UDC 004.896

Y. Pustovoit, Ye. Batrak, N. Tsopa

AUTOMATED CONTROL SYSTEM FOR A ROBOTIC MANIPULATOR BASED ON ESP32 AND FLUTTER

Abstract: This article presents an automated control system for a robotic manipulator utilizing ESP32 and Flutter. The proposed solution provides real-time control, modularity, and mobile accessibility. The developed system addresses the main issues observed in existing educational and prototyping platforms, offering a modern, intuitive, and scalable design.

Keywords: ESP32, Flutter, WebSocket, robotics, manipulator, automation, mobile application, 3D printing, system, architecture, modeling, automatic control, robotic systems.

Introduction

In the modern context of automation, education, and industrial prototyping, a key challenge is the development of affordable and adaptable robotic solutions. Robotic manipulators play a vital role in educational and industrial processes, enabling the simulation of real-world control scenarios, automating routine tasks, and testing object recognition and grasping algorithms. However, most existing systems are either too complex to implement or financially inaccessible for student labs, technical clubs, or hobby projects.

Moreover, many available manipulators lack support for modern real-time communication protocols, scalable architectures, or intuitive user interfaces, limiting their suitability for rapid prototyping, educational use, or home applications. At the same time, the development of microcontrollers with built-in communication modules and the emergence of cross-platform interface development tools create a favorable environment for designing simple, efficient, and affordable solutions.

Currently, several solutions partially or fully meet the needs of educational and prototyping environments. Commercial products like Dobot Magician, uArm Swift Pro, and ArmPi are high-precision platforms with advanced functionality, but their cost hinders widespread adoption. Open-source projects based on Arduino and 3D printing, such as BCN3D Moveo and MeArm, are common but often suffer from low accuracy, lack of real-time feedback, and challenging integration with mobile devices. Some projects, like Robot Arm Web Control, implement WebSocket-based control but require web servers or lack mobile interface support.

Thus, the task of creating a comprehensive hardware-software platform with support for modern technologies remains relevant. A pathway to such a platform lies in combining contemporary hardware design approaches with software development that ensures intuitive control and automation capabilities. This approach not only simplifies manipulator use but also opens new opportunities for learning, experimentation, and practical application, allowing users to customize the system to their needs.

Objective and Tasks

The goal of the development presented in this work is to improve the accessibility of automation for a broad range of users by creating an economical, flexible, and user-friendly manipulator control system capable of performing basic grasping, transferring, and motion automation tasks with high accuracy.

To achieve this goal, the following tasks must be accomplished:

1. Develop hardware using affordable and reliable components;
2. Implement a mobile application with an intuitive interface;
3. Ensure bidirectional real-time communication between the app and the controller;
4. Implement support for both manual and automatic manipulator operation modes;
5. Ensure system scalability (sensors, scenario storage, Internet-based control).

The presented results can contribute to the popularization of robotics by providing an accessible tool for learning and experimentation. The system will allow users to perform various tasks, from classroom demonstrations to prototyping automation in workshops. Additionally, it can serve as a foundation for further development of similar systems, including the integration of autonomous functions, remote control, and eco-friendly solutions.

Main Content of the Research

The system presented in this work consists of hardware and software components. The hardware includes structural elements, a control board, and connectors. The design features four degrees of freedom: a base platform for rotation around a vertical axis, an arm for vertical lifting, an elbow for bending, and a rotary module for adjusting the gripper orientation. The mechanical gripper is a two-finger claw mechanism controlled by a separate servo motor and designed to hold lightweight objects.

The software component includes a mobile application, network communication, and a microcontroller program. The developed system interface allows users to set servo motor angles using sliders, control the gripper with dedicated buttons, and input command sequences for automatic mode.

The system operates in two modes: manual, which enables real-time manipulator control, and automatic, where the manipulator executes a pre-defined sequence of actions. The system starts with power-up, activating the control board and servos. During initialization, the board creates a Wi-Fi access point to which the mobile application connects. After establishing a WebSocket connection, the system is ready to receive commands.

In manual mode, the system allows users to set rotation angles for each servo via sliders in the app. For example, the command "M2:120" corresponds to rotating the second servo to 120 degrees. This command is transmitted via sockets to the control board, which processes it and generates a signal for the corresponding servo. The manipulator moves instantly, allowing for quick position adjustments.

