Introduction to Engineering Design with Professional Development 1

Final Report for
Emergency Detection System

Team: SightRPI
Section: 3
Instructors: Hisham Mohamed / Casey Jakubowski

Version 1.0
June 2017

Prepared by
Alexandra Fearn (ME, 2019)
Hanyuan (Cornelius) Xiao (EE/CSE, 2019)
Junjie Ding (CSE, 2019)
Ziyi Lu (EE/CS, 2019)
Zhen Chen (EE, 2019)

Executive Summary

The emergency detection system that SightRPI has designed is unique from products currently on the market since it features hands-free automatic fall detection and automatic notification methods using an iPhone application. This system features IR break-beam sensors mounted at a specific height to detect the presence of someone in the room. The height these sensors are mounted at is chosen with the goal of not being able to detect small children and pets. The range sensor will begin to read heights once someone has entered the room. If the range sensor detects that one has fallen, the system will ask if they are okay. If the system receives a “no” response or does not receive a response within a certain time frame, then the system will go into emergency mode. The emergency contact will be alerted through our iPhone application. If the system receives a response of “yes”, the emergency response will not be triggered.

In addition to these key features, our system also includes many options to satisfy user preference, such as the ability to change the light’s color.

We have done extensive research into elderly who experience falls. We also have benchmarked against similar emergency detection systems on the market and have examined these products in order to further understand what our customer requirements should be. This research showed that our system must be able to automatically detect falls using hands-free methods and send notifications of emergency automatically to emergency contacts. Furthermore, our system must be able to detect only adults and children over 3 years of age. Finally, our system must also include user preference options.

The prototype that we have created for the sake of this proof of concept has been tested against the technical specifications that we have set forth. The prototype that we have created met all of these specifications. For future iterations, we will implement angled sensors to cover more area in a room and we will also implement voice-recognition. Additionally, we will upgrade from Bluetooth as our main communication method to using a virtual private server.

Table of Contents

Revision History

Version	Date	Name	Reason for Changes
0.0	6/19/17		Initial document creation
1.0	6/27/17		All sections completed
1.1	6/28/17		Document edited for grammar and voice
1.2	6/29/17		Final version

Introduction

All of us have seen commercials for medical alert devices that elderly can buy to get help when they fall. Very few people actually realize that one in three elderly will experience a fall this year, and almost half of them will not be able to get up on their own [11]. It has also been reported that 90% of American older adults wish to stay in their home as they grow older [2]. However, with all that comes the risk of someone's loved one experiencing a fall and having no one there to help them.

Products on the market are nearly all wearable devices with an emergency button and they do not always feature automatic fall detection, as described in upcoming sections. There is the possibility that the user will forget to wear their device or will not be able to press the emergency button when needed. Thus, the SightRPI Team has designed a product that is hands-free and can automatically detect falls. Furthermore, this product will automatically notify emergency contacts in the event that the user is not okay after a fall.

This entire project is based on one simple need of the customer: to be able to get help when they fall. A more detailed list of customer requirements can be found in Appendix A.

This document will provide detailed explanation of the objectives of this project and the customer requirements. It will also describe how customer requirements were interpreted into technical specifications. An assessment of existing products on the market and professional and societal contributions will also be discussed. The system concept development and selection is also featured. On top of that, each subsystem will be discussed including analysis and design of each. Finally, results and technical accomplishments will be discussed and conclusions will be made. This document also includes references and many appendices that feature specific topics related to this project.

Project Objectives & Scope

Mission Statement

Benefits	This project will provide an effective way to let an emergency contact know that an elderly person (user) has fallen down and cannot get up.

Goals:	To create a device that is able to automatically detect when user falls without requiring them to wear a device or do a specific action. Another goal is to be able to let an emergency contact know through our iPhone application that the user has fallen.
Primary Markets:	Elderly living at home or caregivers (such as adult children of the elderly)
Secondary Markets:	Hospital or nursing home
Assumptions:	• The user is initially standing and then falls • The room that our device is installed in has furniture in it and the user cannot be standing in the corners of the room (blind spots) • The user is in the area covered by our sensors
Stakeholders:	Team members, IED instructors, competing medical alert companies

Customer Requirements

The target customer for our system would be elderly people who are seeking ways to make aging at home more feasible. While this product may be appealing to caregivers and adult children of elderly people, our target customer is going to be simply elderly people for the scope of the proof of concept. Additionally, a residential setting is an ideal model due to the fact that the area of the room is generally smaller and there are not many people in one common space at one time (usually). Furthermore, most falls occur in the home [3].

Table 1: Customer Requirements Summary

	Product is easy to use.		**Product is durable.**
*****	No wearable device needed	***	Does not need to replace anything constantly
X	No professional installation needed	***	Does not broken through repeated use
****	Easy to read current status from the smartphone application.	*	It is weather resistant
*****	No training needed to use our product	X	System is water proof
*	System can be relocated.	****	System can be used regularly
***	System can be used in all kinds of rooms	***	System has a long life span
***	System is easy to reset	x	System can prevent mice bites
			Product is convenient.
	Product is safe to use.	***	Can control the light from long distance
****	No information leak (phone number, home address)	**	Can change the color of the light
***	It is stable, will not drop from ceiling	***	Can change the brightness of the light
****	Product will not cause a fire	****	Application will not crush
****	Electronics are properly insulated.	****	Does not ask repeatedly when user want to line down
****	Product will not shock the user	****	Does not detect pets in the room
***	No sharp parts can be reached by user	*****	Does the notification real time

Technical Specifications

Based on the research we conducted, we could conclude that there are basic requirements for our target customers. They are: accuracy, reliability, comfort level and user friendly. Table 2 contains the basic requirements we gathered and their desired technical specifications.

Table 2: Customer Requirements Related to Technical Specifications

Requirement	Technical Spec.	Target Value
Accuracy	Range sensor error	<3 mm

	Response time for range sensor	<100ms
	Time delay for emergency contact receive notification	<2s
	Success rate to detect the fall	>90%
Reliability	Life span	>2 years
	No need to charge	AC power supply
Comfort level	Color change	RGB range: 0-255
	Brightness	100-200 lux
	Application crush rate	<0.1%
User-Friendly	Number of wearable device needed	0
Safety	Voltage through the circuit	<5.5v

Assessment of Relevant Existing Technologies

We researched many medical alert devices that are currently on the market. We compared their features to the features of our own design and made sure that our design was meeting the customer requirements that we have specified.

Consumer Reports published a comparison chart of many of the most popular medical alert devices that are currently available. This comparison is shown below in Figure 1 [5].

Facts to consider	Life Alert	LifeStation	Medical Alert	MobileHelp	Philips Lifeline	Rescue Alert
Monthly service cost						
Landline/Cellular	$50/$60	$26/$33	$30/$35	NA/$35	$30/$42	$29/$43
Features						
Mobile 911 phone	Yes	Yes	Yes	No	No	Yes
Automatic fall detection	No	No	Yes	Yes	Yes	No
Fees						
Minimum obligation	36 months	30 days	90 days	None	None	None
Activation	$95	None	None	None	$0 to $50	None
Cancellation fee	Remainder of contract	None	None	None	None	$0 to $25

Figure 1: Comparison of many medical alert devices

Nearly all of the devices on the market require the user to wear a device and push a button to get help. Additionally, not all of these devices can automatically detect falls. This is not a method that we would have liked to pursue since there is the possibility that the user may forget to wear the device. Additionally, the user may or may not be conscious and alert enough after their fall to press the button to signal that there is an emergency.

In addition to that, nearly all of these devices featured in the comparison chart require a monthly service cost. On top of that, some even include an activation and cancellation fee. The requirement of the monthly service fee being paid is another way that the

system could fail (e.g. if the user does not pay the fee or forgets to pay it on time). This is something we took into consideration when creating a solution.

One of the specific products that we looked into was the Philips Lifeline system, which is the market leader currently with 750,000 customers as of 2015 [17]. Philips offers a model of their LifeLine product that includes automatic fall detection and a model that does not include automatic fall detection [15]. The system is shown below in Figure 2 [16].

Figure 2: Philips LifeLine System

The LifeLine system is connected to a landline phone and is how operators communicate with the user in an emergency. Once the fall has been detected, an operator communicates with the user through the LifeLine system. The model of LifeLine that does not include automatic fall detection still includes the confirmation system. Specific directions to be used in the event that the user does not respond to the operator can be specified [15].

This product does feature a confirmation system, which we agreed is something that we should implement into our system to decrease the amount of false positives. This product also requires the user to wear the necklace so that in the event of an emergency, they can press the button to signal that they need help. Our technical specifications indicate that we will not implement this method into our design since as stated, not everyone is conscious or alert after they fall and able to press the button.

Another product that we investigated was GreatCall's Lively Wearable medical alert device. This device is a wristband that the user wears. The device features the standard button the user must press in the event of an emergency in addition to fall detection. The Lively Wearable device works with an app on the user's smartphone through Bluetooth. This app also can be downloaded by family members and alert them when

the emergency button is pressed. The app also features fitness tracking [9]. This device can be seen flow in Figure 3 [9].

Figure 3: GreatCall Lively Wearable medical alert device

This was a unique feature since the system was able to alert family members in addition to call center agents. We decided that this was a feature (or a similar one) that we wanted to include in our product.

On GreatCall's website they specify that this device relies on a CR2450 Lithium Coin battery. Relying on the battery always being charged is another way that the system could fail [9]. This is another reason why we did not decide to pursue a wearable device.

Another device we investigated was ADT's Medical Alert system. ADT's system is very similar to Philips Lifeline and many of the other devices that Consumer Reports talked about since it features a "help button" on a wearable pendant. They also offer a pendant that features automatic fall detection, but still recommend that the user presses the help button in the event of an emergency if they can [1]. This device is shown below in Figure 4 [1].

Figure 4: ADT Medical Alert system

The products that we researched have helped us conclude that our product is unique. This research has also helped us decide what technical specifications we did want to include and did not want to include. Our findings are summarized in the Existing Technologies Benchmarking (Table 3) below.

Table 3: Existing Technologies Benchmarking

Product	**Description**	**Relation to our product**
Philips LifeLine	Wearable device, includes automatic fall detection, includes confirmation system, requires user to press emergency button	Automatic fall detection, confirmation system
GreatCall Lively Wearable	Wearable device, requires user to press emergency button, GPS capabilities, connects with app through Bluetooth	Communication with smartphone application through Bluetooth
ADT Medical Alert	Wearable device, offers models with and without automatic fall detection, requires user to press emergency button, waterproof	Automatic fall detection, alerts in case of emergency contacts automatically

Professional and Societal Contributions

According to aarp.com, 79% of older adults specified that “personal system that allows for one to call for help in emergencies” as a housing feature that is important to them. The AARP also found that 70% of seniors made home modifications due to safety concerns. Additionally, 60% did so in order to increase the chances that they would be able to live independently [2]. Other medical alert systems on the market require the user to wear the device and some even require the user to press a button to signal that they are in need of assistance. These systems will not be able to help the user if they are not wearing it. Some of these systems, specifically ones that do not have automatic fall detection, will not be able to help the user if they are not conscious/alert enough to press the button after they fall. We believe that our product can combat these issues since our product can automatically detect a fall using hands-free methods.

We also recognize that our product may interest nursing homes, hospitals, office buildings, and many more types of organizations. A nursing home may wish to install our product in each of their patient’s rooms to be able to detect an emergency faster and locate this emergency more efficiently. A hospital may wish to install our product in some of their rooms for the same reason. An office building may wish to install our product since some people have their own office and may not always be checked on by someone.

