotic arm to move according to the predefined instructions: performing actions such as gripping, lifting, rotating, and placing.
6. *Repeating the cycle*: Once all the coordinates have been read and the final motion is executed, the ESP32 restarts the sequence from the beginning of the file. This allows the robotic arm to continuously repeat the same pick-and-place cycle until the mode is changed or power is turned off.

Flowchart

This flowchart in Figure 2 illustrates three control modes for the robotic arm: automatic (predefined movements), IoT-based (Blynk app), and gesture control (camera + OpenCV). Each mode processes user input differently, sends commands to the NodeMCU, and actuates servo motors to perform desired tasks, with optional sensor feedback for automation and repetition.

HARDWARE TOOLS/COMPONENTS

ESP32

The ESP32 is a powerful and cost-effective microcontroller developed by Espressif Systems (Figure 3). It features dual-core Tensilica LX6 processors, operates at a frequency of up to 240 MHz, and includes 520 kb of SRAM, making it ideal for complex embedded systems. It comes with integrated Wi-Fi and Bluetooth capabilities, which makes it perfect for IoT applications requiring wireless communication. The ESP32 typically has 38 pins, including 30 GPIO pins that support PWM.

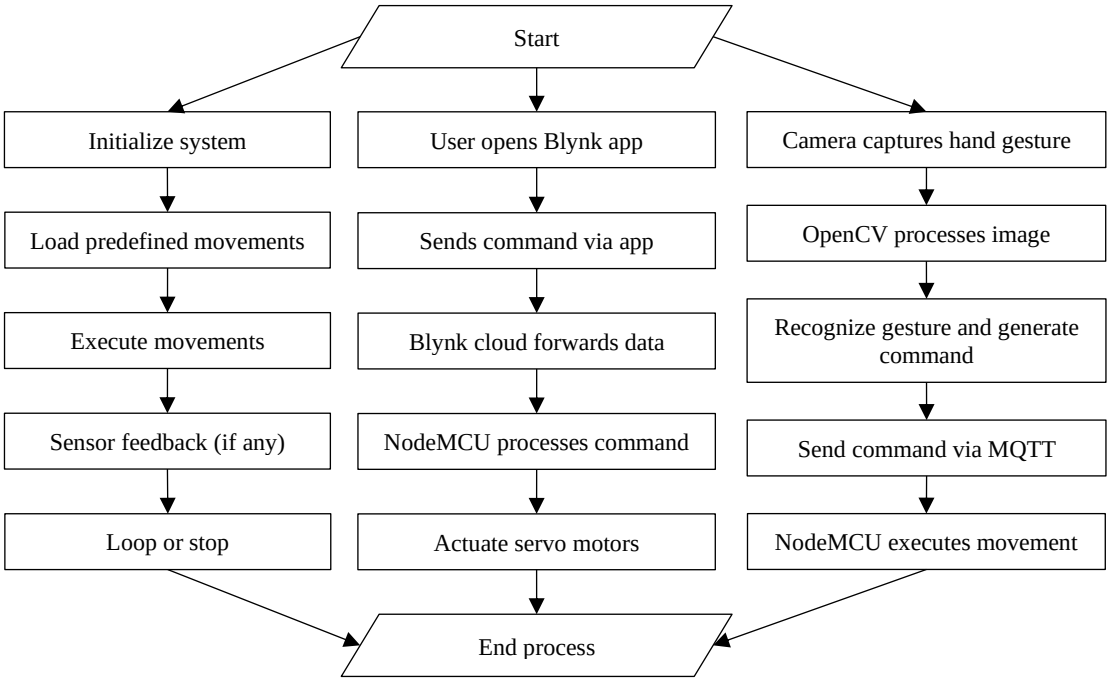

Figure 2. Structural Representation of working of different modes.

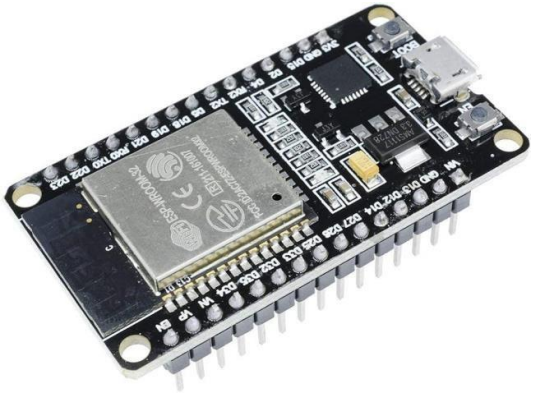

Figure 3. ESP32.