In automatic mode, a sequence of commands can be defined, such as "M1:90, M2:120, M5:30, M1:120, M5:0". This scenario is sent to the board, which executes the movements sequentially, adhering to specified delays between commands. This mode simulates basic autonomous behavior and can be used for demonstrations or repetitive tasks.

Hardware Component of the System

The central hardware component of the developed system is the ESP32 microcontroller, equipped with a dual-core 32-bit Tensilica LX6 processor, built-in Wi-Fi and Bluetooth modules, and a wide range of peripheral interfaces (GPIO, SPI, I2C, UART, etc.). This functionality enables both autonomous and networked control scenarios.

For generating PWM signals to control the servos, a dedicated PCA9685 driver is used. It interfaces via I2C and supports up to 16 independent PWM channels, significantly reducing the microcontroller's workload and ensuring smooth, precise control of multiple servos simultaneously.

The utilized actuators include:

– MG995: Powerful servos capable of generating high torque, used for base rotation, arm lifting, and elbow movement;

– SG90: Lightweight, compact servos for precise movements, such as gripper control.

Servos are powered separately from the ESP32 using an external 6V power source. A 3A power supply connects to the driver through a voltage stabilization module.

Fig. 1 presents the functional diagram of the developed system, illustrating the sequence of actions when using the system in automatic and manual modes. The system includes user interface modules, automatic and manual control modules, and a robot motion control module with servos. The process involves command creation, validation, transmission, processing, and execution, considering system status and responses.

Manipulator Design

The mechanical structure is implemented as a manipulator with 4 degrees of freedom, enabling basic grasping and movement operations:

1. Base rotation (MG995);
2. Arm lifting (MG995);
3. Elbow bending (MG995);
4. Gripper rotation (SG90);
5. Gripper open/close mechanism (SG90).

The design was created in Fusion 360, considering available materials and servo dimensions. Components were 3D-printed using FFF technology with PLA and ABS, accounting for thermomechanical properties.

