Development of a Bluetooth-Controlled IoT-Based Smart Home System for Energy Efficiency and Security

Tran Dinh Thien, Nguyen Phu Cuong, Nguyen Huynh Nhat Thinh, Truong Doan Anh Khoa
FPT University, Ho Chi Minh Campus, Vietnam
ducdnm2@fe.edu.vn

Abstract

This paper introduces the design and development of a smart home system powered by Internet of Things (IoT) technologies. The system leverages Bluetooth and Wi-Fi connectivity to enhance automation, energy efficiency, and home security. Through a combination of microcontrollers (Arduino Uno and ESP32), various environmental sensors, and a web-based interface, the proposed system enables real-time monitoring and control of home appliances and conditions. Experimental validation demonstrates the system's ability to automate responses to environmental inputs, authorize user access through RFID, and allow remote control through a web platform. The paper concludes with discussions on results, limitations, and future improvement directions.

I. Introduction

The modern home increasingly integrates intelligent technologies to enhance user convenience, improve safety, and optimize energy consumption. Internet of Things (IoT) solutions offer homeowners the capability to monitor and control home functions from remote locations using smart devices. However, many existing systems are limited by high implementation costs or lack of customization.

This project aims to build a cost-effective, modular smart home system using open-source microcontrollers and standard communication protocols. The system architecture includes Bluetooth-enabled Arduino and Wi-Fi-enabled ESP32 to manage home access, monitor environmental parameters, and communicate with a cloud-based or local web application.

Our design objectives include:

- Enable secure access using RFID verification.
- Monitor environmental conditions (e.g., motion, gas, obstacles).
- Provide automation through sensor-triggered events.
- Facilitate manual overrides via a responsive web interface.

Fig. 1. First part: Smart home concept illustration.

