Deakin University

Embedded Security System

Technical Report

Ben Marriner
21-5-2021

Table of Contents

Overview

Background

Security Systems are commonly used in many buildings including homes, offices, banks, government and other buildings that house sensitive information. The system can consist of many devices used together to monitor areas in and around a building to ensure that unauthorised entry and activity does not occur. There are many devices used in a security system and can for example, range from motion sensors and cameras to alarms both audible and silent.

Problem Statement

The objective of this project was to solve the real-world problem of keeping the property and assets of one secure. To achieve this, a prototype of a security system was developed that can be controlled manually and via Wi-Fi. It has the capability of being extended with new devices, while also being scalable for the needs of any client. It is also versatile, which means it can be deployed in any sealed off environment such as a home, building or outdoor area to name a few.

Requirements

- 9 x M-F Jumper Wires
- 18 x M-M Jumper Wires
- Arduino PCB-mounted Buzzer
- Arduino PCB-mounted Button
- 3 x Red LEDs
- 1 x Blue LED
- 10kΩ Resistor
- 3 x Arduino PIR Motion Sensors
- Particle Argon
- 2 x Breadboards

Design Principles

Versatility

To be a viable solution for a wide range of clients, the prototype needed to be capable of being set up in all kinds of sealed off environments. This can range from small buildings such as offices and stores, to larger buildings such as research and industrial facilities and also government buildings.

Scalability

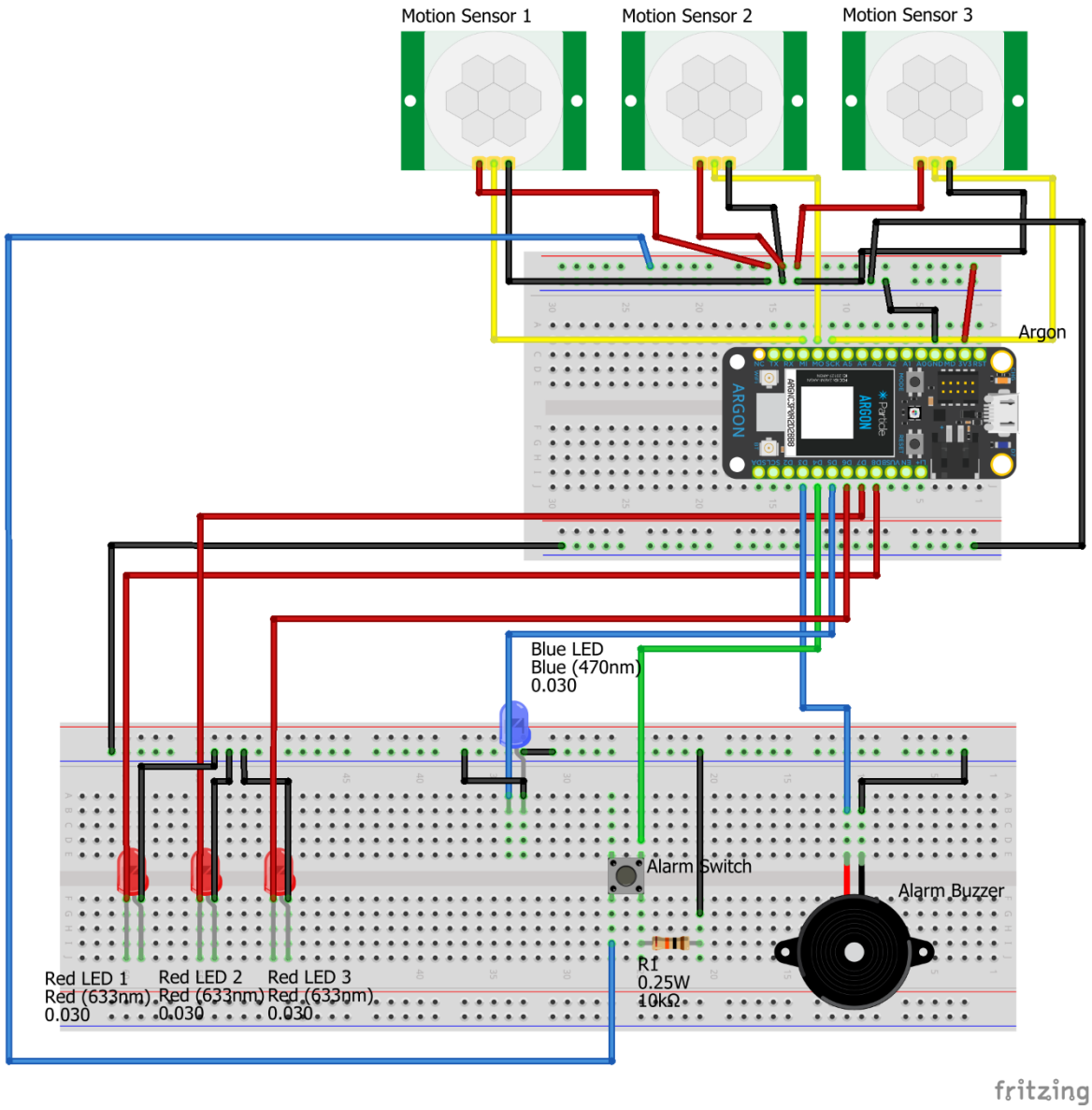
The security system prototype should be able to function in small environments as well as larger ones. For example, the owner of a home may wish to set up a simple security system with a motion sensor placed on the front veranda to prevent intruders from entering the house. A business owner may wish to secure their building and would need a wide range of sensors, authentication devices (keypads, card readers) and cameras. This security system would be more elaborate compared to a simple home security system. Finally, a larger building such as that owned by a corporation may need many devices connected in a security system to secure sensitive information stored in multiple locations.

