The German University in Cairo (GUC)
Faculty of Media Engineering and Technology
Computer Science and Engineering
Embedded System Architecture - CSEN 701

IoT-based Noise Monitoring

Team Members :
Khaled Amr, 55-5060

Mohamed Walid, 55-0810

Shahd Mohamed, 55-0551

Ali Hani, 55-0317

Saif Ahmad, 55-12614

Sarah El-Gazzar, 55-1393

Youssof Ahmad, 55-0992

Team Number :
Team 37

Under Supervision of :
Dr. Eng. Catherine M. Elias

Winter 2024

Contents

1 Project Objectives **1**

2 Introduction **2**
2.1 Background 2
2.2 Purpose 2

3 Methodology **3**
3.1 System Design Overview 3
3.1.1 Subsystems 3
3.2 Hardware Components 3
3.3 Driver Functionality Overview 3
3.3.1 `motion_sensor.c` 3
3.3.2 `proximity_sensor.c` 4
3.3.3 `sound_sensor.c` 4
3.3.4 `servo_ctrl.c` 4
3.3.5 `alarm_ctrl.c` 4
3.3.6 `LED_ctrl.c` 4
3.4 Concurrency and Task Logic 4
3.5 Stateflow Diagram 5

4 Results **6**
4.1 System Hardware Implementation 6
4.2 GitHub Repository 10

5 Conclusion **11**

Chapter 1

Project Objectives

- Develop an IoT-based security system that monitors motion, proximity, and noise levels.
- Integrate sensors and actuators for real-time system response.
- Implement concurrency using FreeRTOS for efficient task management.
- Create a reliable and resettable alert mechanism.
- Document hardware and software implementation comprehensively.

Chapter 2

Introduction

2.1 Background

The IoT noise-based monitoring security system integrates sensors and actuators to detect intrusions and trigger alerts. The system leverages the RP2040 microcontroller and FreeRTOS for multitasking, ensuring efficient operation.

2.2 Purpose

The project aims to design and implement a security system that:

- Detects motion, proximity, and sound level changes.
- Responds to threats with servo-controlled gate closure and alarms.
- Resets automatically when no threats are detected.

Chapter 3

Methodology

3.1 System Design Overview

The system comprises sensors and actuators connected to the Arduino Nano RP2040. Concurrency is achieved using FreeRTOS tasks for each subsystem.

3.1.1 Subsystems

- **Motion Detection:** PIR sensor detects movement.
- **Proximity Detection:** IR sensor identifies objects nearby.
- **Sound Detection:** Sound sensor measures sound intensity.
- **Alert Mechanism:** Servo motor, LEDs, and alarm actuators respond to threats.

3.2 Hardware Components

The table below lists all hardware components used in the project, along with their datasheets and wiring diagrams.

Component Name	Datasheet	Wiring Diagram	Driver Name
PIR Sensor	Datasheet Link	Wiring Diagram Link	`motion_sensor.c`
IR Sensor	Datasheet Link	Wiring Diagram Link	`proximity_sensor.c`
Sound Sensor	Datasheet Link	Wiring Diagram Link	`sound_sensor.c`
Arduino Nano RP2040	Datasheet Link		
Servo Motor	Datasheet Link	Wiring Diagram Link	`servo_ctrl.c`
Alarm Buzzer	Datasheet Link	Wiring Diagram Link	`alarm_ctrl.c`
LED	Datasheet Link	Wiring Diagram Link	`LED_ctrl.c`

Table 3.1 – Hardware Components with Datasheets, Wiring Diagrams, and Driver Names

3.3 Driver Functionality Overview

Each subsystem in the IoT Noise-Based Monitoring System relies on dedicated drivers that manage sensor readings and actuator control. Below is a detailed explanation of the key functions provided by each driver:

3.3.1 `motion_sensor.c`

This driver handles the PIR sensor for motion detection:

— `void initMotionSensor(int pin)`: Initializes the PIR sensor on the specified GPIO pin.

— `int isMotionDetected()`: Checks if motion is detected and returns a binary result (1 for motion, 0 for no motion).

3.3.2 `proximity_sensor.c`

This driver manages the IR proximity sensor:

— `void initProximitySensor(int pin)`: Initializes the IR sensor on the specified GPIO pin.