Figure 4. SG90 servo motor.

In this project, the ESP32 acts as the central controller, processing input from various control modes, such as gesture data from a laptop via MQTT, remote commands from the Blynk IoT platform, joystick signals, and preloaded instructions from an SD card. It generates PWM signals to control up to four SG90 servo motors that drive the robotic arm. Additionally, its low power consumption and ability to handle multiple tasks simultaneously make it well-suited for real-time robotic applications. The ESP32 can also be programmed using the Arduino IDE providing flexibility in development.

SG90 Servo Motor

The SG90 servo motor is a compact, lightweight, and affordable servo commonly used in hobby electronics and robotic applications (Figure 4). It operates on a voltage range of 4.8 to 6 V, with a stall torque of around 1.8 kg.cm at 6 V and a rotation range of approximately 0 to 180°. The SG90 receives PWM (Pulse Width Modulation) signals to control its angle, making it ideal for precise positioning tasks. This servo features three pins: VCC (Power), GND (Ground), and Signal (PWM input). Despite its small size, it delivers reliable performance for light-duty tasks, such as actuating robotic joints or moving arms. Internally, it includes a small DC motor, a gear train, and a feedback potentiometer that allows accurate angle control. In this project, four SG90 servo motors are used to control the robotic arm's various joints, enabling movements like lifting, rotating, and gripping.

SOFTWARE TOOLS AND ALGORITHMS

Tools

Arduino IDE

Arduino IDE is an open-source software platform primarily used for writing and uploading code to microcontroller boards like the ESP32. It supports both C and C++ programming and provides a user-friendly environment ideal for both beginners and experienced developers. The IDE includes features such as a serial monitor for debugging, a library manager for installing external modules, and a board manager to add support for a wide range of microcontrollers. In this project, the Arduino IDE is used to program the ESP32 to control servo motors, handle various input modes like joystick, SD card, and Blynk IoT, and manage communication protocols such as MQTT.

OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful library for real-time computer vision and image processing tasks. It allows developers to perform operations like face detection, object recognition, motion tracking, and more. In this robotic arm project, OpenCV is used to process video input from a webcam to detect hand gestures. By analyzing frames in real time, OpenCV extracts visual features which are further interpreted with the help of MediaPipe to recognize hand positions and movements.

MediaPipe

MediaPipe, developed by Google, is a cross-platform framework designed for building efficient perception pipelines. It provides pre-trained models for facial detection, pose estimation, and hand tracking. In your project, MediaPipe is responsible for detecting 21 landmark points on the hand, enabling accurate gesture recognition. These detected gestures are then translated into control signals that guide the robotic arm's movement. Its real-time processing and compatibility with OpenCV and Python make it ideal for gesture-controlled systems.

Blynk IoT App

Blynk IoT Platform is a user-friendly cloud platform that enables the remote control and monitoring of IoT devices using smartphones or web applications (Figure 5). It allows users to create custom interfaces using drag-and-drop widgets like buttons, sliders, and gauges. For this project, Blynk is integrated with the ESP32, allowing users to control the robotic arm remotely through Wi-Fi. The commands sent from the Blynk app are received by the ESP32 and translated into specific movements, enabling remote and real-time operation of the arm from any location.

Remote

The custom remote interface is created in the Blynk IoT app to control a robotic arm using the ESP32 microcontroller. The layout includes four vertical and horizontal sliders, each assigned to control one servo motor of the robotic arm, allowing precise adjustment of joint angles. Additionally, two large circular buttons at the bottom are configured as joystick controls to facilitate directional movements or additional functionality like gripper control or mode switching. The user can interact with this interface from a smartphone, sending real-time commands to the ESP32 via the Blynk cloud, enabling wireless and remote control of the robotic arm.

Algorithms

Gesture-Based Servo Control Algorithm Using OpenCV and Mediapipe

1. Initialize serial communication with the ESP32 and setup Mediapipe for hand tracking.
2. Start webcam to capture real-time video frames.
3. Detect hand landmarks using Mediapipe on each frame.
4. Check for left hand, and extract the tip and base landmarks of ring, index, middle, and little fingers.
5. Calculate distances between tip and base of each finger and map them to servo angles (0–180°).
6. Send servo angles as JSON data to ESP32 via serial communication.
7. Display the angles on the video feed and update continuously until 'q' is pressed.
8. Release resources and close the program cleanly.