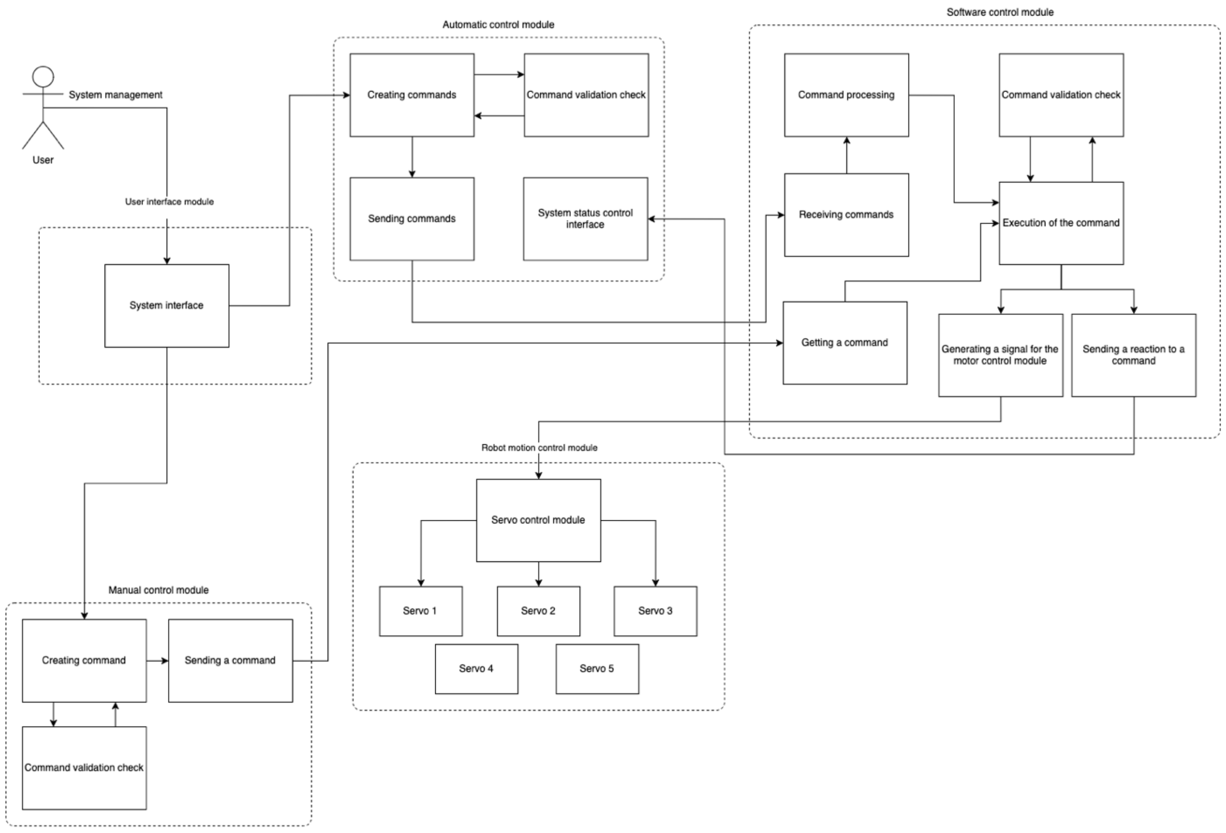


Figure 1. Functional Diagram of the System

The modular design allows for easy upgrades and quick replacement of damaged parts without full disassembly.

Fig. 2 shows the fully assembled manipulator with all its components. The base contains a mounting point for the servo that enables the entire manipulator's rotation. The base consists of a foundation (1 in Fig. 2) and a rotating segment (2 in Fig. 2), responsible for the manipulator's rotation. The first segment (3 in Fig. 2)—the arm—connects the base to the next element. The arm includes a mounting point for the rotary shaft. The next element is the elbow segment (4 in Fig. 2), or second lever, providing additional arm flexibility and driven by a separate servo. At the elbow's end is a mounting point for the gripper (5 in Fig. 2) to the manipulator's body. The final element is the manipulator's gripper (6 in Fig. 2).

3D Modeling

Each manipulator part was modeled individually as a 3D structure, allowing for future replacement or optimization. The gripper design includes two independently moving fingers for reliable object holding.

Key technical characteristics:

– Minimized moving mass;
– Symmetric geometry to reduce inertia;
– Reinforced ribs in critical areas;
– No unnecessary protrusions to minimize jamming risks.

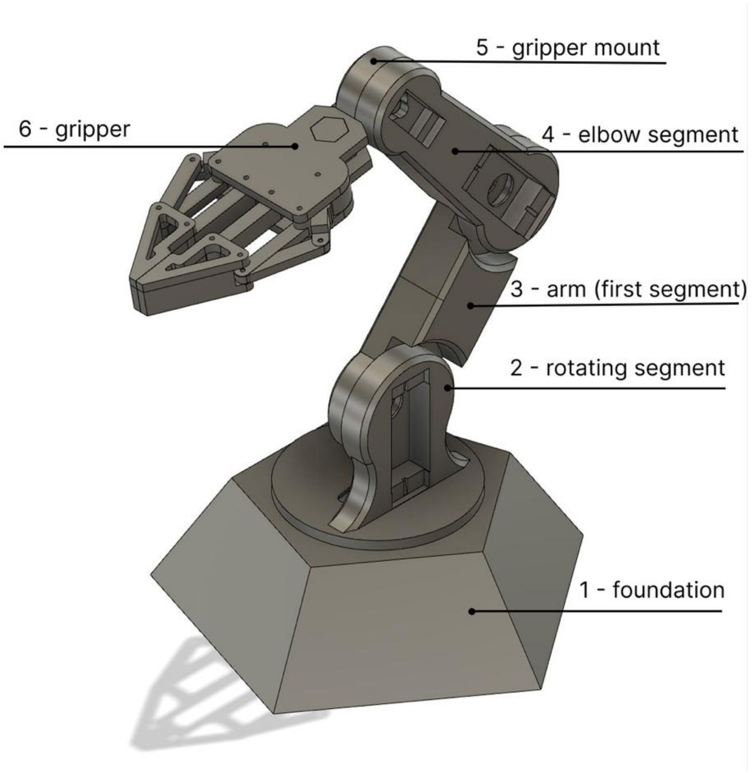


Figure 2. Assembled Robotic Manipulator Design

Geometric accuracy was verified through manual animation in Fusion 360, including checks for rotation tolerances, inter-part distances, and compatibility of mounting holes for servos.