— `int objectDetected()`: Returns 1 if an object is detected within the range of the sensor, otherwise 0.

3.3.3 `sound_sensor.c`

This driver reads and classifies sound intensity:

— `void initSoundSensor(int pin)`: Configures the sound sensor on the specified GPIO pin.

— `uint16_t readSoundSensor()`: Reads the analog value from the sound sensor and returns its intensity.

3.3.4 `servo_ctrl.c`

This driver controls the servo motor:

— `void initServo(int pin)`: Sets up the servo motor on the specified GPIO pin.

— `void setServoAngle(float angle)`: Adjusts the servo motor to the specified angle (e.g., 0° to close the gate, 90° to open it).

3.3.5 `alarm_ctrl.c`

This driver activates and deactivates the alarm buzzer:

— `void initAlarm(int pin)`: Initializes the alarm on the specified GPIO pin.

— `void alarmturnOn()`: Activates the alarm buzzer.

— `void alarmturnOff()`: Deactivates the alarm buzzer.

3.3.6 `LED_ctrl.c`

This driver manages the system's LED indicators:

— `void initLED(int pin)`: Initializes an LED on the specified GPIO pin.

— `void ledturnOn(int pin)`: Turns on the LED connected to the specified GPIO pin.

— `void ledturnOff(int pin)`: Turns off the LED connected to the specified GPIO pin.

3.4 Concurrency and Task Logic

The system employs FreeRTOS tasks for concurrency:

— **Motion Sensor Task:** Continuously reads the PIR sensor status to detect motion.

— **Proximity Sensor Task:** Continuously reads the IR sensor status to detect if an object is within proximity range.

— **Sound Sensor Task:** Reads analog sound levels and classifies intensity.

— **LED and Alarm Control Task:** Manages response to sensor readings.

This structure ensures non-blocking operation and real-time responsiveness.

3.5 Stateflow Diagram

The stateflow diagram below illustrates the behavior of the system under various conditions. It details the transitions between states based on sensor readings and the corresponding actuator responses.

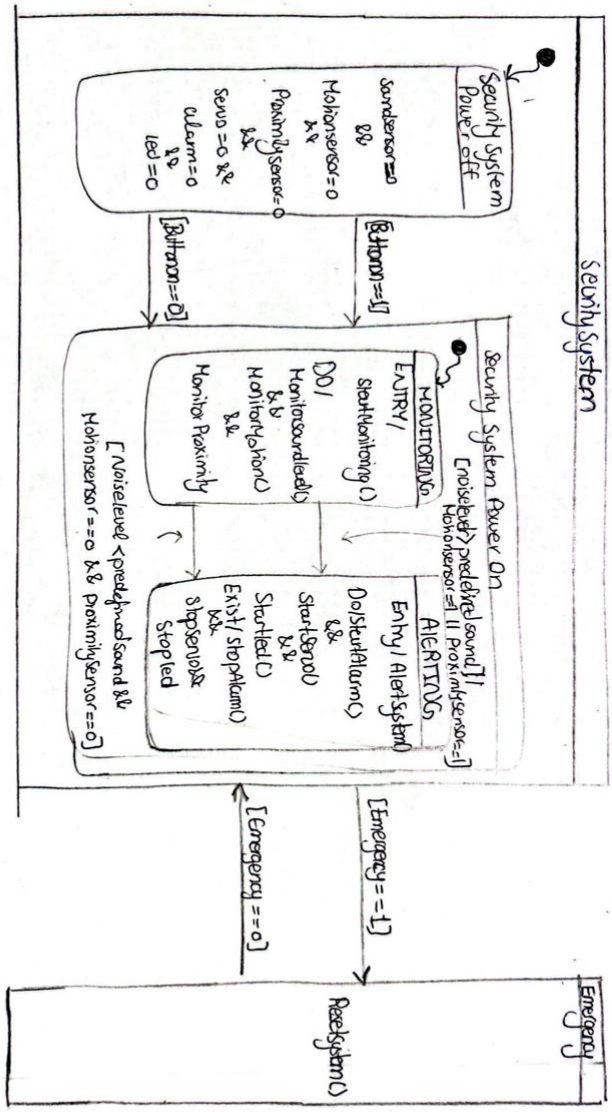


Figure 3.1 – Stateflow Diagram of the IoT Noise-Based Monitoring System

Chapter 4

Results

This section presents the results of the IoT Noise-Based Monitoring System, showing the final hardware implementation .