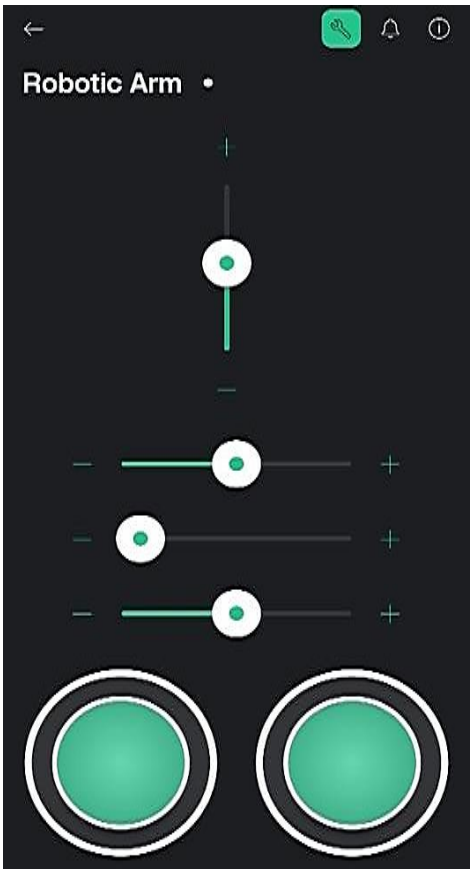

Figure 5. Robotic arm remote in Blynk.

Robotic Arm Mode Switching Algorithm

Initialize Wi-Fi for Blynk, servos, LCD, SD card, and serial communication.

1. Detect the control mode selected by the user through a switch: Joystick, Blynk, Gesture, or Automatic.
2. In Joystick Mode, read analog inputs from the joystick, then map these values to servo angles and move the servos accordingly.
3. In Blynk Mode, receive servo values from the mobile app sliders and then control the servos based on those values.
4. In Gesture Mode, receive JSON data from the Python program performing hand tracking, and then update servo angles based on that data.
5. In Automatic Mode, read pre-stored servo positions from the SD card and then execute those predefined movements.
6. Continuously check for mode selection and repeat the respective steps based on the current user input.

RESULTS

In this section, the hardware results of the gesture controlled robotic hand are presented. The circuit diagram illustrates the complete hardware setup for a multi-mode robotic arm controlled via ESP32 (Figure 6). The ESP32 microcontroller serves as the central controller, interfacing with four servo motors (Servo01 to Servo04) that drive the robotic arm's movements. A joystick module labeled "REMOTE" is connected for manual control using analog X and Y axes and push buttons. A mode selection switch ("MODES") allows toggling between different operational modes: Joystick, Blynk (IoT app), Gesture, and Automatic. An LCD display is used for mode indication and feedback, while an SD card module is included to store and retrieve predefined servo positions for automatic operation. The layout ensures proper power connections (VCC and GND) and logical signal routing, providing a flexible and user-friendly robotic arm control system. Figure 7 shows the 3D PCB design.

