Development and Control of an ESP32-Based Smart Robot Arm Using IoT and Wireless Interface

Sassan Ghazi

***Abstract*— This paper presents the development and control methodology of a 4-DOF Smart Robot Arm powered by the ESP32 microcontroller. The arm is capable of precise movement using servo motors and can be controlled wirelessly via Bluetooth or Wi-Fi through a mobile application. This project demonstrates an efficient combination of mechanical design, embedded systems, and IoT technologies, offering real-world applications in automation and remote manipulation tasks.**

I. INTRODUCTION

With the rise of automation and Internet of Things (IoT), smart robotic systems have become increasingly prevalent in various industries. Among them, robotic arms serve as versatile manipulators in tasks ranging from assembly lines to medical surgeries. In this paper, we present a cost-effective and compact smart robotic arm built using the Keyestudio ESP32 platform. This project explores the design, programming, and control aspects of the robotic arm, including its integration with a mobile control interface using Bluetooth Low Energy (BLE).

II. SYSTEM DESIGN AND METHODOLOGY

The smart robotic arm is constructed using four servo motors that provide degrees of freedom (DOFs) to the base, shoulder, elbow, and gripper. The ESP32 microcontroller serves as the central control unit, receiving commands from a smartphone app via Bluetooth.

A. Hardware Components

- ESP32 microcontroller board
- 4x SG90 servo motors
- Power supply module (Battery or USB)
- Robot arm frame (acrylic or plastic)
- Bluetooth-enabled smartphone

B. Software Stack

- Arduino IDE with ESP32 Board Support
- Keyestudio Robot Arm library
- MIT App Inventor for custom mobile app

III. EXPERIMENTS AND RESULTS

Several tests were conducted to evaluate the arm's response time, positional accuracy, and wireless communication stability.

This work was not supported by any organization.
Email: sassanghazi@gmail.com

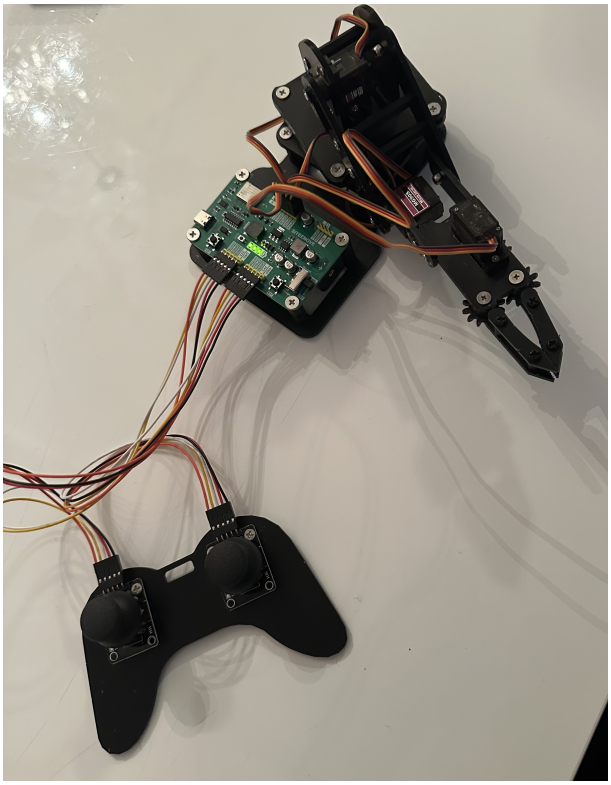

Fig. 1. Full hardware setup of the ESP32-based smart robotic arm.

A. Experiment 1: Motion Accuracy

Commands were sent via mobile app to test each joint. Results showed an average error of less than 5 degrees across movements.

B. Experiment 2: Wireless Control Latency

Bluetooth signal latency was tested over 5 meters and remained under 100ms, allowing near real-time control.

C. Experiment 3: Task Execution

The arm was programmed to pick up light objects (e.g., plastic cubes) and move them to a target location. The success rate was above 90%.

IV. FEATURES

A. Flexibility

The robotic arm provides four degrees of freedom, enabling it to move and rotate in multiple directions. This allows for the execution of complex manipulation tasks such as grasping, placing, and rotating objects in constrained environments.