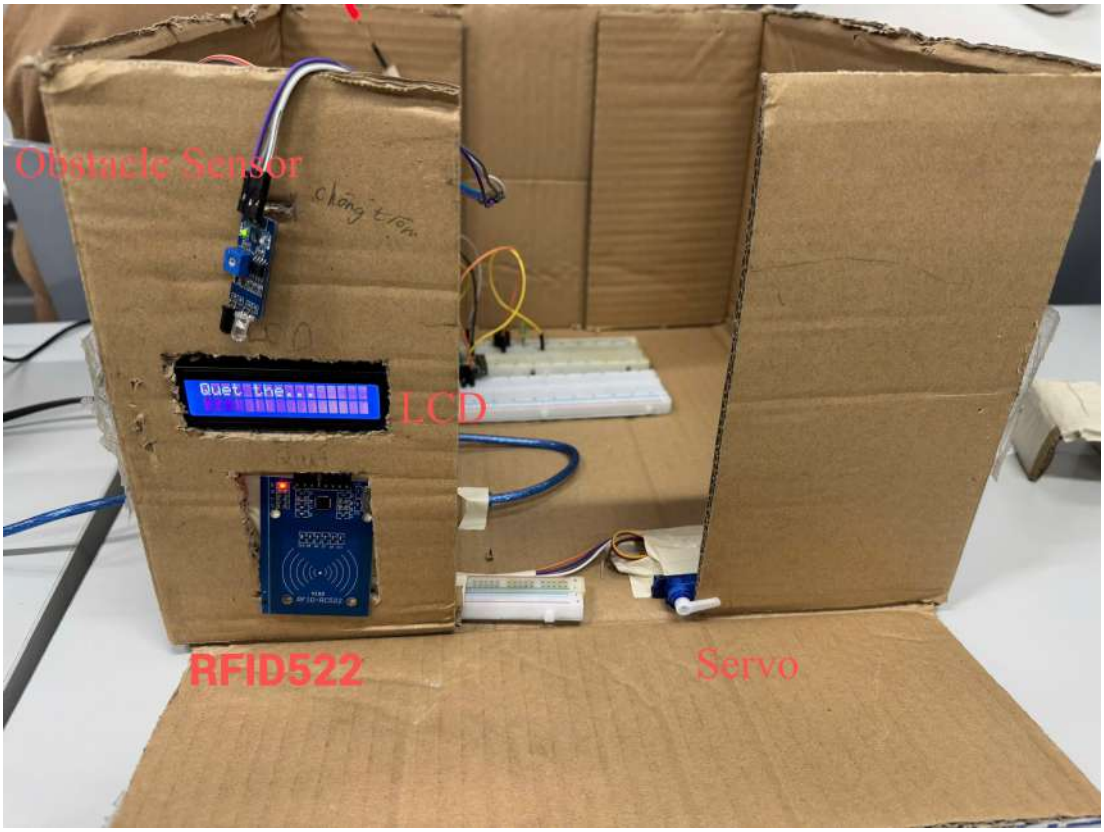


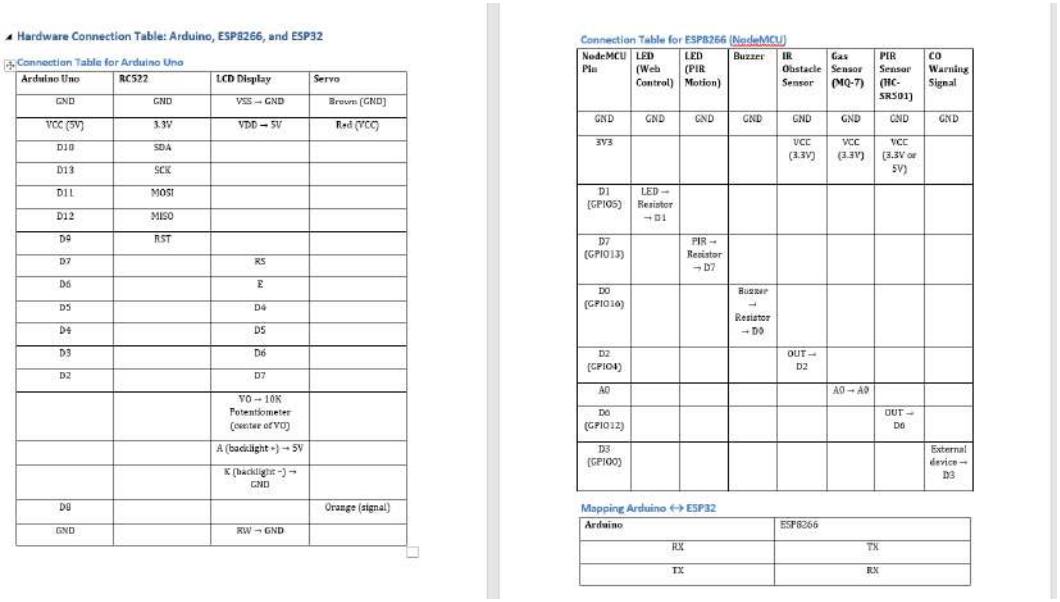
Fig. 2. Second part

II. Background and Literature Review

Smart home systems have evolved significantly with the rise of IoT. Systems like Nest and SmartThings have introduced remote control and automation for lights, thermostats, and locks. However, these solutions often require proprietary hardware or cloud subscriptions.

Open-source platforms such as Arduino and ESP32 allow for more affordable and customizable alternatives. Prior research has demonstrated the feasibility of using Arduino in access control systems with RFID, while ESP32's wireless capabilities make it ideal for networked applications. This project builds upon such literature by integrating both technologies into a cohesive home automation prototype.