Software Architecture

The mobile application was developed using Flutter, a framework that enables cross-platform app development with a single Dart codebase.

This follows a "Clean Architecture" pattern:

– Data: ServoCommand and Preset classes, repositories;

– Manager: ServoControllerManager class, which converts user actions into commands;

– Presentation: User interface with a provider for state management, user sliders, buttons, and modes.

Tabs include:

– Manual control: sliders for each servo;

– Automatic control: command list, delay settings, and scenario execution.

The app features basic data validation, error input protection, and connection loss detection.

Command format:

– "M1:100,M2:80" for automatic mode;

– "M1:100" for manual mode.

Each command is split into individual values, and ESP32 executes them sequentially. In the future, these scenarios will be stored in a device database or memory. The interface is adaptive to various screen sizes, includes error handling, and supports light/dark themes.

Manipulator control is implemented in two modes, switchable via app tabs: "Manual Control" (Fig. 3) and "Automatic Mode" (Fig. 4). This approach clearly separates interaction logic, avoiding interface clutter and simplifying navigation. In manual mode (Fig. 3), each degree of freedom is controlled by a separate slider, with values corresponding to the current angle of the respective servo. Changing the slider position immediately generates a command in the format "M{id}:{angle}", sent to ESP32 via WebSocket. This ensures real-time manipulator control and immediate visual feedback for the user. The model is designed for interactive work, position testing, and individual adjustments.

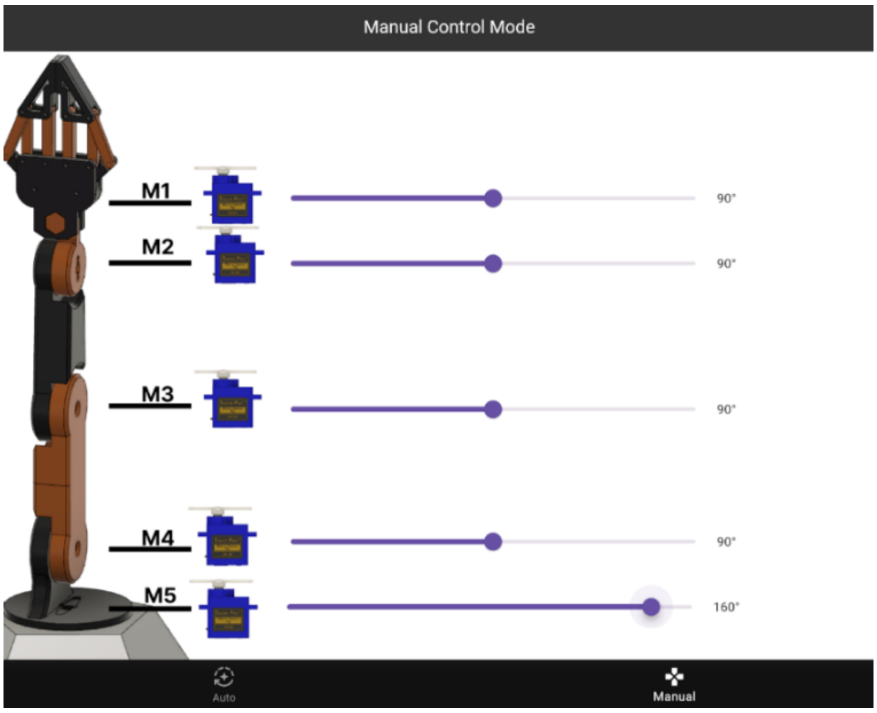
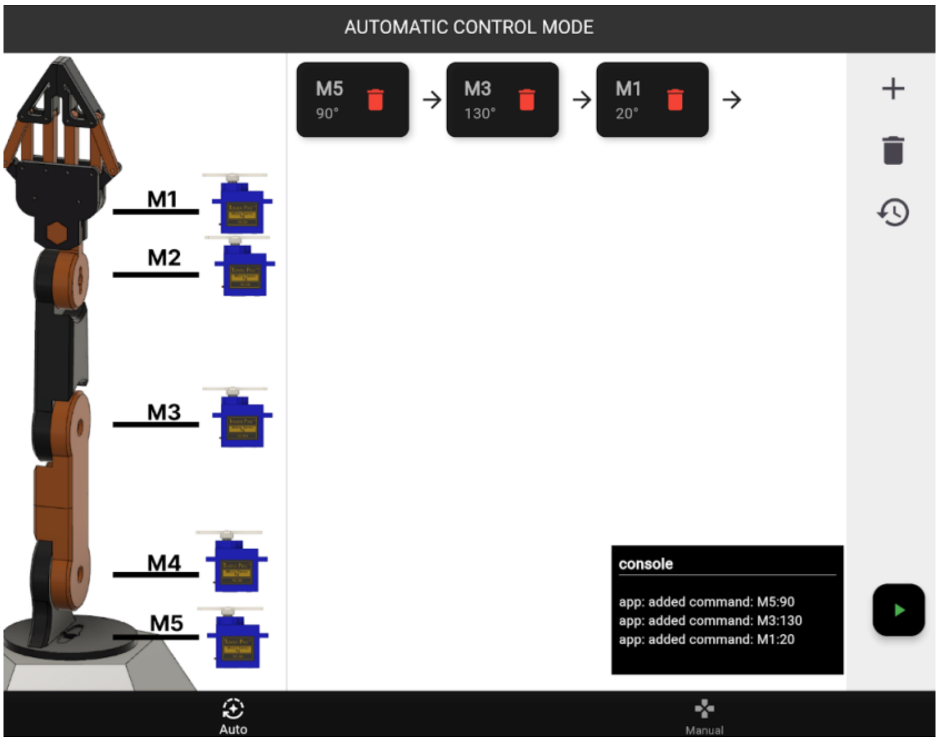