Safety is something that we are considering for future iterations of our system. Since we are implementing a lighting system, we will design a system that is up to industry standards and is safe lighting. Additionally, in future iterations we will be implementing a virtual private server in order to create a more secure and private user information transmission process.

System Concept Development and Selection

In the beginning, we brainstormed four possible solutions to detect if a person is falling down without wearing some portable device and four ways to make the notification. These solutions are shown in the Figure 5. We chose to use range sensor, infrared break-beam sensors, and phone application in the end. Additional concept selection matrices are included in our subsystems and will be discussed in upcoming sections.

Find a way to detect a medical emergency that does not require lifestyle changes or invasive methods.		
Fall detection	***Motion detection***	***Notification***
Xbox Kinect	Ultrasound	SMS
Range sensor	Infrared	Phone Application
Camera	Camera	Sound
Robot		Light

Figure 5: Concept Combination Table

Since we have agreed on what methods we will use in the system, we had to work on how we apply them. In the early stages, we just wanted to install one infrared break-beam sensor on the door to detect if someone enters the room, but then we determined we cannot make a difference between entering and leaving the room. To solve this issue, we installed two infrared break-beam sensors as shown in Figure 6 below. The position for the range sensor is significant since we need accurate readings, therefore we decided to cut a rectangular hole on the ceiling and put a foam board right above the hole. The foam board on the above of the ceiling is shown in the Figure 7.

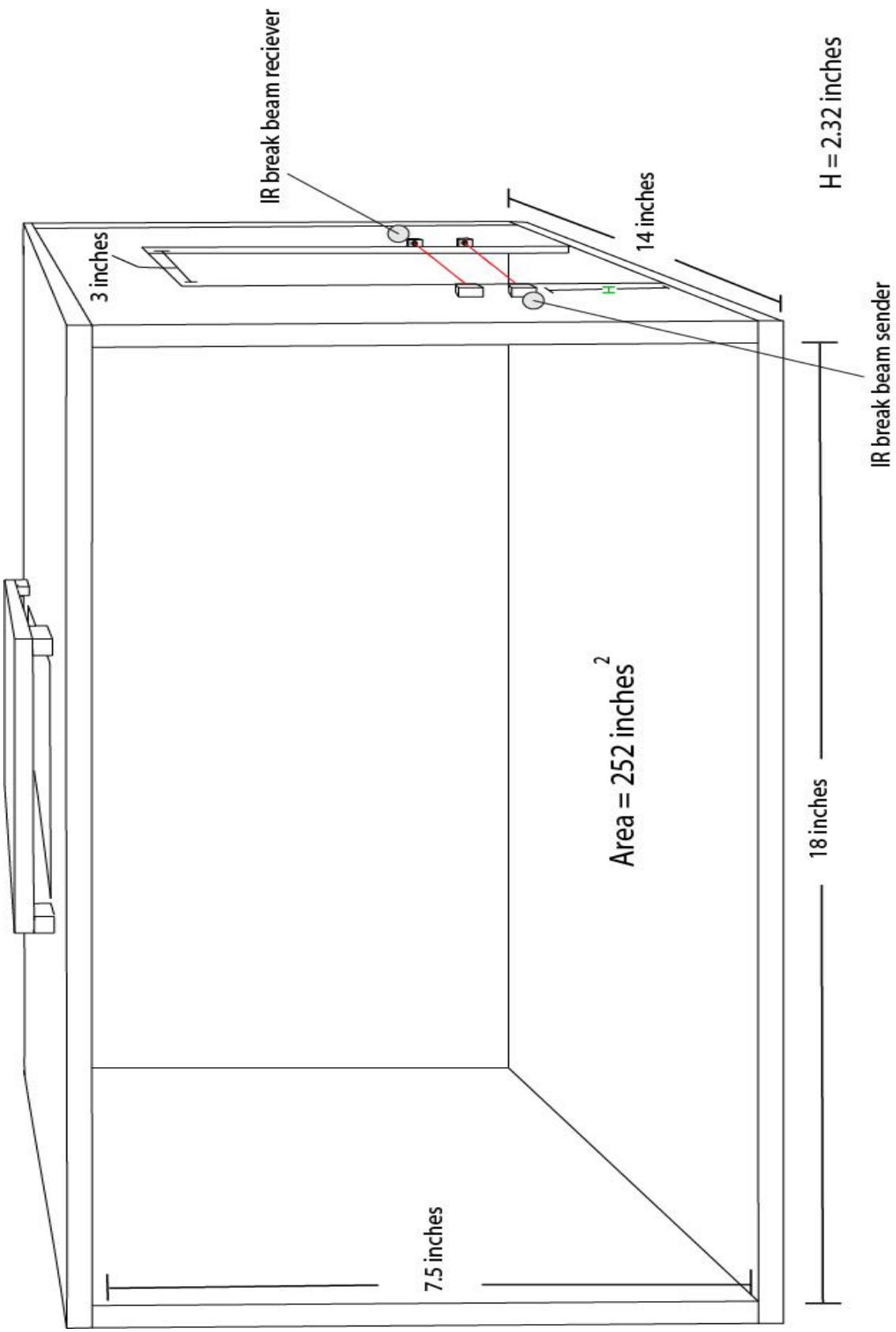

IR break beam reciever
3 inches
14 inches
H = 2.32 inches
IR break beam sender
Area = 252 inches 2
18 inches
7.5 inches

Figure 6: Infrared break-beam sensors as seen in demo room

On that foam board, we have both range sensor and RGB light as shown in Figure 7.

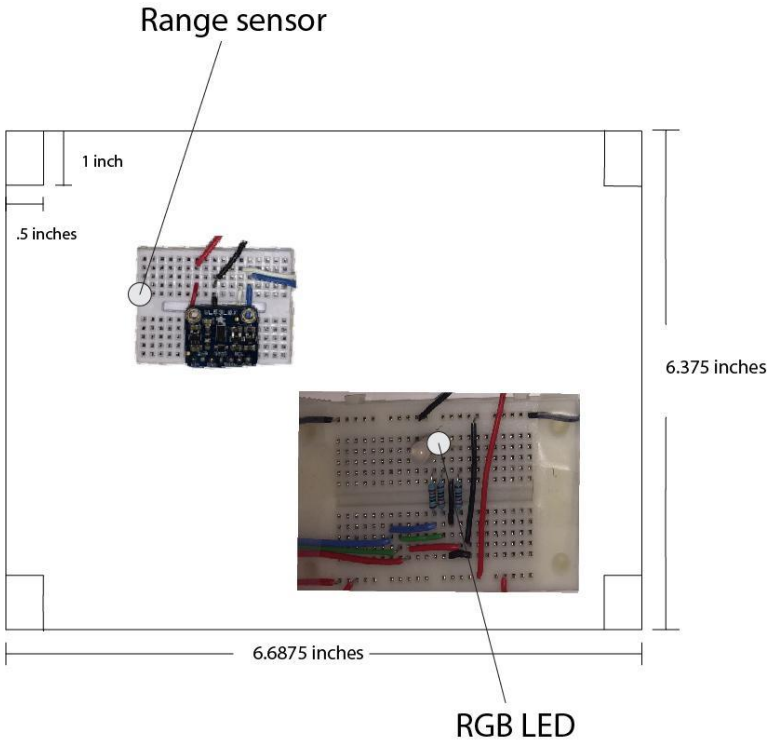


Figure 7

After we knew where to put all the sensors and RGB light in the demo room, we needed to consider how to connect our electrical components to the Arduino UNO. Since Arduino UNO only has very limited pins on it, we tried to use an Arduino Mega. But HM-10, which is our Bluetooth module, did not work on the Arduino Mega. The Arduino UNO is implemented to correct this issue. We carefully sketched a circuit diagram (Figure 8) so that we would not run out of pin before we started building our circuits.

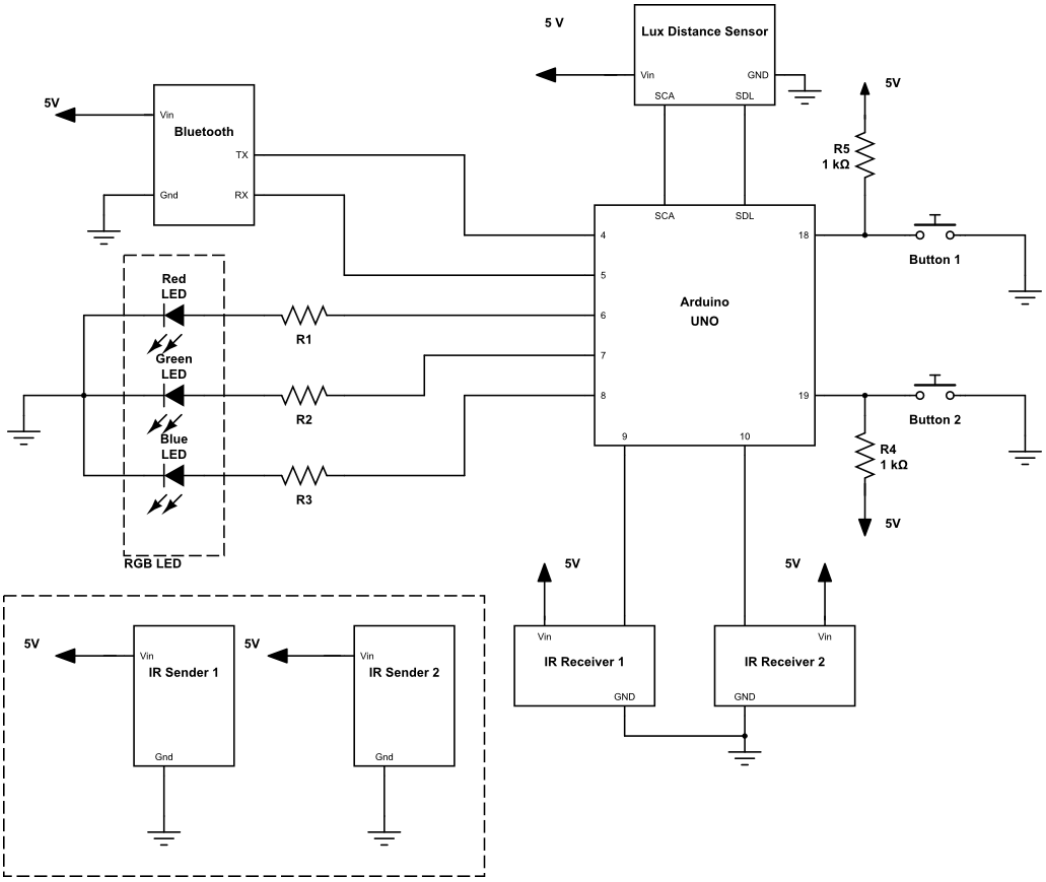


Figure 8: The Circuit Diagram

Subsystem Analysis and Design

The project aims at building a smart fall detection system that can address fall detection cases as well as have entertaining and smart functions for users. Our project is divided into five subsystems and are distributed evenly to each team member. The five subsystems are: Demo Room, Circuit Design & Test Case, Fall Detection & Confirmation System, Wireless Module and Smartphone Application. Each subsystem and its owner are listed in the chart below (Figure 9):

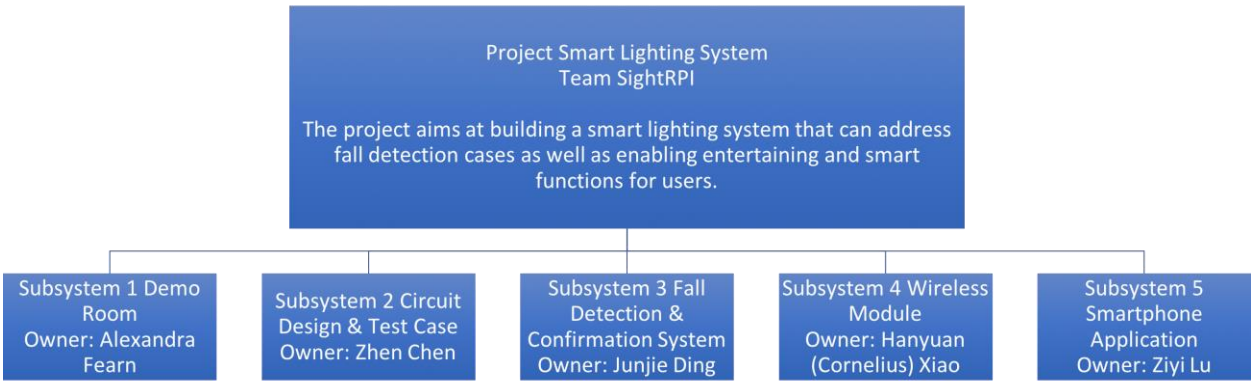


Figure 9: Subsystem breakdown

Subsystem 1: Demo Room

Designing a demo room would be the best way to showcase our prototype due to time constraints and budget constraints. Additionally, a demo room would be the best way to test some of the extreme cases such as a pet or baby being in the room instead of an adult.