III. System Design Overview



Hardware Connection Table: Arduino, ESP8266, and ESP32

Connection Table for Arduino Uno

Arduino Uno	RC522	LCD Display	Servo
GND	GND	VSS → GND	Brown (GND)
VCC (5V)	3.3V	VDD → 5V	Red (VCC)
D10	SDA		
D13	SCK		
D11	MOSI		
D12	MISO		
D9	RST		
D7		RS	
D6		E	
D5		D4	
D4		D5	
D3		D6	
D2		D7	
		VO → 10K Potentiometer (center of VO)	
		A (backlight +) → 5V	
		K (backlight -) → GND	
D8			Orange (signal)
GND		RW → GND	

Connection Table for ESP8266 (NodeMCU)

NodeMCU Pin	LED (Web Control)	LED (PIR Motion)	Buzzer	IR Obstacle Sensor	Gas Sensor (MQ-7)	PIR Sensor (HC-SR501)	CO Warning Signal
GND	GND	GND	GND	GND	GND	GND	GND
3V3				VCC (3.3V)	VCC (3.3V)	VCC (3.3V or 5V)	
D1 (GPIO5)	LED → Resistor → D1						
D7 (GPIO13)		PIR → Resistor → D7					
D0 (GPIO16)			Buzzer → Resistor → D0				
D2 (GPIO4)				OUT → D2			
A0					A0 → A0		
D6 (GPIO12)						OUT → D6	
D3 (GPIO0)							External device → D3

Mapping Arduino ↔ ESP32

Arduino	ESP8266
RX	TX
TX	RX

Fig. 3. System design showing communication between Arduino Uno and ESP32, and connected components.

The proposed system includes two main processing units:

- **Arduino Uno**: Handles RFID-based access control, LCD notifications, and servo actuation.
- **ESP32**: Collects data from sensors and communicates with a mobile/web interface via Wi-Fi.

Interconnected sensors and actuators include:

- PIR motion detector
- CO (carbon monoxide) sensor
- Obstacle detection sensor
- Buzzer and LEDs for alerts

Communication between Arduino and ESP32 is enabled via serial interface, ensuring modular control and data sharing.

IV. Block Diagram and Architecture

Figure 4 shows the overall interaction between hardware components in the system.