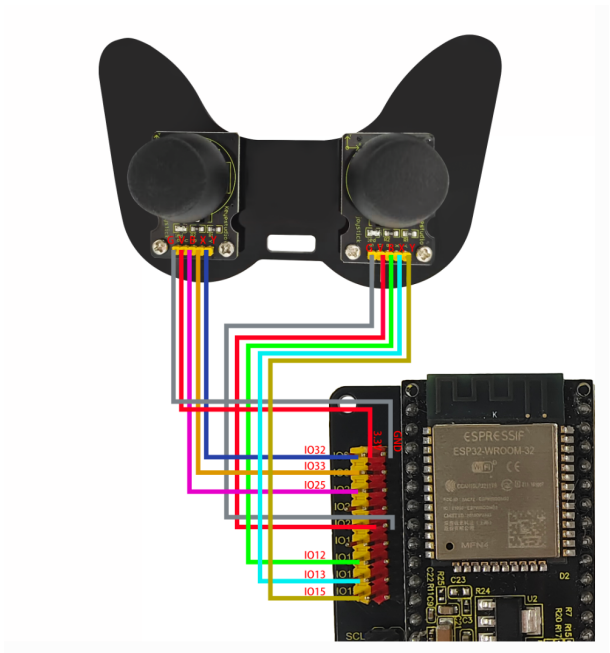

Fig. 2. Wiring connections between the ESP32 board and servo motors.

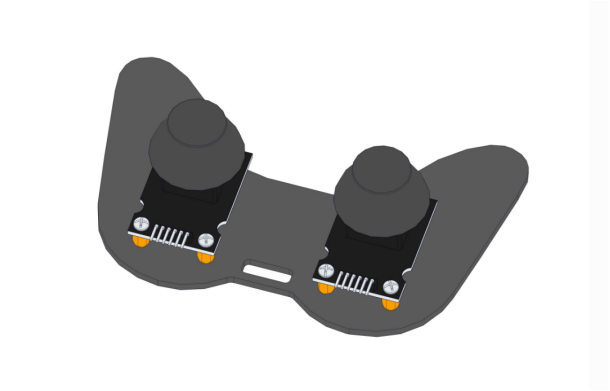

Fig. 3. Gaming controller replica used for controlling the motions of the robot arm.

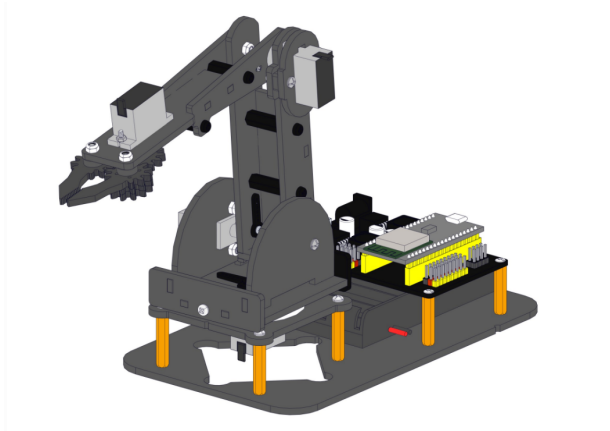

Fig. 4. The 4-DOF robotic arm used in the experiments.

B. Convenient Control

The system is based on the ESP32 microcontroller, which serves as the control core. The ESP32 features a rich set of interfaces that facilitate communication and control with peripheral devices, sensors, and external systems.

C. Programming Flexibility

The robotic arm supports programmable motion trajectories and behaviors. Users can implement automated operations such as object recognition, pick-and-place routines, and real-time adjustments via software.

D. Openness and Expandability

The ESP32 platform is part of an open-source ecosystem, making it highly customizable and extendable. Additional sensors and actuators can be added to support a wide range of application scenarios, from industrial automation to educational use.

V. KEYESTUDIO ESP32 MAIN BOARD

A. Introduction

The Keyestudio ESP32 Core Board is based on the ESP-WROOM-32 module and designed for ease of development and prototyping. It provides standard 2.54mm-spaced headers on both sides, simplifying the connection to peripheral components. The ESP32 module integrates a Wi-Fi and Bluetooth solution, requiring minimal external components and featuring excellent RF performance.

It leverages TSMC's low-power 40nm technology, offering high performance with low energy consumption. The platform is secure, reliable, and suitable for a wide variety of embedded applications.

