# **Project Documentation: IoT-Based Control System Using Raspberry Pi, ESP32, ESP8266, and MQTT**

## **1. Introduction**

This project presents an advanced IoT-based control system integrating multiple embedded devices, including **Raspberry Pi Zero W**, **ESP32**, and **ESP8266**, leveraging the **MQTT** protocol for seamless communication. The system is designed to process sensor data, execute motor control, and manage servo operations based on user input and real-time sensor readings. **Mosquitto** is utilized as the MQTT broker to facilitate efficient and lightweight data exchange.

## **2. System Architecture**

The system comprises the following interconnected components:

1. **Raspberry Pi Zero W**: Functions as the central node, reading IR sensor data and publishing it to the MQTT broker.
2. **ESP32**: Responsible for motor control, receiving commands via MQTT.
3. **ESP8266**: Manages servo motors based on IR sensor input and password validation through MQTT.
4. **Bash Script**: Provides a command-line interface for users to send control commands via MQTT.

The communication framework is built on **MQTT**, ensuring robust and efficient device interaction.

## **3. Hardware Components**

* **Raspberry Pi Zero W** – Reads IR sensor data and publishes messages to MQTT.
* **ESP32** – Controls two DC motors using an H-bridge motor driver.
* **ESP8266** – Operates servo motors via a PCA9685 PWM driver.
* **IR Sensors** – Detect environmental changes and provide data to Raspberry Pi.
* **Servo Motors** – Controlled by ESP8266 based on IR sensor data and password authentication.
* **DC Motors** – Managed by ESP32 according to MQTT commands.

## **4. Software Components**

* **Mosquitto MQTT Broker** – Facilitates device communication.
* **Paho MQTT Client** – Enables MQTT connectivity for Raspberry Pi and ESP devices.
* **Adafruit PWM Servo Driver Library** – Controls servo motors via ESP8266.
* **Bash Script** – Provides a CLI-based control interface.

## **5. System Workflow**

### **5.1 Raspberry Pi Zero W**

* Reads IR sensor data from GPIO pins.
* Publishes sensor data to the MQTT topic ir/sensors.
* Captures user password input and publishes it to password/topic.

### **5.2 ESP32**

* Subscribes to MQTT topics motor1/control and motor2/control.
* Controls two DC motors based on received MQTT messages.

### **5.3 ESP8266**

* Subscribes to MQTT topics ir/sensors and password/topic.
* Operates servo motors based on sensor input.
* Validates the password and adjusts servo positions accordingly.

### **5.4 Bash Script**

* Provides a CLI for motor control.
* Sends MQTT commands to actuate motors.

## **6. Detailed Component Breakdown**

### **6.1 Raspberry Pi Zero W**

**Key Features:**

* Reads IR sensor data via GPIO pins (2, 3, and 4).
* Publishes data to MQTT broker using Paho MQTT.
* Captures user-inputted passwords and relays them through MQTT.

**Workflow:**

1. Initialize GPIO pins for IR sensor integration.
2. Establish connection with the MQTT broker.
3. Continuously monitor IR sensor readings and publish updates.
4. Receive user password input and transmit it via MQTT.

### **6.2 ESP32**

**Key Features:**

* Controls two DC motors through GPIO (pins 27, 26, 25, and 33).
* Communicates via MQTT to receive motor control commands.

**Workflow:**

1. Connect to Wi-Fi and MQTT broker.
2. Subscribe to motor control topics.
3. Actuate motors based on received MQTT messages.

### **6.3 ESP8266**

**Key Features:**

* Uses PCA9685 for PWM-based servo control.
* Processes password authentication for servo actuation.

**Workflow:**

1. Connect to Wi-Fi and MQTT broker.
2. Subscribe to ir/sensors and password/topic.
3. Operate servo motors based on IR sensor feedback.
4. Validate password input and adjust servo accordingly.

### **6.4 Bash Script**

**Key Features:**

* CLI-based user interaction for motor control.
* Publishes MQTT messages for motor actuation.

**Workflow:**

1. Prompt user for motor selection (Motor 1, Motor 2, or Both).
2. Capture user input for motor state (ON/OFF).
3. Publish MQTT message to control motors.

## **7. MQTT Topic Structure**

The system utilizes the following MQTT topics:

* ir/sensors – Publishes IR sensor readings from Raspberry Pi.
* password/topic – Publishes user-inputted passwords.
* motor1/control – Controls DC Motor 1.
* motor2/control – Controls DC Motor 2.

## **8. Conclusion**

This project showcases a sophisticated IoT system integrating **Raspberry Pi Zero W**, **ESP32**, and **ESP8266** using **MQTT** for efficient device communication. The system is designed to read sensor data, execute motor control, and manage servo actuations, ensuring reliable and scalable operation.

## **9. Future Enhancements**

1. **Security Improvements** – Implement **TLS/SSL encryption** for secure MQTT communication.
2. **User Interface** – Develop a **web-based or mobile application** for enhanced user experience.
3. **Data Logging & Analytics** – Integrate **database storage** for sensor and control logs.
4. **Cloud Integration** – Extend functionality using **AWS IoT** or **Google Cloud IoT**.