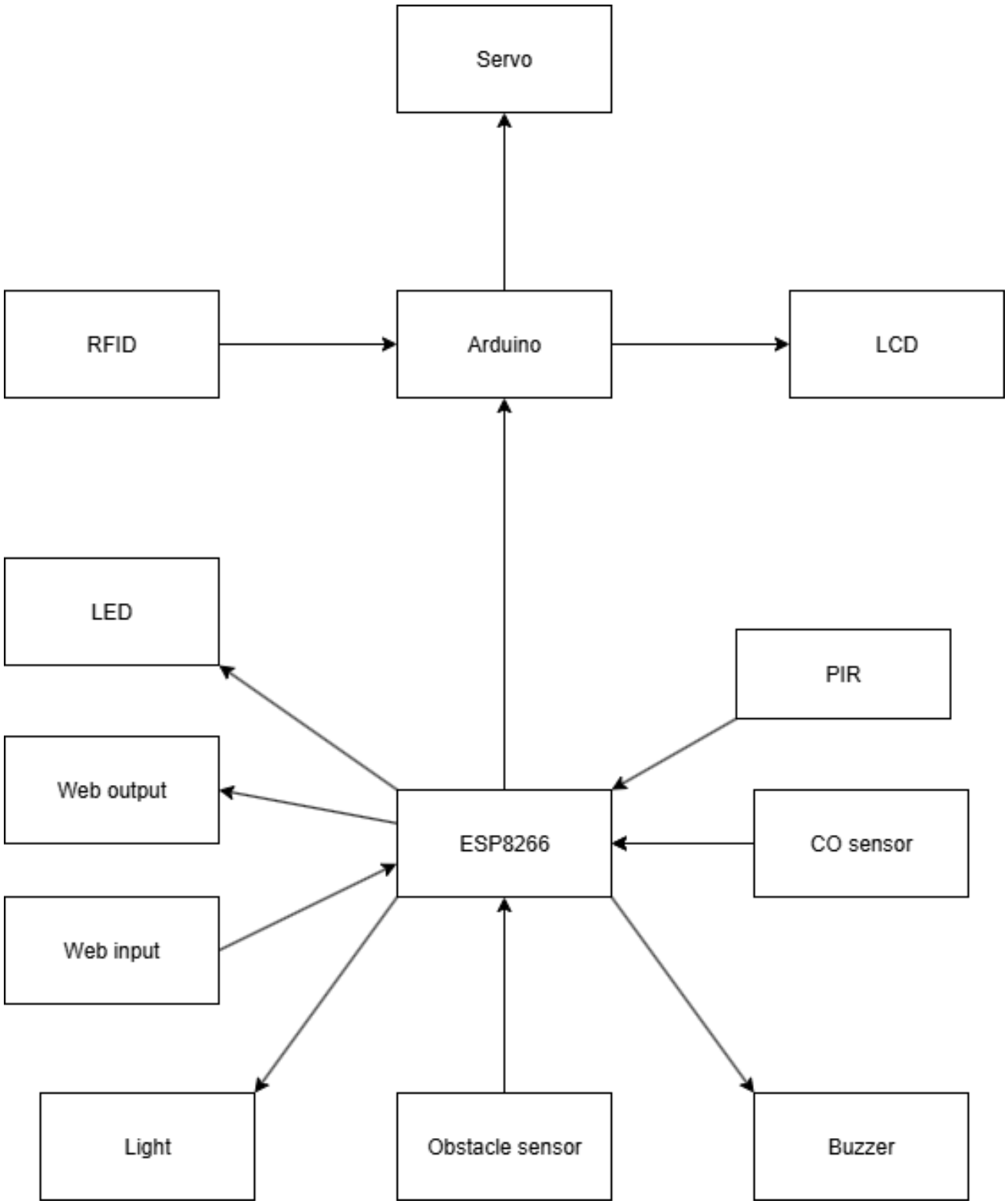


Fig. 4. System block diagram.

The RFID reader communicates with Arduino to verify access. If verified, it triggers the servo motor to unlock the door. Sensor data from the environment is processed by ESP32, which acts accordingly and updates the status via web interface.

V. Hardware Description

A. Component List

- **Arduino Uno** — Main microcontroller for RFID and servo control.
- **ESP32 Dev Kit** — Handles sensing and wireless communication.
- **RFID Reader (MFRC522)** — For access control.
- **Servo Motor** — To control locking mechanism.
- **LCD 16x2 Display** — Displays status of access attempts.
- **PIR Sensor** — Detects motion within range.
- **CO Sensor** — Monitors indoor gas levels.
- **Obstacle Sensor** — Detects physical barriers or movement.
- **Buzzer and LEDs** — Alerts users of unsafe conditions.

B. Circuit Schematic

Figure 5 demonstrates the connection layout between modules.