That being said, the demo room needs to feature many of the same elements that a common room in a house would have, such as doors, windows, furniture, and lights. The material of the demo room was selected based on cost, feasibility, whether or not it will interfere with our sensors, how easy it is to work with, and if it makes sense for our scale. The concept selection matrix for the material of the demo room is shown below in Table 4.

Table 4: Demo Room Material Concept Selection Matrix

	Demo Room Material			
	Wood	**Plastic (3D Printing)**	**Foamboard**	**Sheet Metal**

Selection Criteria				
Will not interfere with sensors	-1	1	1	-1
Cost	1	-1	1	-1
Feasibility	1	-1	1	-1
Easy to work with	0	-1	1	-1
Makes sense for our scale	1	-1	1	0
Sturdy	1	1	0	1
Sum of +1's	4	-1	6	-2
Sum of 0's	3	-2	5	-3
Sum of -1's	2	-3	4	-4
Net Score	3	-2	5	-3

The most important thing that needed to be considered was whether the material would interfere with our sensors or not. Metal can interfere with the sensors reading and yield inaccuracies [13]. Therefore, sheet metal was assigned a -1 immediately. Wood was also assigned a -1 since assembly of a wood model would require nails or screws, which are most often found to be metal. A plastic material and foam board were assigned 1 since they did not contain metal and therefore would not interfere with our sensor.

Cost was something that we also considered since we are students and also wanted to allocate as much funds as possible to other parts of the system. Sheet metal and plastic (3-D printing) were assigned -1 since both of those materials can cost a significant amount. Wood and foam board were assigned a 1 since they are relatively inexpensive.

When considering feasibility, we assigned scores based on whether the material would be easy to obtain or not. Again, plastic and sheet metal scored -1 since both of them are not as easy to obtain as wood or foam board. On a similar note, sheet metal and plastic again were assigned -1 for easiness to work with. Sheet metal requires special tools to cut it, and a plastic model would require a model designed in CAD and then printed on a 3D printer, which is something the team does not have immediate access to. Wood was assigned a 0 since it is not difficult to work with, but it is not perfect either. Foam board was assigned a 1 since it is very easy to work with due to how light it is and how easy it is to cut it.

Materials were also evaluated on whether they made sense for our scale. Sheet metal does not make sense for our scale because it is typically sold in large quantities and was therefore assigned -1. A plastic model does not make sense for our scale either since not all 3D printers can print an item that is the size of our model. Plastic was assigned a -1 for this reason. Wood and foam board were assigned 1. Both of these

materials make sense for our scale since they are sold in various quantities and would not be too heavy for the team to transport.

Finally, materials were evaluated on sturdiness. Wood, plastic, and sheet metal all scored 1 for this criteria since they are all generally sturdy and would not collapse under a small weight (if applied). Foam board was assigned a 0 for this criteria since in order for it to be sturdy, it must have a certain thickness.

All things considered, foam board scored the highest based on all of our criteria. Based on this fact, we selected foam board to be the material for our demo room. We feel comfortable with this choice since Professor Connor and students of the RPI Light Center use this material to demo their designs as well.

Since the goal of the demo room is to model a real common room in a house, we decided that our room should be to scale and account for standards set forth in the International Building Code [6]. Furthermore, we decided that our demo room should consider the average size of a common room in an American home, which we found to be 223 square feet for a living room and 296 square feet for a family room. Since both of those spaces are common areas, we decided to use the average 259.5 square feet [7].

Using a 1:12 scale would be the best scale for our purposes. This size can provide a large enough demo room that the audience can see what is happening. This scale is also large enough where the audience will be able to see and understand our system. An Illustrator drawing of our demo room is shown below in figure 10.

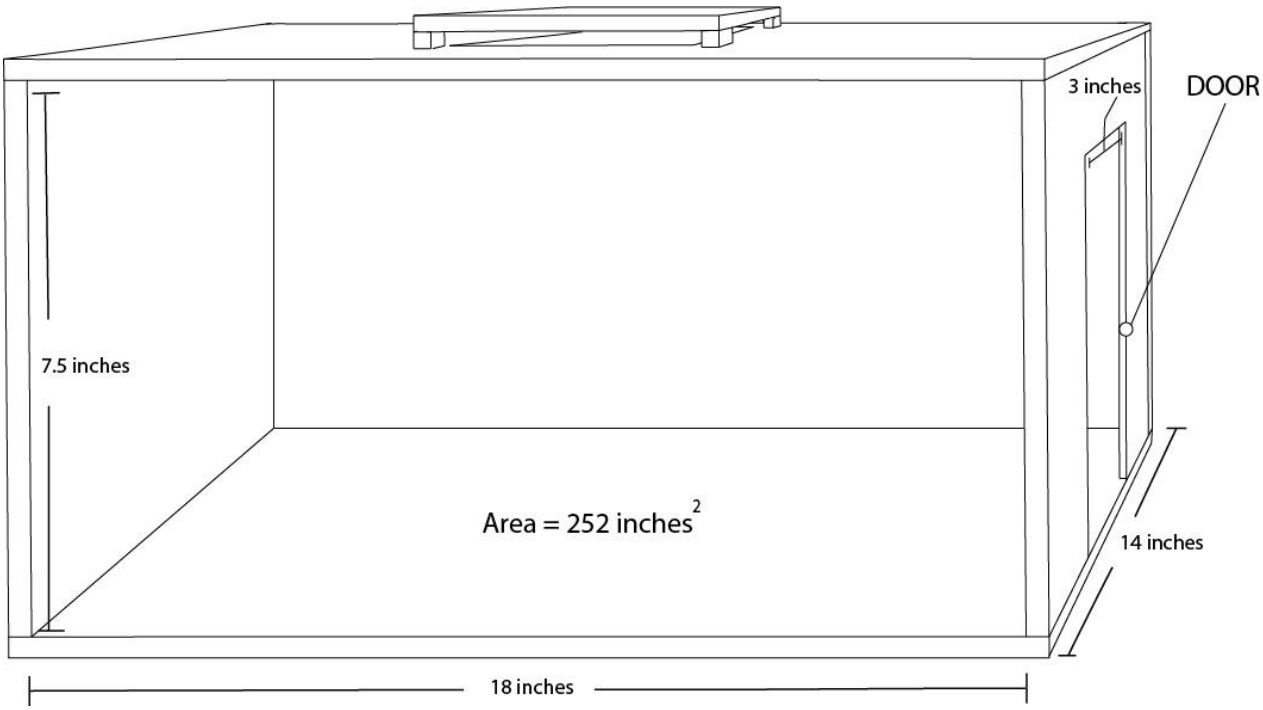


Figure 10: Demo room dimensions

Since our demo room is to scale, we would also need human models that are to scale. We chose human models based on the reported average height of humans according to the CDC. These are found in Table 5 below [4].

Table 5: Average Human Heights [4]

	Average Height (inches)	**Height according to our scale (inches)**
Men	69.3	5.775
Women	63.8	5.316
Average	66.55	5.546

The human models that we will use were purchased from Toys R Us. They were measured with a caliper in-store and were chosen since they most closely matched the human figure height that we were looking for. The male figure (Mike) is 5.68 inches tall, which is slightly below the average reported height of males in America. The female figure (Abbie) has been found to be 5.47 inches tall, which is slightly above the average reported height of females in America.

We have included a child human model (Siri) because our technical specifications include being able to detect a child that is 3+ years old. The height of Siri is 2.7 inches,

while the average height of children (3 years of age) is around 2.9 inches (on our scale) [4]. Since being able to detect a child greater than 3 years of age is one of our technical specifications, we added a .2 inch think piece of foamboard to Siri's feet in order to make her the correct height for our demonstration.

The human models are shown below in Figure 11 [18].

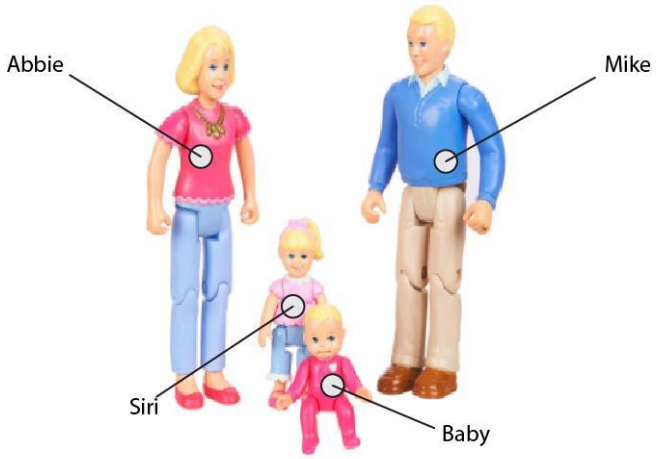


Figure 11: Human models used in demonstration

The demo room has been designed with certain test cases in mind. Our test cases include two extremes in addition to a normal situation with a human adult or child. These two extremes are detecting the difference between a pet or a baby and an adult human.

The pet model we will use is a plastic dog model that was purchased from Michael's. The dog model is 2.167 inches tall and is to scale. The height of the model was measured from the floor to the top of the tail, as shown in the figure below (Figure 12).

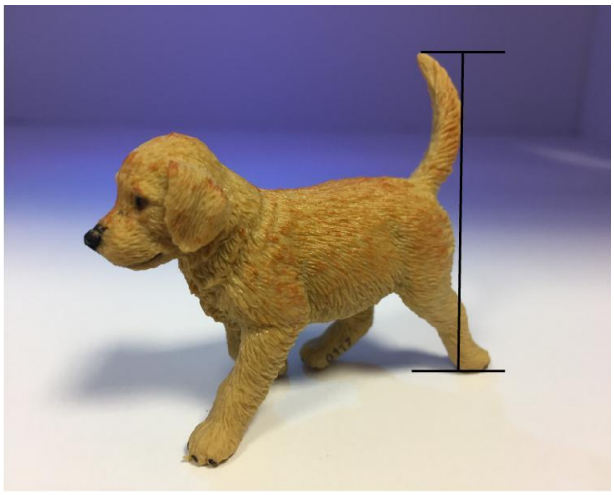


Figure 12: Dog model used in demonstration

It is also worth noting that for the sake of our proof of concept, we will be using only one range sensor due to time constraints. However, in future iterations, more sensors will be installed to cover more area. Initial calculations are shown below for how many sensors would be needed to cover an area of 252 feet (average size of a common area in American home) [7].