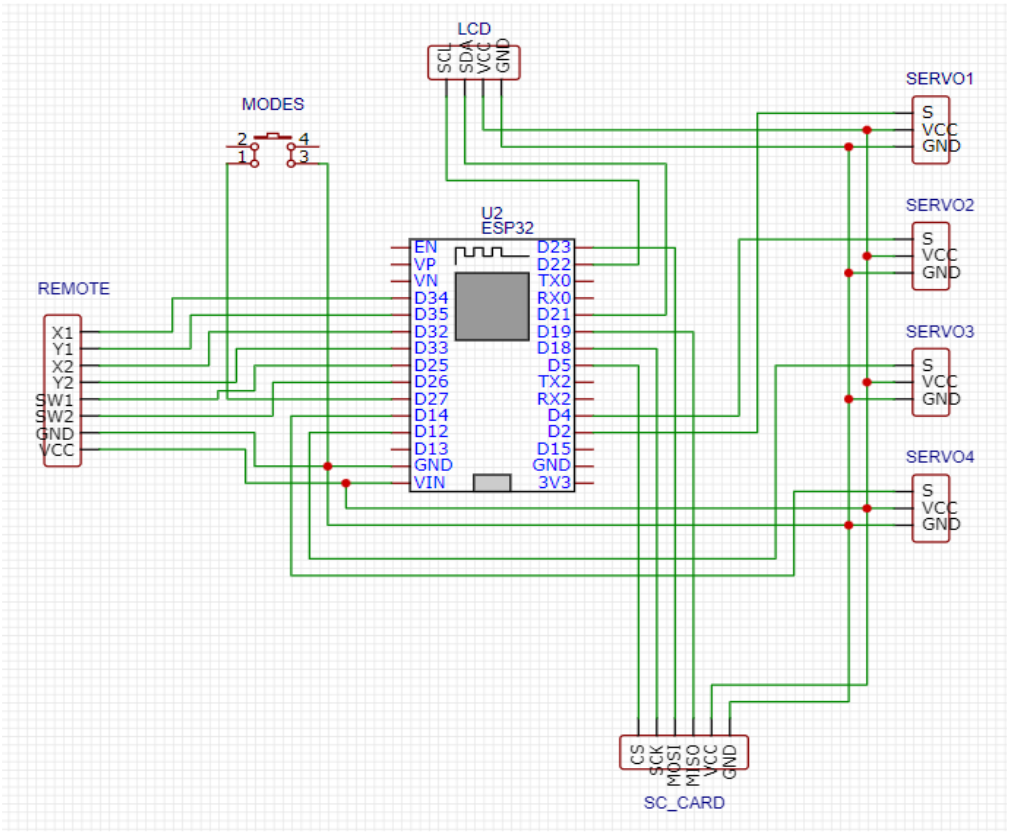

Figure 6. Circuit diagram of proposed system.

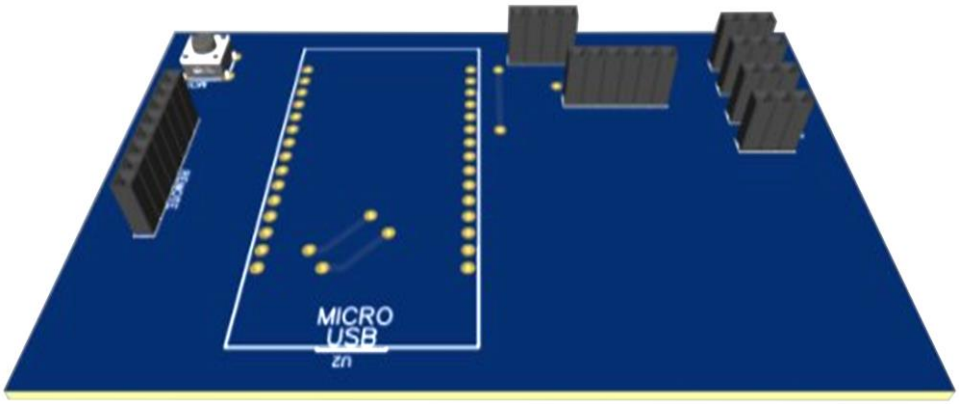

Figure 7. 3D PCB design.

The image captures the real-time hand gesture detection process implemented using Python and OpenCV with the help of the MediaPipe framework (Figure 8). The red dots and lines on the hand represent the landmark points detected by the algorithm, which are used to analyze finger positions and orientations. This data is then converted into pseudo servo positions, which simulate the servo angles that would be sent to the robotic arm for movement. As shown in the terminal output, the system continuously prints the calculated servo positions, in this case, each joint corresponds to an angle of 144°, demonstrating how hand movements are translated into angular values to control the robotic arm's servos in gesture mode (Figure 9).

The robotic arm in this project employs SG90 servo motors, which are lightweight and cost-effective micro servos capable of producing a torque of approximately 2.5 kg.cm at 5 V. With this torque specification, each servo can theoretically lift around 250 gm when the load is positioned 1 cm from the axis.