Figure 3. "Manual Control" Interface Tab

Automatic mode (Fig. 4) allows the creation and execution of command sequences. This tab features an interface for step-by-step instruction addition, accumulating commands in an internal list. The formed sequence is displayed as an editable list that can be cleared before sending. After editing, the user presses "Start," combining all saved commands into a comma-separated string (e.g., "M1:90,M2:45,M3:120") and sending it to ESP32 as a single packet. The microcontroller processes the instructions sequentially, enabling basic autonomous behavior without manual intervention.

The use of both modes provides flexibility for experimental control and convenience for repeating scenarios, which is particularly useful in educational and demonstration settings where visibility and controllability are key.



console

app: added command: M5:90
app: added command: M3:130
app: added command: M1:20

Figure 4. "Automatic Control" Interface Tab

ESP32 Software Logic

The logic on ESP32 is structured as follows:

– Network initialization and access point creation;
– WebSocket server startup;
– Receiving text messages (commands);
– Parsing: splitting by commas, extracting Mx:angle values;
– Validation: checking valid ranges;
– Sending signals to PCA9685;
– Outputting PWM signals to the corresponding channel.

In case of connection loss, ESP32 enters a waiting mode.

The sequence of interactions between the mobile app and ESP32 microcontroller is depicted in Fig. 6 as a sequence diagram, illustrating real-time command exchange.

Testing and Analysis of System Performance

Testing confirmed the developed robotic manipulator control system's functionality across all key aspects, including hardware, software, and communication components. Testing covered ESP32 initialization, MG995 and SG90 servo operation, WebSocket connection stability, positioning accuracy in manual and automatic modes, and mobile app interface usability.

ESP32 reliably created a Wi-Fi access point in 3-5 seconds, though delays reached 10 seconds in some cases due to power fluctuations. MG995 servos performed reliably at 0° and 180° without load, but with objects over 300g, slight deviations occurred due to 3D-printed part play. SG90 servos effectively held small objects but were limited by low torque (2.5 kg·cm). The PCA9685 driver ensured smooth control of five servos with high resolution.