B. Technical Specifications

- **Microcontroller:** ESP-WROOM-32
- **USB-to-Serial Chip:** CP2102-GMR
- **Operating Voltage:** 5V DC
- **Average Operating Current:** 80mA
- **Minimum Supply Current:** 500mA
- **Operating Temperature Range:** -40°C to +85°C
- **Wi-Fi Modes:** Station / SoftAP / SoftAP+Station / P2P
- **Wi-Fi Protocol:** IEEE 802.11 b/g/n/e/i (up to 150 Mbps)
- **Bluetooth Protocol:** Bluetooth v4.2 BR/EDR and BLE
- **Frequency Band:** 2.4–2.5 GHz
- **Dimensions:** 55mm × 26mm × 13mm
- **Weight:** 9.3g

C. Pinout Overview

Although the ESP32 has fewer I/O pins compared to some other development boards, it offers multi-functional pins that allow for efficient use. Key pin types include:

- **Power Pins:** +5V and 3.3V output to power external modules
- **GND:** 3 ground pins
- **Enable (EN):** Used to enable or disable the board; high to enable

- **GPIO:** 32 general-purpose input/output pins (note: GPIO6 to GPIO11 are reserved for flash memory and not recommended for use)
- **ADC:** 16 analog-to-digital converter channels
- **DAC:** 2 digital-to-analog converter channels (8-bit resolution)
- **Touch Pads:** 10 capacitive touch pins
- **SPI:** Two SPI interfaces
- **I2C:** SDA and SCL for inter-device communication
- **UART:** Two UART serial ports (UART0 supports CTS/RTS)
- **PWM:** All I/O pins can generate PWM signals

Important Note: The operating voltage of ESP32 GPIO pins is 3.3V. When interfacing with 5V devices, a level shifter or voltage divider is required to prevent damage.

D. Main Components

The ESP-WROOM-32 module includes the following internal components:

- Antenna switch
- RF balun
- Power amplifier
- Low-noise amplifier
- Filters
- Power management module

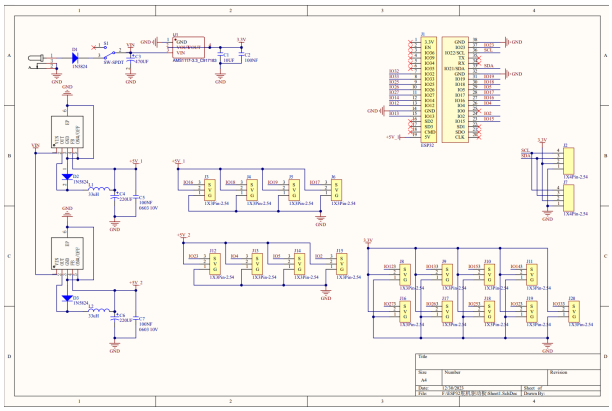

Fig. 5. Schematic diagram showing ESP32 interfacing with servo motors.

VI. Wi-Fi Control and Web Interface

A. Overview

In addition to Bluetooth control, the ESP32-based robotic arm also supports control via Wi-Fi, allowing access to a web-based interface for real-time manipulation from any device connected to the same network.

B. Wi-Fi Setup Requirements

To enable Wi-Fi control, you will need:

- A 2.4 GHz Wi-Fi network (hotspot or router)
- A phone, tablet, or computer connected to the same network
- The Wi-Fi SSID and password

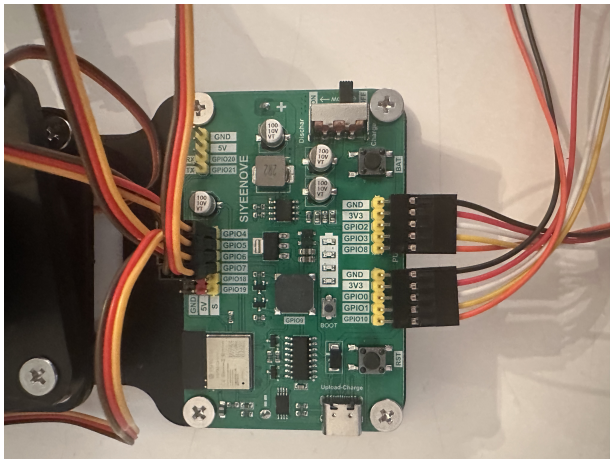

Fig. 6. Siyeenove sensor board optionally used for system expansion.