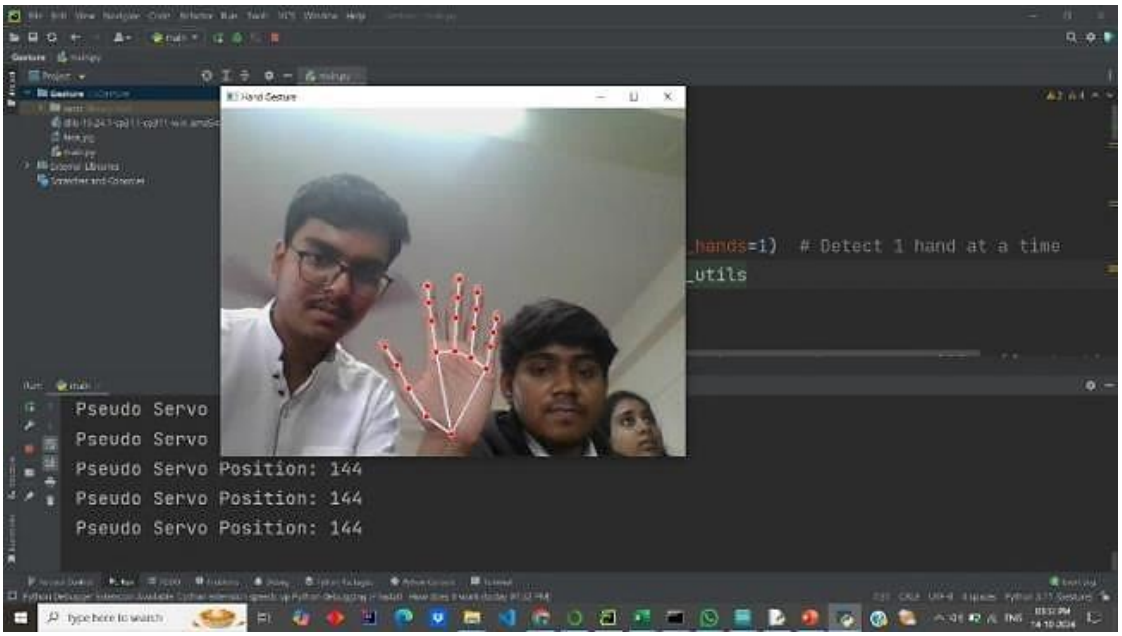

Figure 8. Hand gesture using OpenCV.

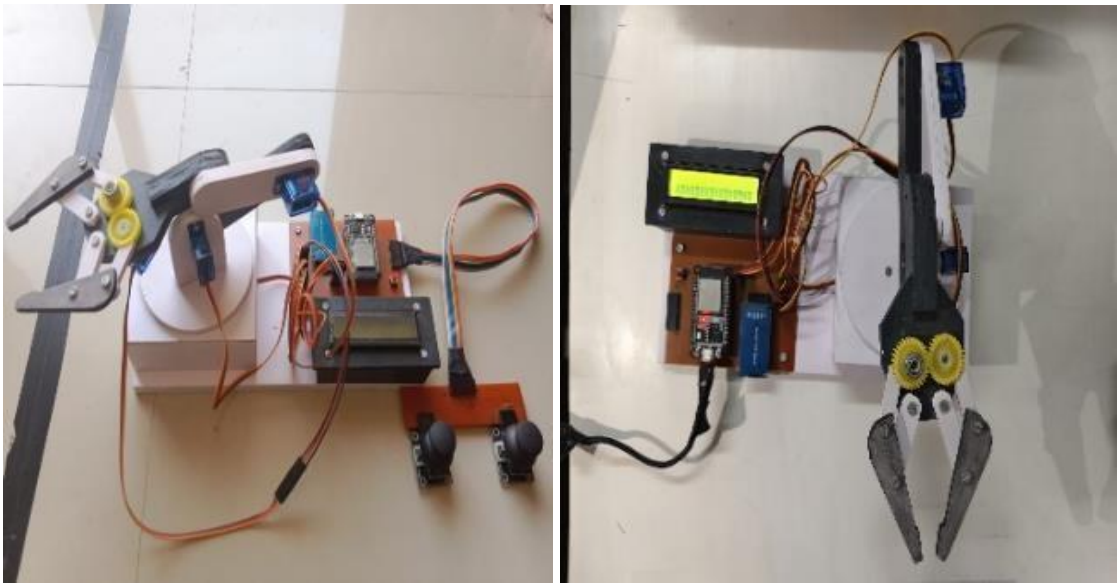

Figure 9. Robotic arm.

However, considering the length of the arm segments and torque loss due to gravity and friction, the practical lifting capacity is limited to approximately 10–15 gm at the end effector for stable operation. Each SG90 servo motor supports an angular range of 0 to 180°, enabling smooth and flexible movement across different joints of the robotic arm, including base rotation, vertical movement, and gripper control.

The robotic arm shows reliable performance in all four modes: Joystick, Blynk, Gesture, and Automatic. The servo motors respond within 0.1 to 0.2 sec for small-angle movements, ensuring quick yet precise positioning. Hand gesture recognition and app-based control have an average latency of around 200–300 msec, depending on communication and data processing. While the arm is not designed for heavy-duty tasks, it functions effectively in educational or demo environments where it manipulates lightweight objects such as paper, foam, or small plastic components. The lightweight construction and responsive motion make it ideal for learning purposes, testing control systems, and showcasing automation principles. Additionally, Table 1 highlights the comparison of the modes of operations.