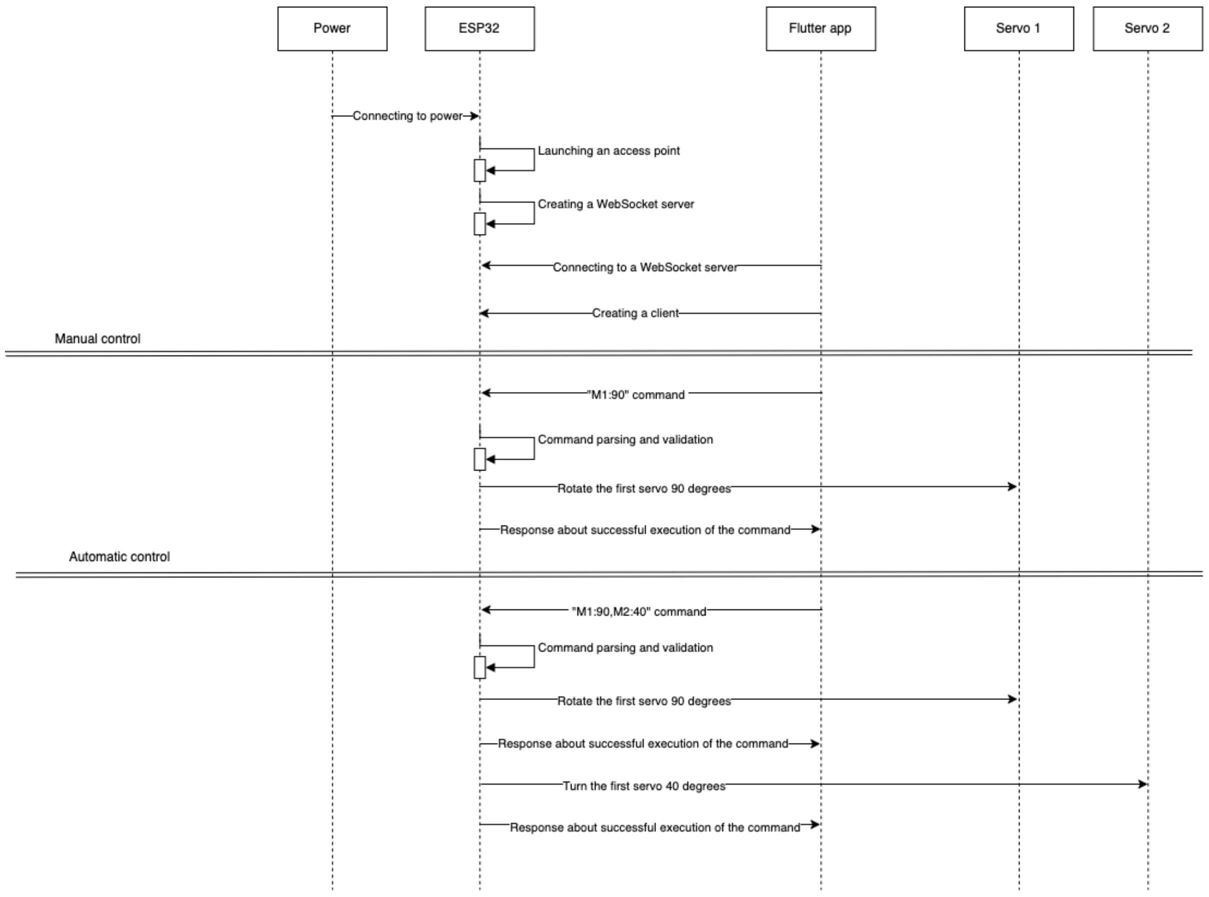

Power
ESP32
Flutter app
Servo 1
Servo 2
Connecting to power
Launching an access point
Creating a WebSocket server
Connecting to a WebSocket server
Creating a client
Manual control
"M1:90" command
Command parsing and validation
Rotate the first servo 90 degrees
Response about successful execution of the command
Automatic control
"M1:90,M2:40" command
Command parsing and validation
Rotate the first servo 90 degrees
Response about successful execution of the command
Turn the first servo 40 degrees
Response about successful execution of the command

Figure 6. Command Processing Diagram on ESP32

Average WebSocket command transmission delay was 70-100 ms at 1m and 80-120 ms at 5m, sufficient for local use. Invalid commands were handled correctly with error messages ("ESP:ERR"). In manual mode, positioning error was ±2° for MG995 and ±1.5° for SG90, with a reaction time of 0.4-0.6s. In automatic mode, command sequences executed correctly, but with 6-command scenarios, 25% showed delays up to 0.7s due to ESP32 buffer overload. The Flutter app interface proved adaptive and user-friendly across various screen sizes.