Because a range sensor only covers a small pinpoint, another option would have to be implemented in order to cover more area in the demo room for future iterations.

Preliminary calculations indicate that our room would need to have 27 sensors to cover most of the area with minimal blind spots along the perimeter and in the corners. This is too many sensors to install and could be complicated as well as costly.

Preliminary calculations indicate that installing range sensors at an angle in all four of the corners of the room (assuming it is a square like our demo room) could decrease the amount of sensors required to cover an area and increase the simplicity of our system. The Illustrator drawing below demonstrates this concept (Figure 13).

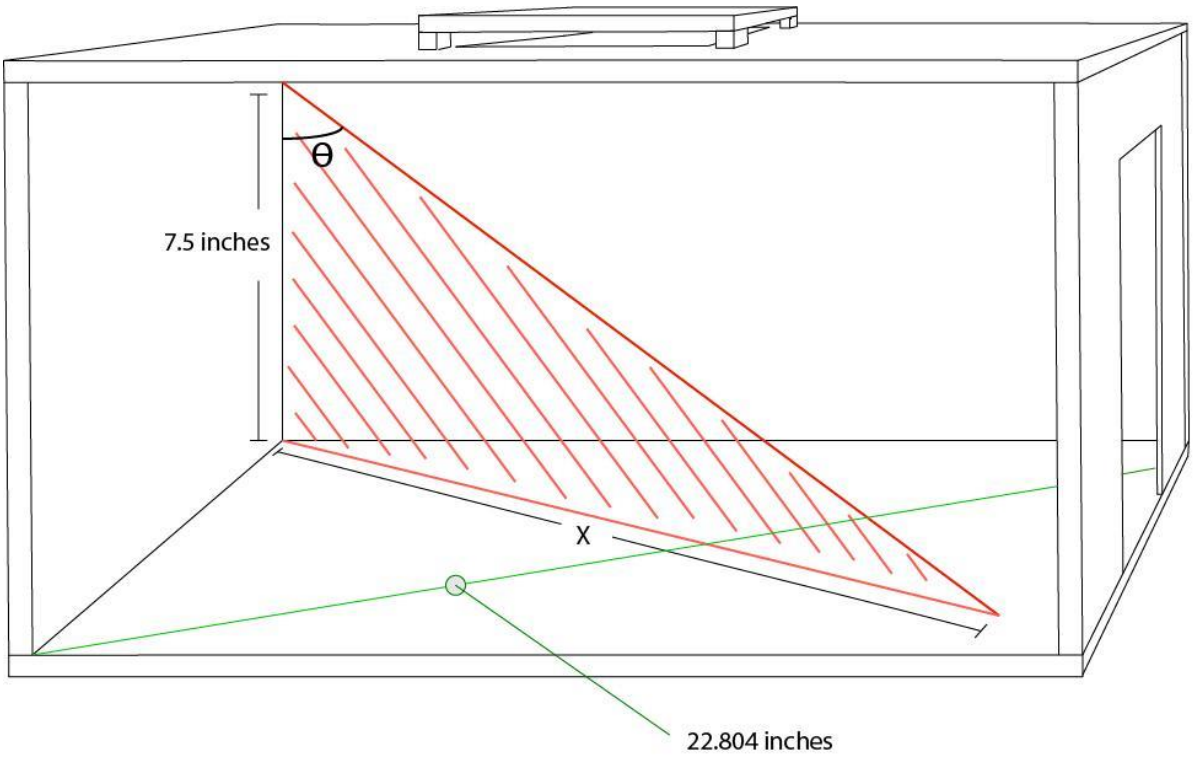


Figure 13: One angled sensor in a corner of the room

As shown in the figure, the diagonal length across the demo room is 22.804 inches (as indicated by the green line). Table 6 shows the length covered for a certain angle. The angle that we have selected is shown in green. We did not choose 80 degrees for our angle since it would yield a distance that is too large and would never make contact with the floor. We also are assuming that the perimeter of the room contains furniture and that there is no need to cover that space.

Table 6

θ (degrees)	**X (inches)**
10º	1.322
20º	2.73
30º	4.33
40º	6.29
50º	8.94
60º	12.99
70º	20.6

80º	42.53

Since using only four sensors in the corners will have blind spots, we will also implement more sensors along the center of the walls at the top to cover even more area (shown by the green dots). The Illustrator drawing below demonstrates this concept for sensors placed to cover the length of the room (Figure 14).

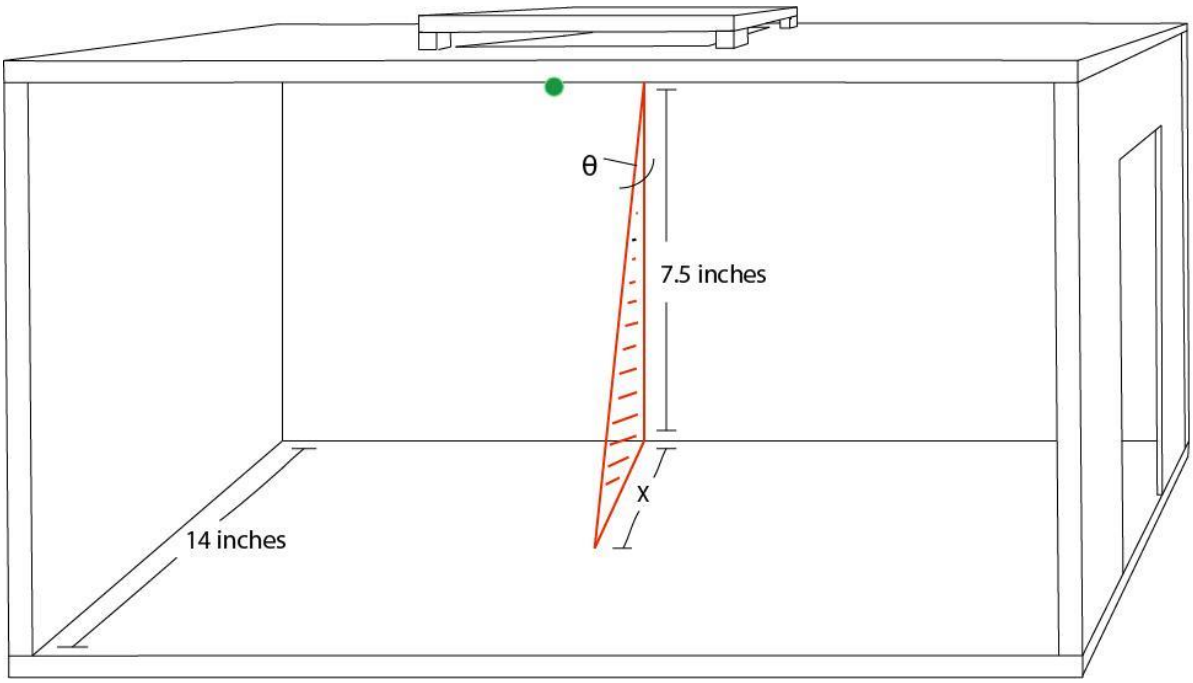


Figure 14: One angled sensor in center of demo room wall

The Illustrator drawing below demonstrates sensor placement and angle for the width of our demo room (Figure 15). While one is shown in the figure, additional sensors are shown by a green dot.

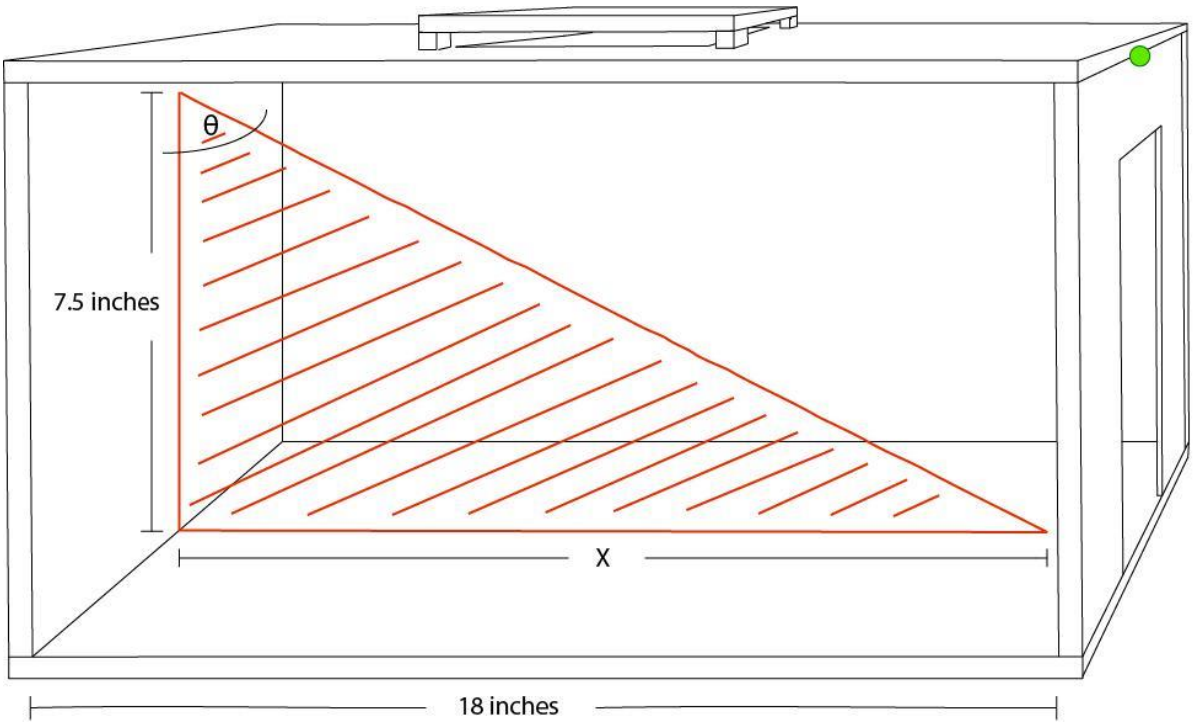


Figure 15: One angled sensor in center of demo room wall

These sensors will be at a different angle since the diagonal length is different from the room's width and length. These calculations are shown below in Table 7. The angle we selected (shown in green) was selected because the length of our room is 14 inches. However, the sensors can be set at different angles for different distances.

Table 7

θ (degrees)	**X (inches)**
10^o	1.32
20^o	2.73
30^o	4.33
40^o	6.29
50^o	8.94
60^o	12.99

If all of the sensors are installed at the angles that were discussed, preliminary calculations indicate that 8 sensors would be needed to cover the area of our demo

room with some blind spots. Further calculations and other options will have to be explored will have to be done to determine if this is indeed the best option.

Subsystem 2: Circuit Design & Test Case

For building a fall detection emergency system, using the proper hardware components in the circuits is important. Because of the size difference between our demo room and an actual room, the components implemented are different.

The processing chip we used is an Arduino, a single-board microcontroller. In the market, Arduino and Raspberry Pi are the two types of board fitting our project: inexpensive and small. They are compared below in Table 8. We finally chose Arduino (Figure 16 and Figure 17) because it's less costly, easier to use, smaller and uses less power.