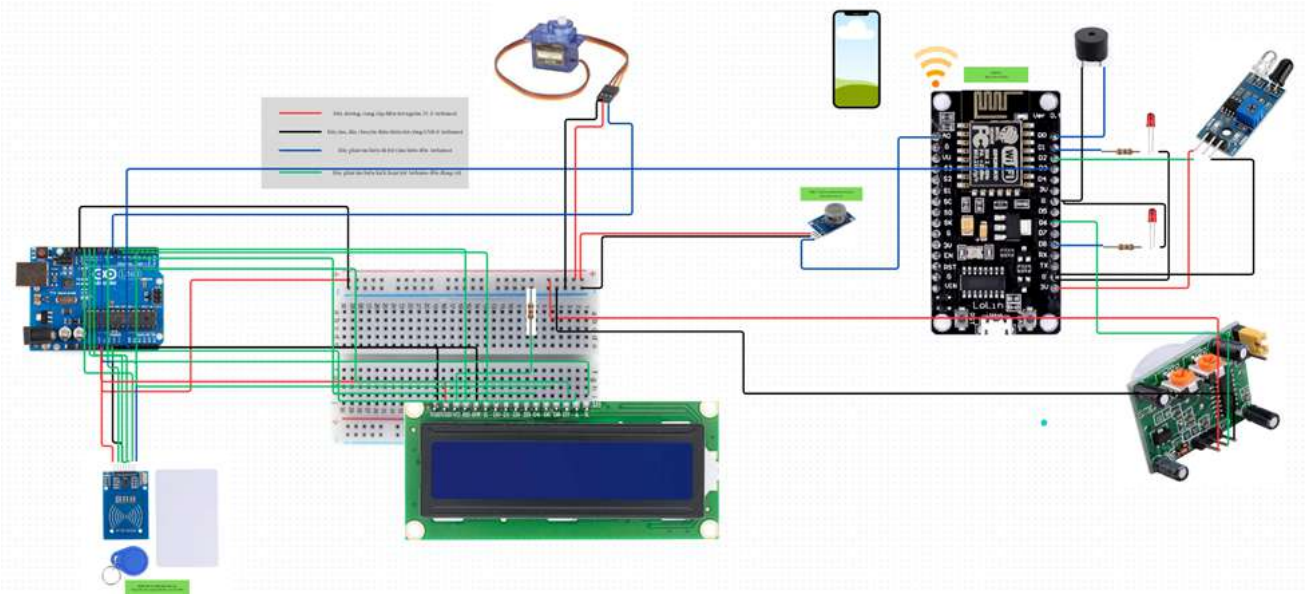


Fig. 5. Hardware wiring diagram.

VI. Software Development

The Arduino code was written in the Arduino IDE using standard libraries for RFID, servo, and LCD communication. The firmware listens for RFID inputs, compares tags with authorized UIDs, and then sends success/failure messages to the LCD and ESP32.

ESP32 firmware, programmed via the Arduino framework, handles:

- Reading values from sensors
- Making decisions based on thresholds
- Hosting a web interface on local Wi-Fi network
- Activating buzzer/LEDs on abnormal events

A. Flowchart

Figure 6 illustrates the software's logic.