Extendibility

The installation of new sensors and devices for use in the security system should be made easy for the client. Some clients will own businesses which are naturally intended to expand. The client's existing security system should be capable of doing the same thing. They should be able to integrate new security devices into their system with ease as their business grows physically.

Prototype Architecture

Prototype Breadboard Schematics



Particle Argon Pin Configuration

	RST		
Power for Motion Sensors	3.3		
	MD		
Circuit Ground	GND		
	A0	Li+	
	A1	EN	
	A2	USB	
	A3	D8	Red LED 1
	A4	D7	Red LED 2
	A5	D6	Red LED 3
Motion Sensor 3	D13 (SCK)	D5	Blue LED
Motion Sensor 2	D12 (MO)	D4	Alarm Switch
Motion Sensor 1	D11 (MI)	D3	Alarm Buzzer
	D10 (RX)	D2	
	D9 (TX)	D1 (SCL)	
	NC	D0 (SDA)	

Particle Argon Software

GitHub Repository

https://github.com/Goolog/PSS

Global Constants & Variables

```
// LED Constants
const int RED_LED_1 = D8;
const int RED_LED_2 = D7;
const int RED_LED_3 = D6;
const int BLUE_LED = D5;

const int LEDS[] = { RED_LED_1, RED_LED_2, RED_LED_3, BLUE_LED };
const int LEDS_SIZE = 4;

// Motion Sensor Constants
const int MOTION_SENSOR_1 = D11;
const int MOTION_SENSOR_2 = D12;
const int MOTION_SENSOR_3 = D13;

const int MOTION_SENSORS[] = { MOTION_SENSOR_1, MOTION_SENSOR_2, MOTION_SENSOR
_3 };
const int MOTION_SENSORS_SIZE = 3;
const int MOTION_SENSOR_BAUD_RATE = 9600;

// Alarm Constants
const int ALARM_SWITCH = D4;
const int ALARM_BUZZER = D3;
const int ALARM_SOUND_FREQ = 1000;

// Button pressed state
bool lastButtonState = false;

// Is the alarm system activated/deactivated
bool sysEnabled = false;
```

Functions & Methods

LEDs

```
// Turn on a specified LED
void turnOnLED(int pin)

// Turn off a specified LED
void turnOffLED(int pin)

// Turn of all LEDs in the system
void turnOffAllLEDs()

// Toggle an LED between on and off
void toggleLED(int pin)
```

Alarm Buzzer

```
// Sound the alarm for the given number of seconds
void buzz(double seconds)
// Sound the alarm indefinitely
void buzz()

// Silence the alarm if it is buzzing
void silence()
```