Table 8: Arduino UNO and Raspberry Pi Comparison

	Arduino Uno	**Raspberry Pi Model B**
Price	$30	$35
Size	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
Memory	0.002MB	512MB
Clock Speed	16 MHz	700 MHz
On Board Network	None	10/100 wired Ethernet RJ45
Multitasking	No	Yes
Input voltage	7 to 12 V	5 V
Flash	32KB	SD Card (2 to 16G)
USB	One, input only	Two, peripherals OK
Operating System	None	Linux distributions
Integrated Development Environment	Arduino	Scratch, IDLE, anything with Linux support

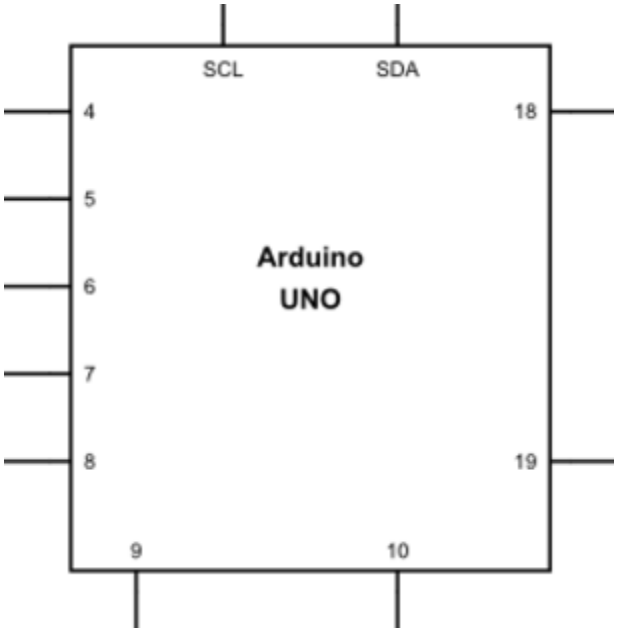


Figure 16: Schematic of Arduino

We used 13 pins on the Arduino: SCL, SDA, 4,5,6,7,8,9,10,18,19, +5V and GND. Below is a description of what each pin is used for (Table 9).

Table 9: Pin Number Guide

Pin Number	Name of the Wire (Color)
4	TX (Orange)
5	RX (Yellow)
6	Red RGB (Red)
7	Green RGB (Green)
8	Blue RGB (Blue)
9	IR 1 (Yellow)
10	IR 2 (Yellow
11	
12	
13	
14	
15	
16	
17	
18	Push Button Yes (Green)

19	Push Button NO (Brown)
SDA	(Blue)
SCL	(White)

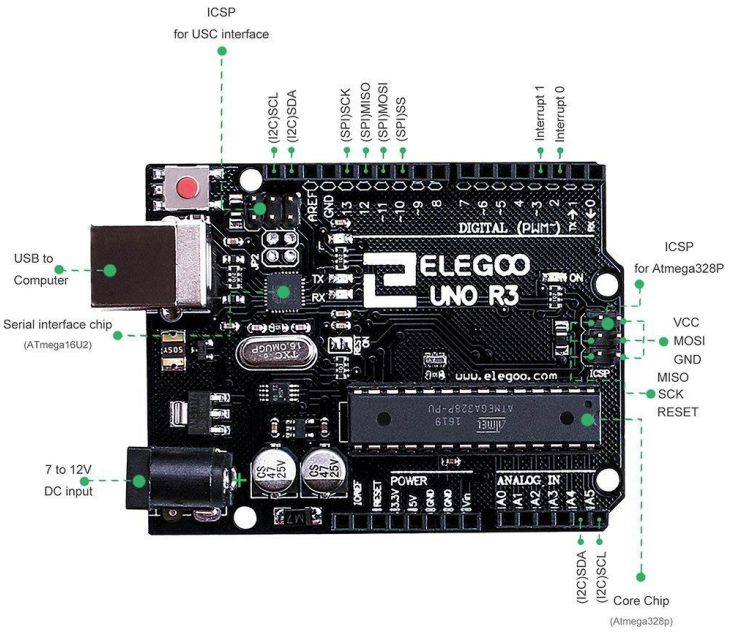


Figure 17: Arduino UNO

To have color shifting lighting, we needed to have a RGB LED and the one included in our Arduino starter kit fits our demo room (Figure 18). Through our research, all the four pins RGB LEDs have similar prices and brightness. So we decided to use the one in the Arduino kit and it works stable in our whole project. In our actual product, we will use a larger color shifting LED with improved aesthetic and brightness. The small RGB LED for demo room is about $0.2 in average and Color Shifting String Light is about $8.33 per bulb (Table 10). We chose the four pins RGB LED for our project demo because it's cheap and small.

Table 10: Light Specifications

Type	Price	Weight
RGB LED (demo room)	$0.20	1.6 ounces

Color Shifting String LIght (actual room)	$8.333	16 ounces.

Figure 18: RGB LED used in demo room

A RGB LED has four pins: Red, GND, Green and Blue. Every color pin can receive the signal from 0-255. If three pins are all 255, the color of the LED is white. If all the pins are 255, the RGB LED is off. The schematic is shown below in Figure 19.

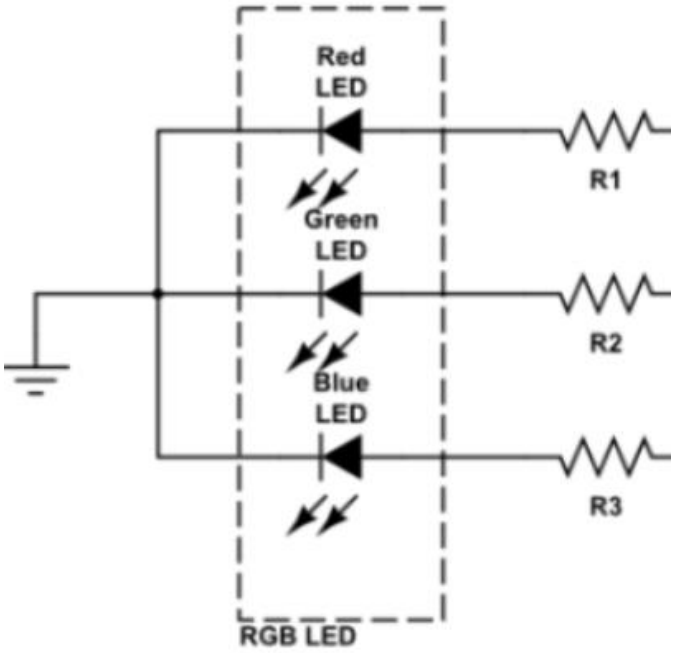


Figure 19: RGB LED schematic

We are looking into implementing a different kind of RGB LED for future iterations to improve brightness and area covered by the light.

In order to detect the number of people inside the room, we need a device to recognize people entering and leaving from the door. There are four devices can be used in our project: IR break-beam sensor, motion sensor, camera and temperature sensor (Table 11).

Table 11: Occupancy Concept Selection Matrix

	Motion sensor	IR break beam sensor	Camera	temperature sensor
Selection Criteria				
Difficulty	-1	1	-1	1
Cost	1	1	-1	0
Privacy	1	1	-1	1
Feasibility	0	1	1	-1
Sum of +1's	2	4	1	2
Sum of 0's	1	0	0	1
Sum of -1's	1	0	3	1
Net Score	1	4	-2	1
Rank				

Through our criteria for subsystem problems, we find that IR break-beam sensor is the most fit one for our emergency system. The major problem for a motion sensor is the difficulty; the knowledge base relates to the machine learning and we do not have enough time to work on it in the summer course. For camera, the lack of privacy is a problem. Data shows that 24% of people feels cameras for security would not be acceptable [10]. The temperature sensor was eliminated because of the feasibility: there are many factors that can affect the temperature of objects inside the room and cause false readings. The IR sensors will be implemented as shown below in Figure 21.

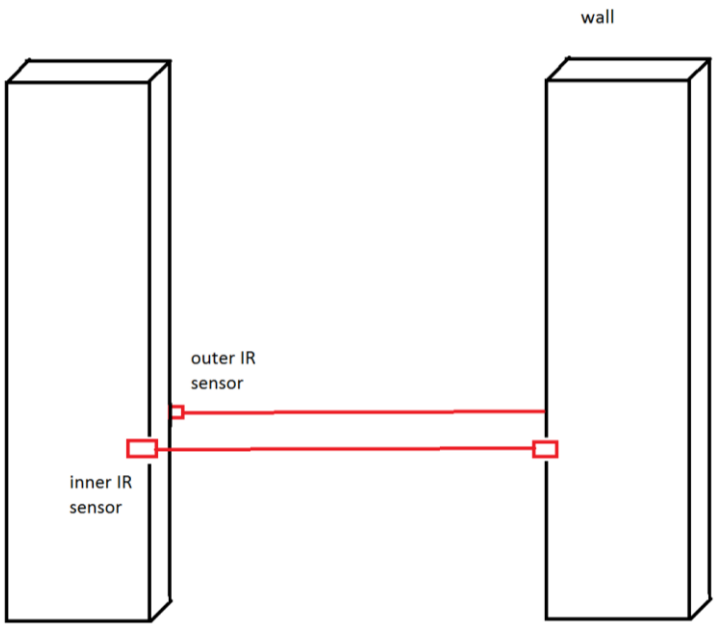


Figure 21: Beams break when people entering or leaving through the door

For our demo room, we find two IR break-beam sensors. One is IR 3mm with a maximum range of 25 cm and another one is IR 5mm with a maximum range of 50 cm. According to our door size, IR 5mm fits while the range IR 3mm is too small. The one we have selected is shown in Figure 22.