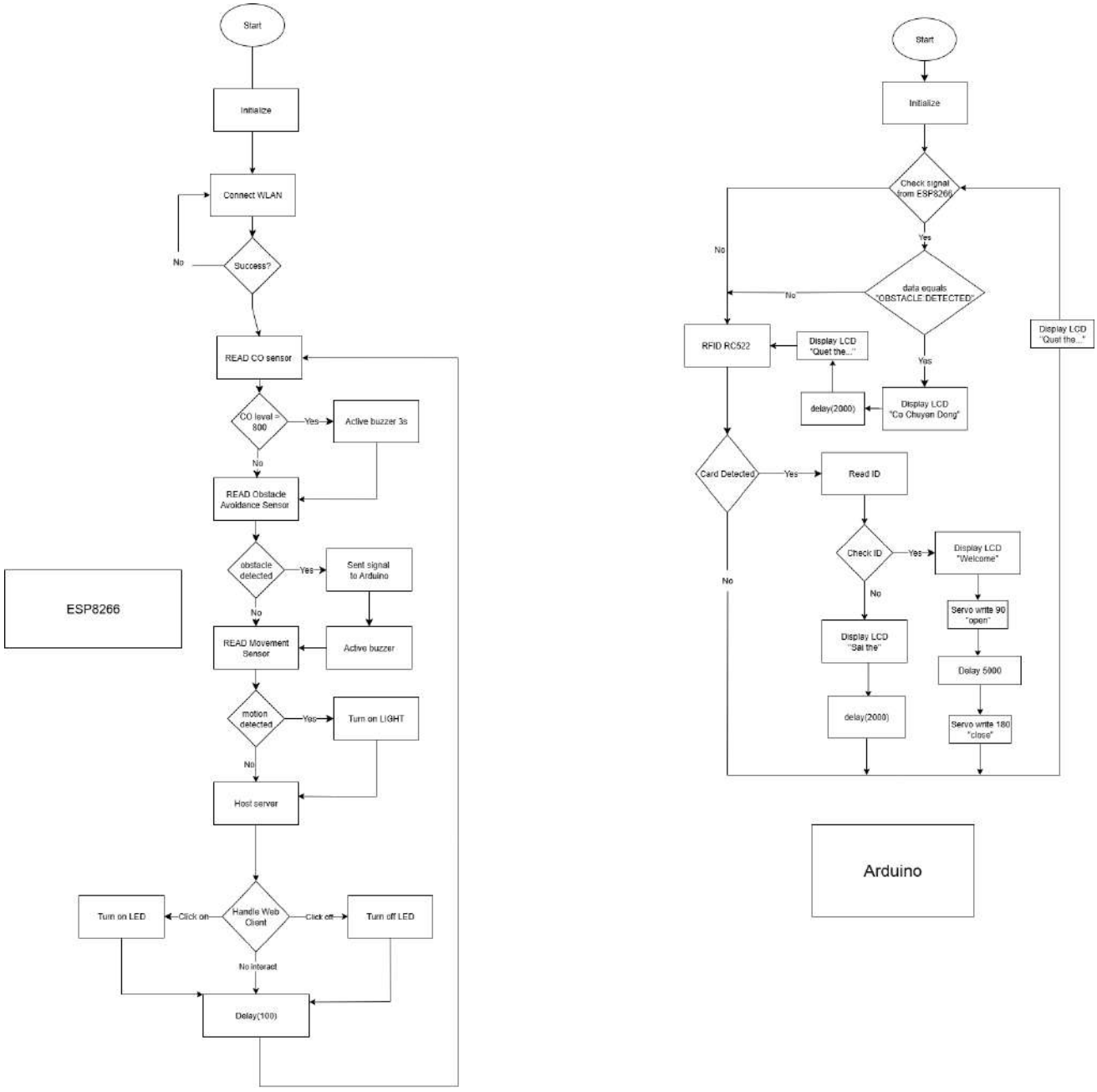


Fig. 6. Software operation flowchart.

VII. Prototype Assembly and Testing

A. Assembly Process

The prototype was constructed on a breadboard. Components were wired according to the schematic. The RFID reader and servo were placed to simulate a door lock, while sensors were fixed at representative positions (e.g., door, hallway).

B. Functional Testing

Multiple tests were performed:

- RFID authentication: Valid tags granted access; invalid tags were denied.

- Motion detection: Triggered ESP32 to start web server.
- CO sensor: Buzzer activated when levels exceeded threshold.
- Obstacle sensor: Alerts user when objects approach critical zones.
- Remote toggling: From web interface, users could activate LEDs.

VIII. Results and Observations

The system successfully performed real-time access control, environmental sensing, and user interface interaction. The response times were within 1–2 seconds, and sensor triggers were consistent.

Issues observed:

- Wi-Fi instability occasionally delayed interface loading.
- RFID reader sensitivity varied with distance and tag orientation.

IX. Discussion

Our smart home prototype demonstrates a practical use of open-source platforms for real-world applications. Its modular nature allows for easy integration of new sensors or features. However, further refinements are needed:

- **Security**: Current system lacks encryption; future versions should use HTTPS or VPN tunneling.
- **Scalability**: For multi-room homes, additional ESP32s or mesh networking should be used.
- **Cloud Storage**: Logging sensor data would help in analytics and long-term monitoring.

X. Reference Document

- **Chat GPT**
- **Youtube**: https://www.youtube.com/watch?v=qyl4SEGSqKM
- **Youtube**: https://www.youtube.com/watch?v=gZ4hLL-SfdA

XI. Conclusion

This project achieved its goal of creating a functional, affordable smart home system using ESP32 and Arduino. The dual-controller architecture enabled real-time automation and responsive remote control. Results validate its use in basic home automation and pave the way for expanded implementations.

XII. Future Work

To enhance system functionality and reliability, we recommend the following future developments:

- Implement two-factor authentication for secure access.
- Add camera modules for video monitoring.
- Transition to cloud-based dashboard with user login.

XIII. Author Contributions

TABLE I
Team member contributions

#	Name	Task	Contribution
1	Tran Dinh Thien	Flowchart, diagrams	25%
2	Nguyen Phu Cuong	System architecture	25%
3	Nguyen Huynh Nhat Thinh	Programming, testing	25%
4	Truong Doan Anh Khoa	Documentation, editing	25%
Total			100%