Conclusions

This work presents the development of an accessible, modular, and scalable automated control system for a robotic manipulator based on the ESP32 microcontroller and a Flutter mobile application. The proposed solution addresses the main issues observed in existing educational and prototyping platforms, offering a modern, intuitive, and expandable design.

Key achievements include:

1. Hardware Integration: The combination of ESP32 with the PCA9685 driver enabled efficient real-time control of multiple servos with minimized microcontroller load.

2. Mechanical Design: The modular 3D-printed structure allows easy maintenance, upgrades, and part replacement, making the manipulator ideal for iterative prototyping and educational settings.

3. Cross-Platform Interface: The Flutter-based mobile app provides seamless control on Android and iOS devices, with separate modes for manual and automatic operation, catering to diverse user needs.

4. Real-Time Communication: WebSocket implementation ensures low-latency, bidirectional data exchange between the app and ESP32, supporting precise and fast control scenarios.

The system architecture also allows for future enhancements, such as:

– Storage and execution of pre-defined action sequences;
– Integration of additional sensors and feedback loops;
– Remote access via Internet connectivity;
– Expansion of the command protocol for advanced functions.

5. Verified System Functionality: Testing results confirmed stable operation of all components. ESP32 creates a Wi-Fi access point, MG995 and SG90 servos demonstrate sufficient accuracy, the PCA9685 driver ensures smooth control, WebSocket operates with low latency for real-time management, automatic mode correctly executes command scenarios, and the app interface is adaptive and user-friendly.

As a result, the developed robot control system serves as a reliable foundation for educational purposes, amateur experiments, and further research in automation, mobile robotics, and user-oriented interface development.

REFERENCES

1. Dobot Magician – Educational Robotic Manipulator *DOBOT*: website URL: https://www.dobot-robots.com/products/education/magician.html (application date: 18.06.2025).

2. uArm Swift Pro – Open-Source Robotic Manipulator *UFACTORY*: website URL: https://www.ufactory.cc/products/uarm-swift-pro (application date: 18.06.2025).

3. ArmPi – Robotic Manipulator with Intelligent *Hiwonder*: website URL: https://www.hiwonder.com/products/armpi (application date: 18.06.2025).

4. BCN3D Moveo – Open-Source Robotic Manipulator [Electronic resource] // GitHub. – Available from: https://github.com/BCN3D/BCN3D-Moveo (application date: 18.06.2025).

5. MeArm – DIY Robotic Arm Kit *MeArm*: website URL: https://mearm.com/ (application date: 18.06.2025).

6. Wi-Fi Browser-Controlled Robotic Arm with Arduino [Electronic resource] // Instructables. – Available from: https://www.instructables.com/Wi-Fi-Browser-Controlled-Robotic-Arm-with-Arduino-/ (application date: 18.06.2025).

7. WebSocket – Real-Time Communication Protocol *Wikipedia*: website URL: https://uk.wikipedia.org/wiki/WebSocket (application date: 18.06.2025).

8. Espressif Systems – ESP-IDF Programming Guide [Electronic resource]. – Available from: https://docs.espressif.com/projects/esp-idf/en/latest/esp32 (application date: 18.06.2025).

9. MG995 – Servo Motor [Electronic resource]. – Available from: https://www.alldatasheet.com/view.jsp?Searchword=Mg995 (application date: 18.06.2025).

10. SG90 – Servo Motor [Electronic resource]. – Available from: https://www.scribd.com/document/436710755/sg90-datasheet-pdf (application date: 18.06.2025).

11. PCA9685 – 16-Channel 12-Bit PWM Servo Driver with I2C [Electronic resource]. – Available from: https://arduino.ua/prod1442-16-kanalnii-12-bit-pwmservo-modyl-s-i2c-interfeisom-na-pca9685 (application date: 18.06.2025).

12. Flutter – Mobile Application Development Framework *Flutter*: website URL:: https://flutter.dev/ (application date: 18.06.2025).