4.1 System Hardware Implementation

The image below showcases the final hardware setup of the system, including all sensors and actuators connected to the Arduino Nano RP2040.

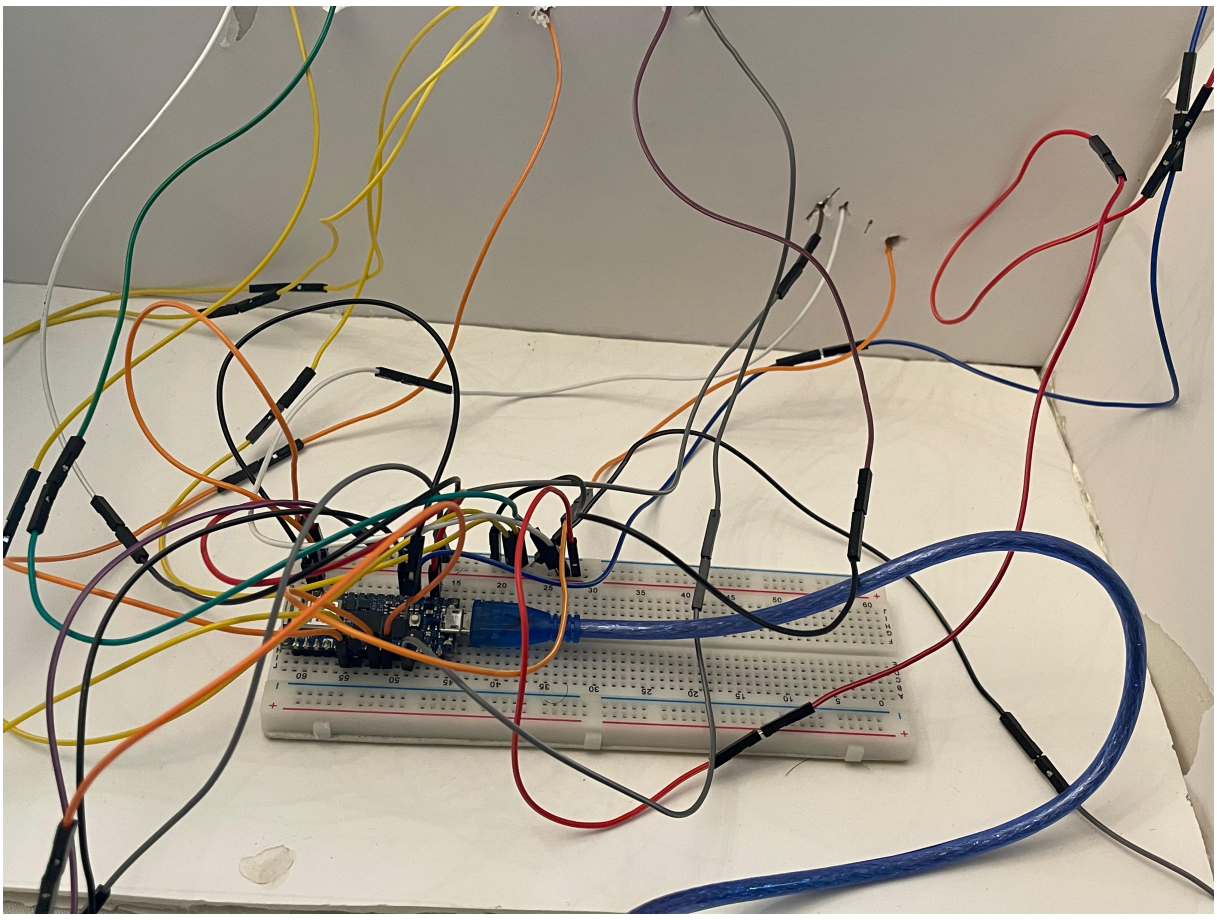
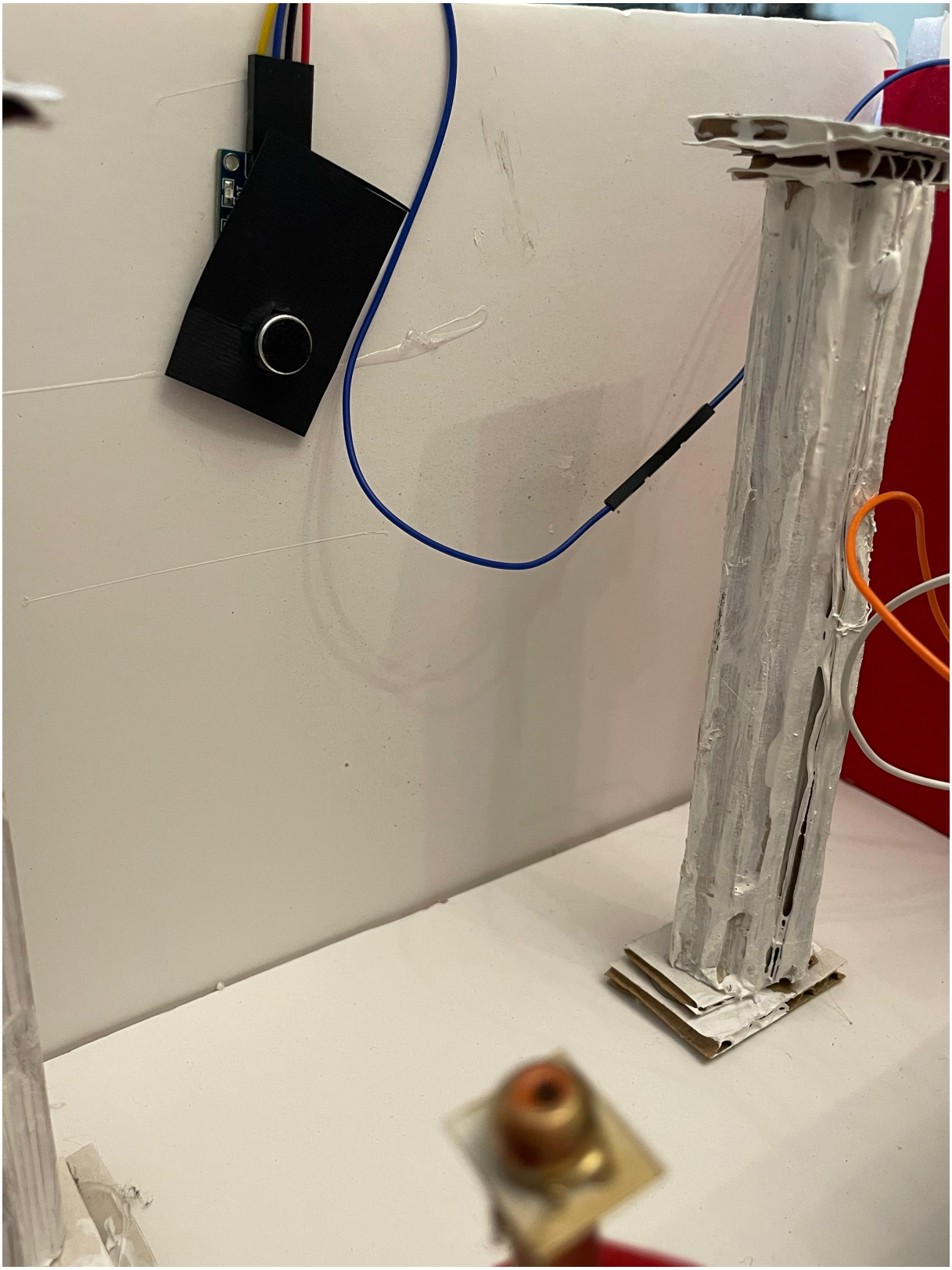
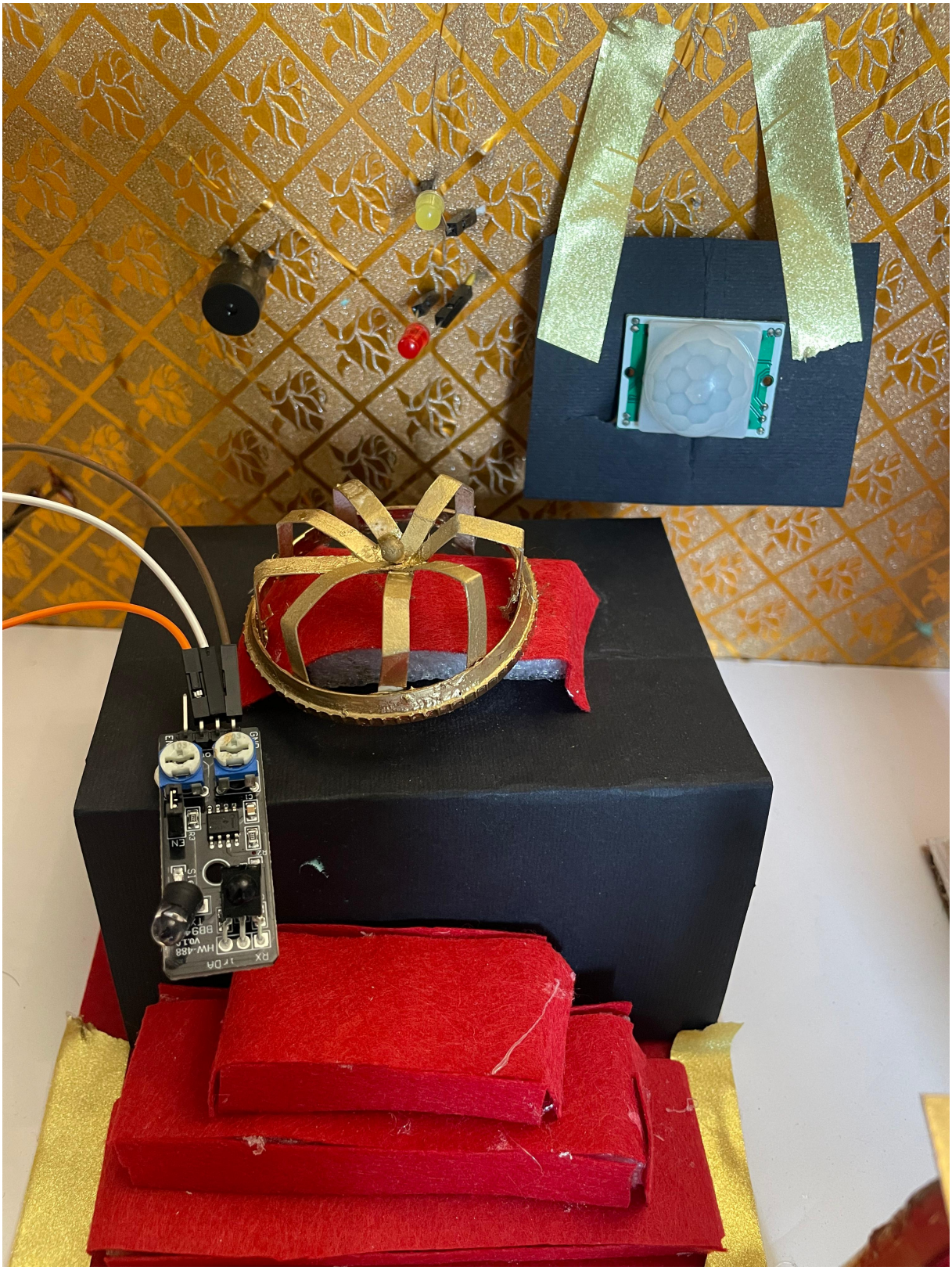


Figure 4.1 – Final Hardware Implementation of the IoT Noise-Based Monitoring System





4.2 GitHub Repository

The complete source code for the IoT Noise-Based Monitoring System. You can access the repository using the link below:

— **GitHub Repository Link:** https://github.com/mohameddwalid/rp2040-freertos-template.git

Chapter 5

Conclusion

The IoT Noise-Based Monitoring System successfully integrates multiple sensors and actuators with the RP2040 microcontroller to create an efficient and responsive security solution. By leveraging FreeRTOS for concurrency, the system achieves real-time responsiveness while ensuring non-blocking operations for each subsystem.

The project demonstrated the effective use of hardware components, such as the PIR sensor, IR sensor, sound sensor, servo motor, LEDs, and alarm buzzer, to detect motion, proximity, and sound intensity. The automated reset functionality further enhances the system's usability by ensuring a consistent operational state. This work highlights the potential of IoT-based solutions in security applications.

Overall, the project achieved its objectives and provided a robust foundation for further development in IoT-based security systems.