Before uploading the code, replace the placeholder strings with your Wi-Fi credentials:

```
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

```
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
```

Fig. 7. Example of how to input Wi-Fi name and password in the ESP32 code.

C. Basic Wi-Fi Connection Code

The following code connects the ESP32 to a Wi-Fi network and prints the IP address to the Serial Monitor:

```
/*
  ESP32 WiFi Connection Example
*/
#include <WiFi.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected.");
  Serial.println("IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

D. Web-Based Control Interface

An advanced version of the code allows for a full web interface with sliders for controlling each servo. When accessed via the IP address of the ESP32, the web page provides a GUI for real-time control.

Due to its length, the full code is included in the project repository, but follows this structure:

- Initializes servos and Wi-Fi
- Sets up a local web server on port 80
- Handles incoming HTTP requests
- Updates servo angles based on slider input
- Returns an HTML page with JavaScript-based gauges and sliders

Key functions include:

- ‘baseControl(int angle)‘
- ‘armControl(int angle)‘
- ‘forearmControl(int angle)‘
- ‘gripperControl(int angle)‘

These allow smooth servo motion with real-time feedback.

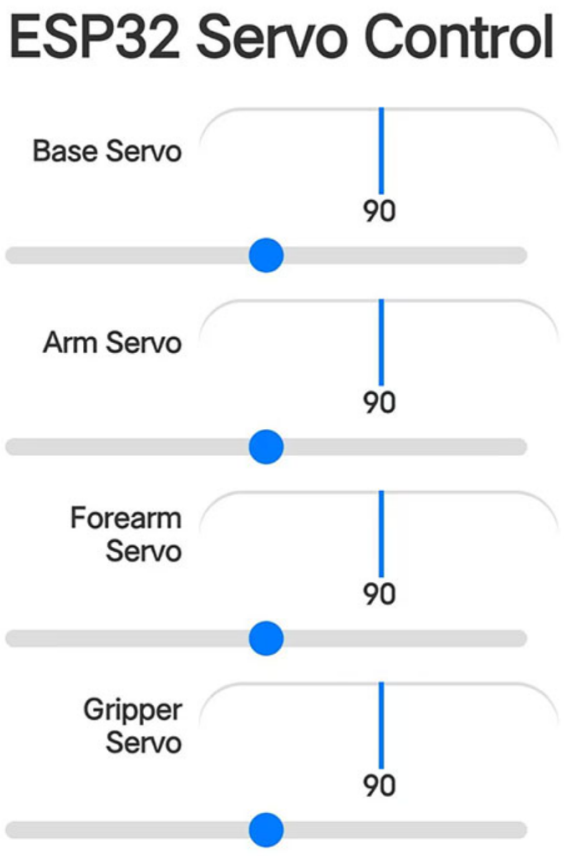

Fig. 8. Web-based mobile interface for real-time control of the robotic arm via Wi-Fi.

VII. CONCLUSION

This paper demonstrates a practical implementation of a smart robotic arm using ESP32 and servo motors. The project highlights how low-cost hardware combined with open-source tools can create effective robotic systems for educational and hobbyist applications. Future work may involve integrating sensors for object detection and implementing inverse kinematics for more complex control.

FUTURE WORK

Future improvements may include:

- Integration of camera and vision-based object detection
- Use of inverse kinematics for precise point targeting
- Cloud-based control or OTA updates
- Voice or gesture-based control system

ACKNOWLEDGMENT

I would like to thank Keyestudio for their detailed tutorial and support resources. Special thanks to my instructors and peers who provided guidance throughout this project.

PROJECT REPOSITORY

Source code, documentation, and additional resources are available at:
https://github.com/sassanghz/ESP32-Smart-Robot-Arm

REFERENCES

[1] Keyestudio Smart Robot Arm Kit Tutorial, FKS0003. Available: https://docs.keyestudio.com/projects/FKS0003/en/latest/docs/Smart
[2] Arduino Reference, Available: https://www.arduino.cc/reference/en/
[3] ESP32 Datasheet. Espressif Systems. Available: https://www.espressif.com/en/products/socs/esp32
[4] MIT App Inventor. Available: https://appinventor.mit.edu/