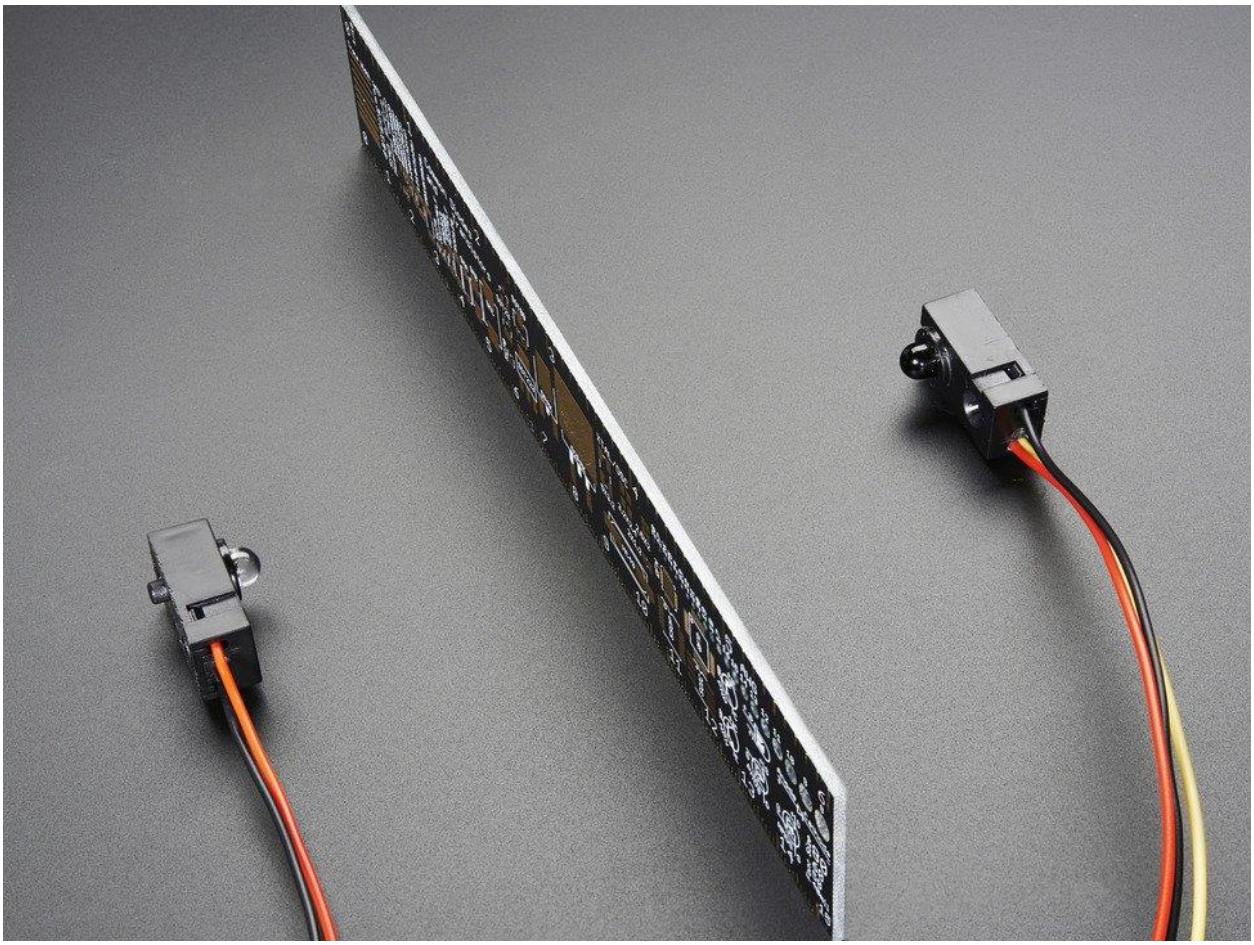


Figure 22: IR break-beam sensor we use in demo room

In a set of IR break-beam sensors, there is a receiver and a sender. The sender sends infrared beam to the receiver. When the receiver stops receiving the infrared beam, it means there is object breaking the beam. If the outer beam breaks first and then the inner beam breaks, it means there is someone going into the room. If the situation is reverse of this, it means there is someone leaving the room. Through this method, we can get the number of people inside the room. The IR break-beam sensor schematic is shown below in Figure 23.

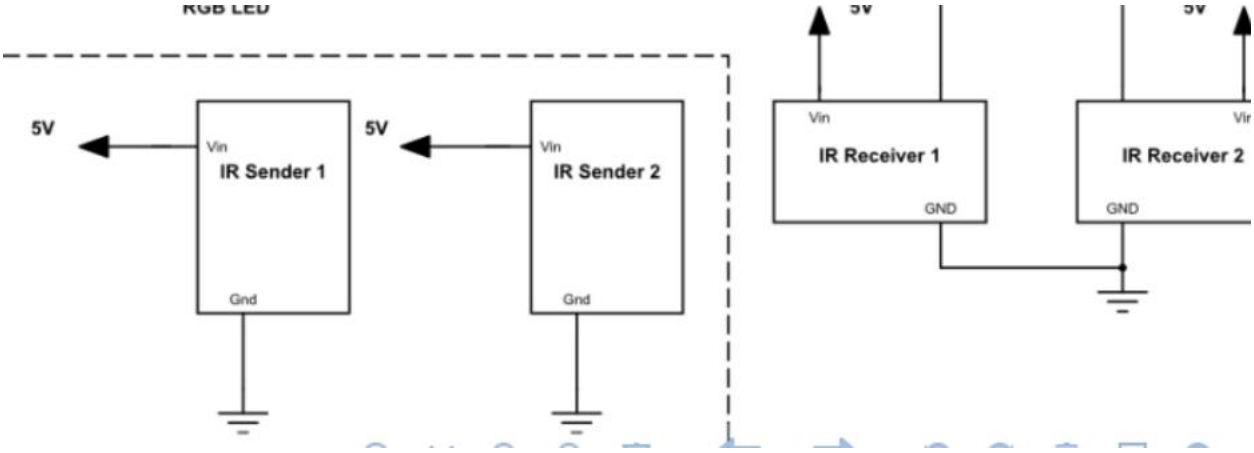


Figure 23: IR break-beam sensor schematic

After detecting the number of people, we need to find a device to recognize people falling down. Through the research on the market, we find motion sensor, range sensor, camera and robot suit our target. We finally choose range sensor based on our selection matrix shown below in Table 12.

Table 12: Fall detection concept selection matrix

	Fall detection				
	Kinect sensor	**Range sensors**	**Camera**	**Robot**	
Selection Criteria					
Reliable	1	1	0	-1	
Accuracy	0	-1	0	0	
Cost	0	1	-1	-1	
Privacy	-1	1	-1	-1	
Energy efficient	1	1	0	0	
Aesthetics	1	-1	0	0	
Easy to use	1	1	1	-1	
knowledge base	-1	1	-1	0	
Sum of +1's	4	6	1	0	
Sum of 0's	0	0	0	0	
Sum of -1's	2	2	3	4	
Net Score	2	4	-2	-3	

The motion sensor is hard in six-weeks course because the knowledge base is related to the machine learning. For the camera, the privacy and security is a concern for consumers [10]. If we use the robot to check for falls, safety, privacy, feasibility and

ease of use to the consumers are problems. Range sensor scored the highest and was therefore chosen as our fall detection method.

We have three different types of distance sensor. They are listed on the table below (Table 13). When we considered which sensor we should use in our demo, the detection range was the most important selection criteria. As shown in the table, neither VL5310 nor VCNL4010 is able to measure the distance of the floor to the ceiling of our demo room. So we can only choose VL53L0x. The range sensor that we have selected is shown in Figure 24.

Table 13: Range Sensor Selection Matrix

	Three types of Range Sensor				
	VL5310	**VCNL4010**	**VL53L0X**		**N/A**
Selection Criteria	**(0.5-10)**	**(10-20cm)**	(5-120cm)		
Detection Range	-1	-1	1		
Cost	0	1	0		
Size	0	0	0		
Easy to use	0	0	0		
Energy efficient	0	0	0		
Accuracy	0	0	0		
Easy to use	0	0	0		
Sum of +1's	0	1	1		
Sum of 0's	6	5	6		
Sum of -1's	1	1	0		
Net Score	-1	0	1	0	
Rank					
Continue?					
Rank					
Continue?					

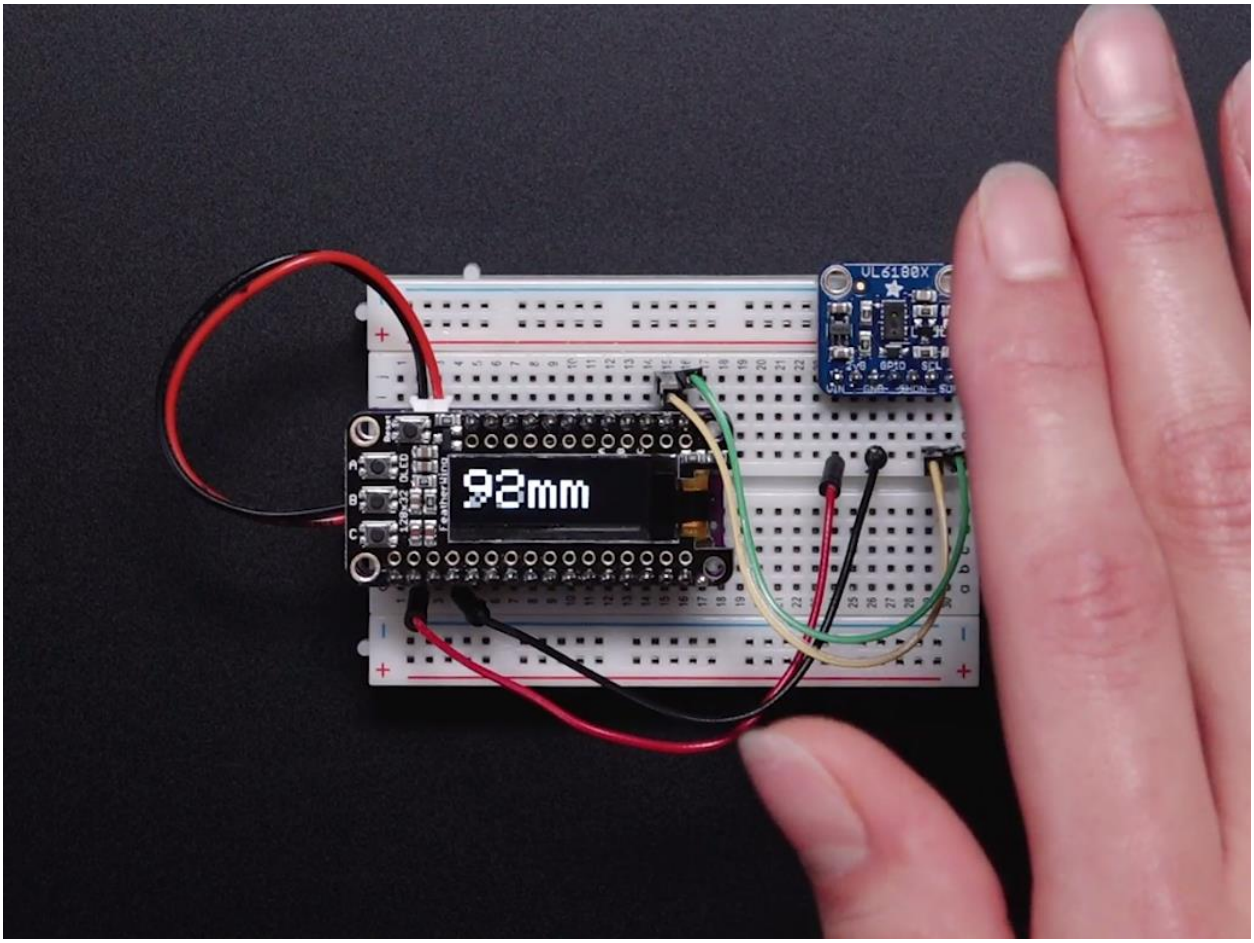


Figure 24: Range Sensor

Range sensor (Lux Distance sensor) reads the distance through time flight of light. There is a tiny laser source inside; the range sensor detects how long the laser light taken to bounce back. So if there is no one under the sensor, the distance measured is the distance between the sensor and the floor. If someone is right under the sensor, the reading number is the distance between the sensor and the top of the person. The difference in the number helps us detect people falling down. The schematic for the range sensor is shown in Figure 25.

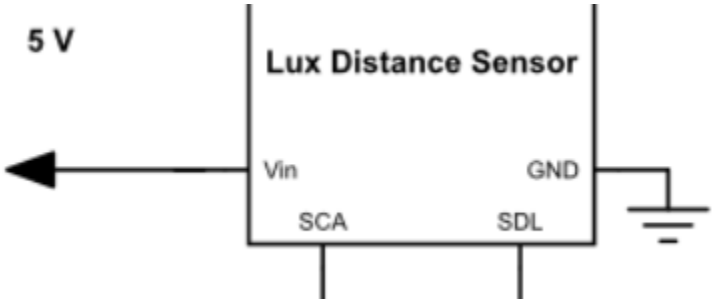


Figure 25: Schematic of Range Sensor

In connecting the range sensor to the circuits, we need to connect the SDA and SCL on range sensor to the SDA and SCL on the Arduino. SDA is the Clock Line and the SCL is the Data Line. SDA is used to send and receive the data and SCL is the clock cycle. For our project, range sensor has four wire connection.