Table 1. Comparison of the modes of operations.

Parameter	Automatic mode	IoT mode (Blyn k App)	Gesture-based mode
Average response time (ms)	200 ms	500 ms	800 ms
Accuracy (%)	98%	92%	85%
Max load capacity (gm)	15 g	15 g	15 g
Max movement angle (°)	180	180	160
Speed (°/sec)	90°/sec	70°/sec	60°/sec
Power consumption (W)	15 W	17 W	20 W
Control latency (ms)	Minimal	Network-dependent	Processing delay
Stability	High	Medium	Low
User dependency	None	Moderate	High
Complexity of implementation	Low	Medium	High

CONCLUSION

The developed robotic arm successfully operates in three distinct modes: Automatic, IoT-based via Blynk, and Gesture-controlled using OpenCV, demonstrating its multifunctional and adaptive capabilities. The integration of the ESP32 microcontroller and Blynk IoT platform allowed seamless remote control through a smartphone, fulfilling the IoT-based objective. Real-time hand gesture recognition was efficiently achieved using computer vision tools such as OpenCV and Mediapipe, enabling intuitive control over the robotic arm. The use of serial communication and MQTT ensured fast and reliable data exchange between the control interfaces and the arm. This versatile design showcases the system's potential for use in diverse applications such as industrial automation, assistive technologies, and rehabilitation systems, thus meeting all intended objectives of the project.

Acknowledgement

I would like to express my sincere gratitude to Dr. T. V. Kafare sir, our respected guide and professor, for his invaluable support, guidance, and encouragement throughout the course of this project. His insightful suggestions, constant motivation, and expert supervision played a crucial role in the successful completion of our work.

Dr. Kafare sir's deep knowledge, approachable nature, and unwavering support inspired us to explore innovative solutions and tackle challenges with confidence. It was truly a privilege to work under his mentorship, and we are immensely thankful for the time and effort he dedicated to our academic growth and the development of this project.

REFERENCES

1. Sihombing P, Muhammad RB, Herriyance H, Elviwani E. Robotic arm controlling based on fingers and hand gesture. In 2020 IEEE 3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology (MECnIT). 2020 Jun 25; 40–45.
2. Saleheen MM, Fahad MR, Khan R. Gesture-controlled robotic arm. In 2023 IEEE International Conference on Computer Science, Information Technology and Engineering (ICCoSITE). 2023 Feb 16; 495–499.
3. Paterson J, Aldabbagh A. Gesture-controlled robotic arm utilizing opencv. In 2021 IEEE 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). 2021 Jun 11; 1–6.
4. Singh G, Singh AK, Yadav A, Bhardwaj I, Chauhan U. IoT developed Wi-Fi controlled rover with robotic arm using NodeMCU. In 2020 IEEE 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN). 2020 Dec 18; 497–501.
5. Janagiraman S, Mithun P, Hemashri S, Susan MB. Gestures to Movements Crafting A Robotic Arm With OPENCV. In 2024 IEEE International Conference on Emerging Research in Computational Science (ICERCS). 2024 Dec 12; 1–6.

6. Alshdadi AA. Evaluation of IoT-based smart home assistance for elderly people using robot. Electronics. 2023 Jun 11; 12(12): 2627.
7. Chai X, Li G, Lin Y, Xu Z, Tang Y, Chen X, Zhou M. Sign language recognition and translation with kinect. In IEEE Conf on AFGR. 2013 Apr 22; 655: 4.
8. Lu S, Metaxas D, Samaras D, Oliensis J. Using multiple cues for hand tracking and model refinement. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003 Proceedings. 2003 Jun 18; 2: II–443.
9. Ahmed S, Alam R, Hossain MR, Islam MM, Hossain MI, Tabassum T. An IoT based Smart Robot that Aids in the Prevention of COVID19 Spread. In 2022 IEEE 4th International Conference on Sustainable Technologies for Industry 4.0 (STI). 2022 Dec 17; 1–6.
10. Abusukhon A. Toward achieving a balance between the user satisfaction and the power conservation in the internet of things. IEEE Internet Things J. 2021 Jan 18; 8(14): 10998–1015.