Motion Sensors

```
// Return true if motion is detected by the specified sensor
bool motionDetected(int sensorPin)
```

Alarm Switch

```
// Returns the switch of the state such that the state will be true for only o
ne loop
bool getSwitchState()
```

Security System

```
// Toggle the enabled state of the security system
void toggleSystem()

// Test all the hardware connected to the system
void testHardware()

// Initialise Argon to correct pin modes
void initHardware()
```

Testing Approach

Hardware

During the development of the prototype, the hardware connected to it needed to be tested to ensure it functioned correctly. To accommodate for this, a function was written in the Argon's program code that tests all the devices connected to it. During the initialisation process of the system, the argon will execute a series of tests via the testHardware() function. It will sequentially test the 4 LEDs and the alarm buzzer by making them blink on and off and make a noise respectively. The motion sensors are also tested; however, it requires that the argon be connected to a computer with a serial connection open to it. This serial connection is required to test the sensors because there is no real way of indicating whether they are working aside from using the console to read out a message. The button is also not tested as there is no way to test the button without the user pressing it.

Sensor Functionality

The motion sensors used for this prototype come with two dials that control their delay and sensitivity to motion. The LEDs were used to detect how susceptible the motion sensors were to motion. Each sensor in the code is assigned an LED. When motion is detected, the LED will light up and the alarm buzzer will sound.

User Manual

Setting up the Hardware

Assemble the prototype using the two breadboards and Particle Argon as shown in the breadboard schematics. See the requirements section for all the necessary materials.

Installing the Software

1. Plug in the Particle Argon to a computer
2. Go the Particle Web IDE and open a new document
3. Copy and paste the code from GitHub into the document
4. Verify and flash the code to the Argon

Features

- Upon running the software, the program will initialise itself and all the hardware connected to it. It will run the required test procedures to test the devices connected to it. You should see all the LEDs flash sequentially and then the buzzer will make a brief sound. If you do not see this, then there is either a misconfiguration with one of the pieces of hardware or the component(s) are non-functional. Your computer may tell you that the Argon has disconnected. This is not the case; it is just testing the sensors. If you open a serial monitor on your computer while this initialisation process is taking place, you will be able to determine if the sensors are working.
- **Red LEDs:** This prototype has 3 Red LEDs. Each LED will light up if its corresponding sensor detects movement.
- **Alarm Buzzer:** This will make a sound when one of the connected sensors detects movement. It can be silenced by disabling the system.
- **Alarm Switch:** Pressing the alarm switch button will toggle the system on and off. The blue LED will tell you if the security system is enabled or disabled. Pressing the button to disable the system will silence the alarm and turn off any LEDs that were switched on.
    - **LED on =** Security system enabled
    - **LED off =** Security system disabled

Conclusion

The Embedded Security System project is one of the most complex projects I have completed. This was due the system itself being more elaborate to design compared to other systems I have assembled in the past. I did encounter a few problems while assembling the system and writing the software for it.

1. Due to the current worldwide shortage of silicon, I could only find one more PIR motion sensor that was identical to the one I already owned. This explains why the third motion sensor I used in my prototype is different to the one in the schematic. All three sensors are functionally the same, nevertheless.
2. The alarm switch was causing the Argon to reset itself when pressed. This was probably due to the amount of voltage returning to the Argon. It was fixed by placing a 10kΩ resistor between the button's ground pin and a ground pin on the breadboard.
3. The alarm buzzer would either make a faint sound or no sound at all. This was due to an incorrect signal being sent to the buzzer via the wrong function. There is a specific wire function called tone() that sends sound signals to buzzers for them to make sounds.

Ultimately, this prototype was a good start to what an embedded security system could look like and how it can solve the problem of physical security. For future prototypes, the following improvements could be made:

1. Multiple Particle Argon devices could be used to group together sets of sensors and devices. They could then be used to connect to a master Argon device or even a Raspberry Pi. This could act as the mainframe for controlling the entire security system.
2. A printed circuit board for connecting sensors to the Argon could be used.
3. Wireless sensors could be used to minimise the number of jumper wires being used.