In our plan, we use microphone as voice control but the knowledge base and the time limits forced us to find a solution. We use push button to show two situations in our proof of concept: “Yes, I’m OK.” and “NO, I’m not OK.”

Figure 26: Push button

In wiring the push button, we need to put in a 1K resistor. The resistor here works as a pull-up-resistor to ensure that the signal will be a valid logic level.

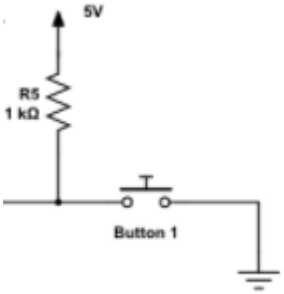


Figure 27: Schematic of push button

To ensure all the components work in the system, the power needs to be controlled. If the power is too high, the devices will be burned down. If the power is too low, the components can’t work. For Arduino, the maximum voltage is 9V. The voltage range for the Bluetooth Module, IR Beam Sensor and the Range Sensor are from 3V to 5V.

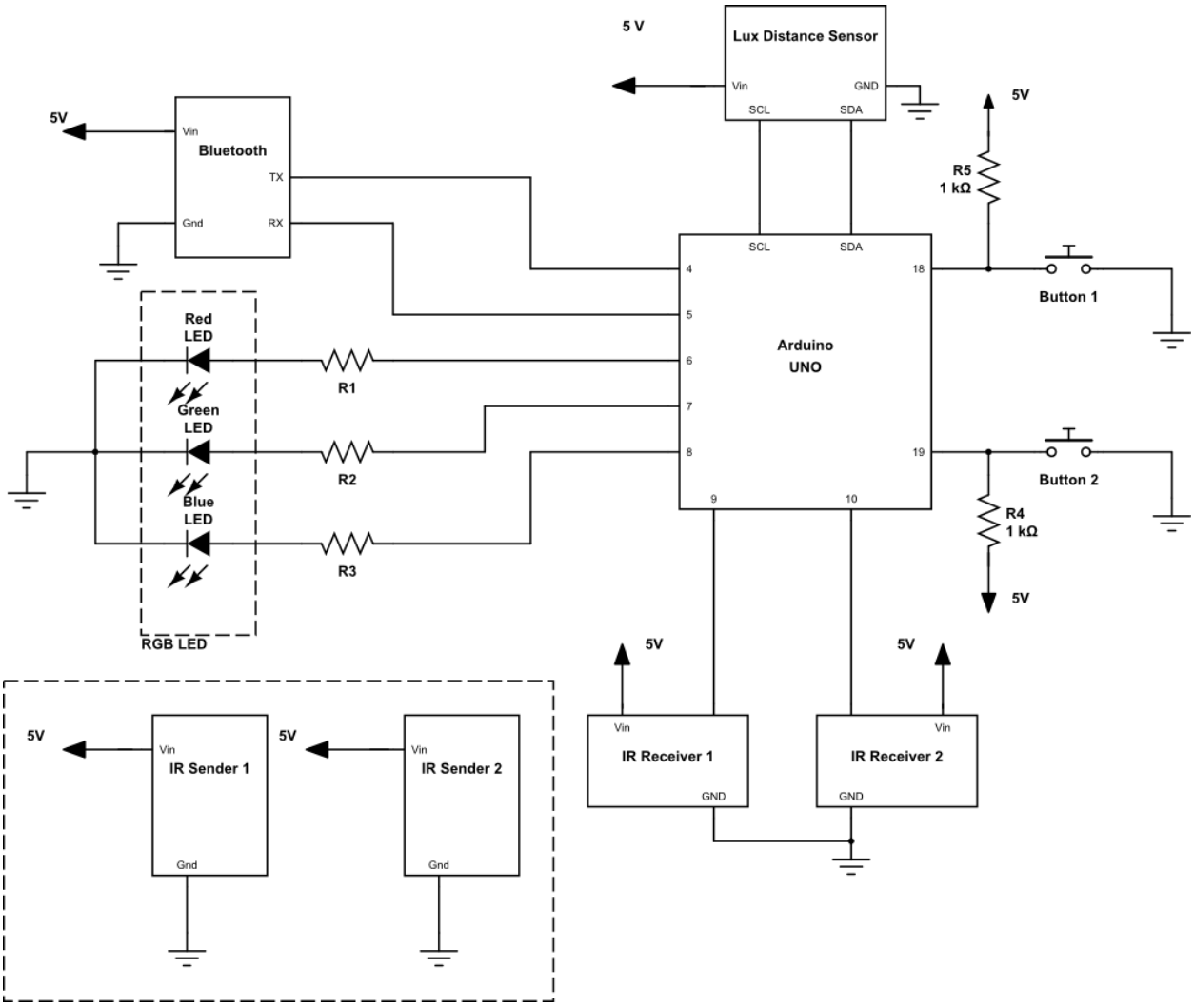


Figure 27: Arduino, RGB LED (Color shifting LED), IR break beam sensor (Longer Distance Sensor), Range Sensor, Push Button (Voice Control), Power

Subsystem 3: Fall Detection & Confirmation System

Arduino IDE 1.8.3 is used as the programming platform for all of our coding in the fall detection and confirmation system. The Arduino IDE includes two basic functions which are **setup**() function and loop() function

- **void setup()**: This function is used for writing code to initialize all the pin on the Arduino uno that connecting to the electrical components, so that I can gather data from sensors and give input signal to the pushbutton and RGB light.

In the beginning, we need to know if there is anyone in the room. Our system implements two IR break-beam sensors on the door used to detect if there is a person passing through the door. I designed a function called **LoEroom()** to get the data from

the IR break beam sensor and to determine if the person is entering or leaving the room.

- **Void LoEroom():** This function first reads the state of the 2 IR breaker beam sensors. If the reading of the IR break beam sensor is LOW, it means that the IR beam is broken.There are 2 conditions in this function, these 2 conditions are basically the same except the order that the 2 IR beam are broken. The function uses these 2 conditions to tell the difference between entering and leaving the demo room. Also the function will record the number of the people in the demo room.

When we know the room is not empty (someone is in the room), the RGB light will turn on automatically. I used a function called **lightup().** In order to turn off the light when all the people in the room leaves (number of people in the room becomes 0) ,a function called **lightdown()** is used. I used **lightup()** and **lightdown()** in all the circumstances that needs to turn on/off the light in my code.

- **void lightup()**: This function serves to turn on the RGB light.

- **void lightdown():** This function serves to turn off the RGB light.

We can also control the the color of the light from the smartphone application (app). For the Arduino UNO to read the data transferred from the app, I created a function called **readvalue().** Once the data is received, **setColor()** function will make the RGB light to change color according to the data from the mobile app. Combining the light switch function and **readvalue()** function, **SFLM()** function can be used to switch on/off the RGB light from mobile app.

- **int readvalue():** This function transfers the data from smartphone app to Arduino UNO, and returns integer values (C type).

- **void setColor()**: This function is used to control the color of RGB light by changing RGB color values.

- **void SFLM()**: The name of the function stands for switch on/off lights from mobile app. **SFLM()** function calls **readvalue()** function to read data sending from mobile phone, and switch on/off the light correspondingly.

When the number of people in the room is larger than 0, range sensor will start recording the distance between the area it’s detecting and the ceiling. If the reading goes between 210 and 229 centimeters, the code will enter a **while** loop. The system

will confirm that there is a person lying down under the sensor only when the program has 30 readings from 210 to 229 centimeters coming from range sensors. The usage of 30 times is to prevent some noisy data and misreading from sensors influencing the fall detection. As soon as we know there is a person lying on the ground, calling **mySerial.print()** is able to send notification signal to the smartphone app to make the sound '**Are you ok?**'.

- **mySerial.print()**:The function is used to send signal directly to the smartphone application.

After hearing "**Are you ok?"** customer can answer "YES", "NO" or not respond to the system. Since we only have one month for this project, we could not do the voice recognition in such a short time. We use a "YES" button and a "NO" button instead, there are 3 options available:

1. **Pressing YES button**: If someone just wants to lie on the ground, he/she can press the yes button, nothing will happen.
2. **Pressing NO button:** The code enters a **while** loop, calls the **blinklight()** function and uses **mySerial.print()** to send a specific alert signal to smartphone application to trigger the emergency mode in the phone app. The **while** loop will not break until there is no one lying down under the sensor.
3. **No action after 5 seconds**: When there is no response to the system in 5 seconds, what will happen is the same as **Pressing NO button**. I use **millis()** function to make sure the response time is exactly 5 seconds.

- **void blinklight():** This function is used to make the RGB light blink in red in a high frequency.
- **Int millis()**: Starts to record time once beginning running code, returns a time in milliseconds when calling the function.

Subsystem 4: Wireless Module

Wireless Module is used to enable communication between *Arduino UNO* and *smartphone application*. There are five different options for implementation: Wi-Fi, Bluetooth, Arduino chip as Wi-Fi hotspot, VPS (Virtual Private Server) and SMS messages (Figure 28). For our prototype, we will implement *low-energy Bluetooth* (i.e. *BLE*, *Bluetooth 4.0*, Figure 29). We chose this method considering time-limit, budget-limit and easiness of setup. There are three options of Bluetooth modules available on market: HC-05, HM-10 and HM-11. Because HC-n modules are not low-energy type and iOS run devices do not have protocol to receive data from such Bluetooth modules and cannot build connection bridges with them. HM-11 is unsoldered and only

compatible with 3.3V VCC, so it requires our further assembly and soldering, which may generate unnecessary and avoidable bugs in hardware. Considering limited time for our project and result of *concept selection*, we choose HM-10 to enable wireless communication. Here the table below is the concept selection matrix (Table 14):

Table 14: Selection Matrix for Bluetooth Modules

	Low-energy Bluetooth Modules				
	HC-05	**HM-10**	**HM-11**	**N/A**	**N/A**
Selection Criteria					
Difficulty	1	1	1		
Cost	1	0	-1		
Reliablity	0	1	1		
Feasibility	-1	1	1		
Further Assembly/Soldering	1	1	-1		
Power	-1	1	1		
Sum of +1's	3	4	3		
Sum of 0's	1	1	0		
Sum of -1's	1	0	2		
Net Score	2	4	1		
Rank	2	1	3		
Continue?	No	Yes	No		

Cutline for Table 14:

- Difficulty: 1 -- our knowledge base is enough to implement, 0 -- we need to require guidance and driver functions to implement, -1 -- we need to build kernel functions to support a raw hardware implementation
- Cost: based on price on Amazon and manufactures' websites (DSD, DSD, Qunqi)
- Reliability: 1 -- Very little noisy data generated and connection is stable, 0 -- some noisy data and connection is unstable in long distance, -1 -- very noisy data or unstable
- Feasibility: 1 -- work with iOS and Arduino UNO perfectly and have sample code and libraries as open source, 0 -- only work with iOS and Arduino UNO but no available code, -1 -- cannot be implemented for iOS or Arduino UNO
- Further Assembly: 1 -- No any more assembly needed, 0 -- Need voltage divider to adjust 5V to 3.3V, -1 -- Need soldering and other assembly steps
- Power: 1 -- BLE, -1 -- not BLE, No 0 (zero)

Because Bluetooth is a built-in function on the *Arduino* chip, we do not need to purchase more hardware. We are also assuming that the user's emergency contact(s) has a smartphone that has Bluetooth capabilities for our solution.

Figure 28: Five Options for Wireless Module

BLE
(Low-energy Bluetooth, Bluetooth 4.0)

Figure 29: Low-energy Bluetooth

For the ideal module (future iterations), we will use *VPS* (*virtual private server*, figure 28) to provide a communication service for customers. This requires the house that installs our system to have at least a Wi-Fi connection. As mentioned previously, in 2016, 73% of American adults had Internet access in their home. Additionally, 77% of American adults owned a smartphone as of 2016 [18]. In 2012, 61% of American households had a Wi-Fi connection [19] and in 2014 over 90% had three or more devices connecting to the Internet in their home [9]. These numbers are have increased in 2016 [18]. Based on these facts, we feel comfortable using this method. This allows our ideal module, VPS, to communicate with Arduino via user's Wi-Fi. Considering the

rest small amount of houses who do not have Wi-Fi implementation, we could contact telecom carriers in future iterations to enable SMS text message communication and cellular network.

Figure 28: VPS (Virtual Private Servers)

The way to implement VPS is to rent a server from a large server-service provider, such as Amazon and GoDaddy. There is no establishment fee for a VPS and the cost would be approximately $32 per month to have 240GB storage, 8GB memory and unmetered bandwidth [10]. If our product is sold in large amount, we can build our own servers to provide more private and secured web services to users. The initial investment to servers is high but, because it is fixed cost, the cost can be neglected in long run. Hence, here is our ideal plan for VPS and server use:

- In the short run, the company we rent VPS from will keep data and perform maintenance.
- In long run, we will establish a hardware maintenance team and build our own server, which can ensure safety and privacy more because we have self-owned and local server.

Subsystem 5: Smartphone Application

Our project uses the Apple iOS platform to build an application, “SightRPI”, where customers can get a fast and effective notification of the one at home experiencing an emergency. The overall concept is when a person falls down at the place implemented with our product, the application (Emergency Contact mode only) can get a pop-up notification in less than 2 seconds.

SightRPI requires operator select between User and Emergency Contact. When the operator chooses User, a control system will be displayed to control different light preferences. If operator selects the Emergency Contact, a home phone number is needed to contact the person (user) who is using our product and application will only

show the status of the current room (emergency or safe). This is demonstrated in Figure 29 and Figure 30.

Home page

The homepage is simply a login page. Operator will need to choose a identity to login to a certain page and provide us the phone number that will be called when meet the emergency case. As we are using low-energy Bluetooth for our prototype, we need connection to HM-10 to be full functioning, so the top left button is designed for connect. When connected, this button will turn to red and text will change to “Disconnect” (Figure 30).

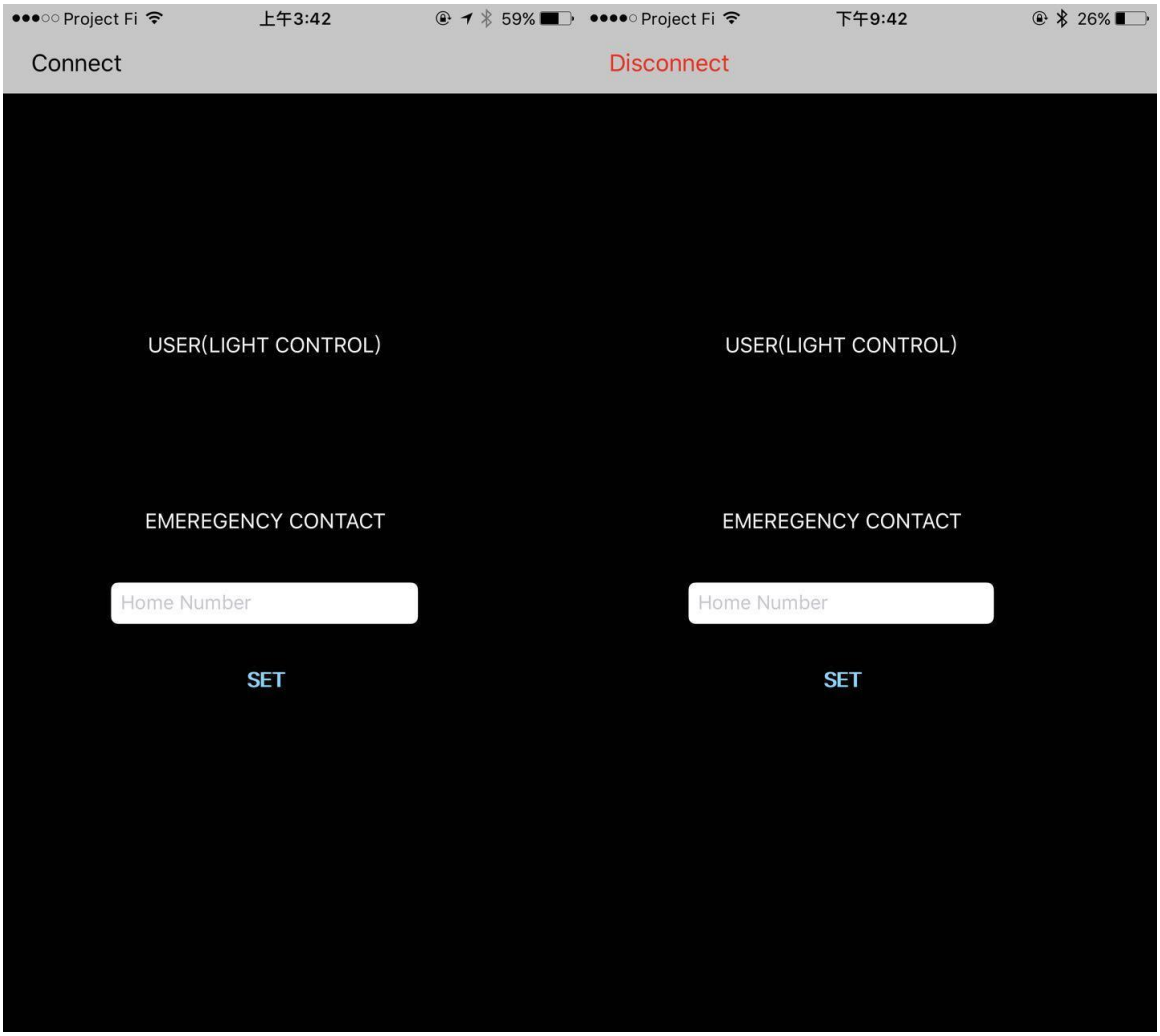


Figure 29: Home page 1

Figure 30: Home page 2

User (light control) page

User can change the color of the light when they adjust the RGB (Red, Green, Blue) sliders and press the “change” button. They can turn on or off the light by using the

“LIGHT” switch and press “SEND” button. The “LIGHT” switch also indicate the light status (on or off).

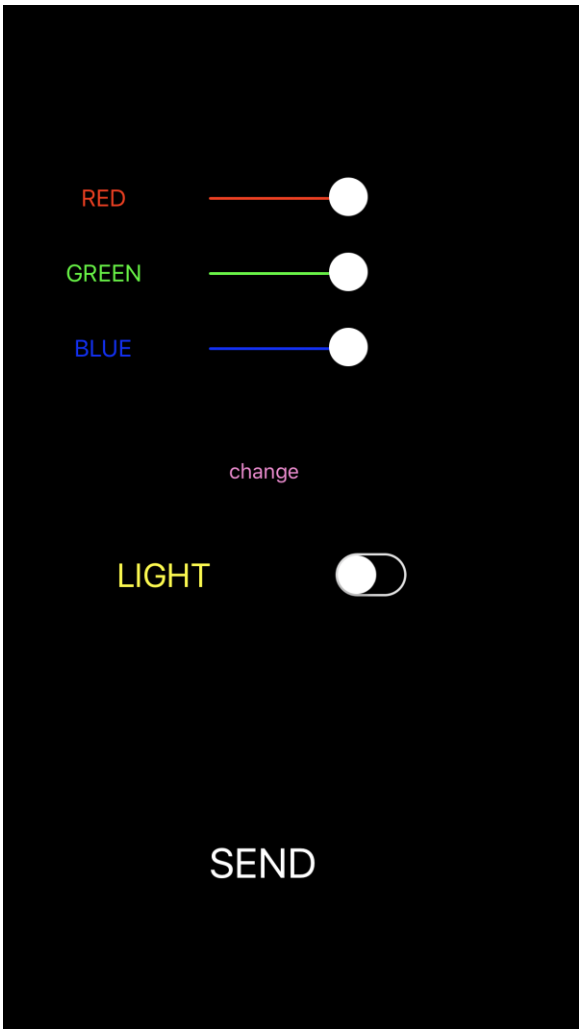


Figure 31: User (light control) mode

When user falls down, it will ask “Are you OK?”. If user answers “no” or does not answer within 5 seconds, the application will play alarm sound.

Emergency Contact page

In this page, user can get the status of the person at the place where our product is implemented. The eye means it is safe and system is guarding the person. Emergency sign shows that an emergency situation has happened and 3 switches will show to let user decide what to do next.

- Clear switch: is only used for test. When the clear switch is triggered, it means the emergency situation has been take care of. Once the situation is cleared, the application will go back to eye page and hide all 3 switches
- Ask switch: triggering this switch will allow user to call the number that is recorded at the home page. It is used as double conformation to make sure if the person is okay or not.
- 911 switch: It will let the operator call 911 in a one button press.

Figure 32: Eye page

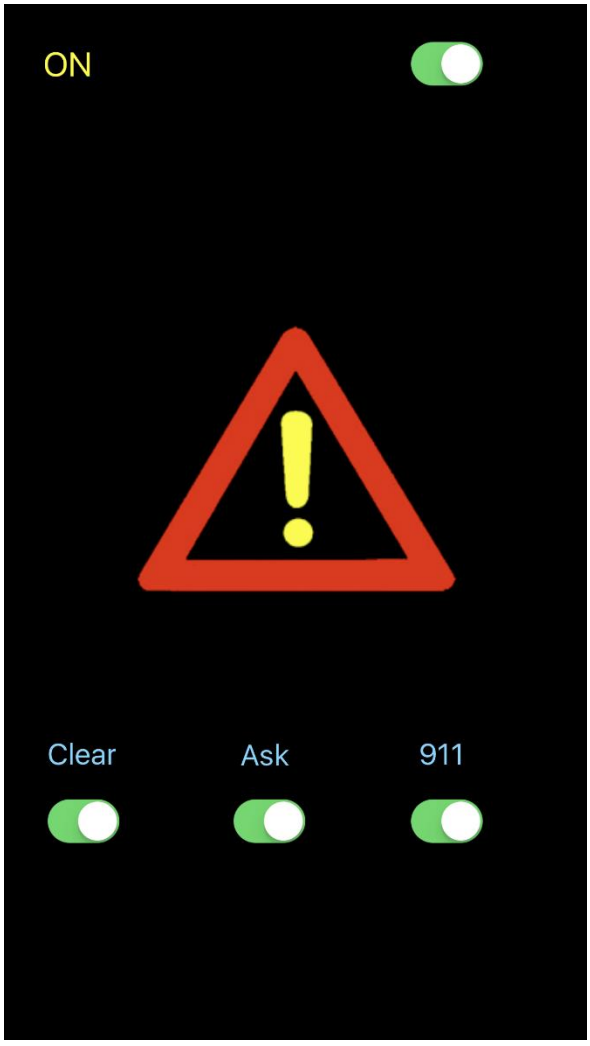


Figure 33: Emergency page

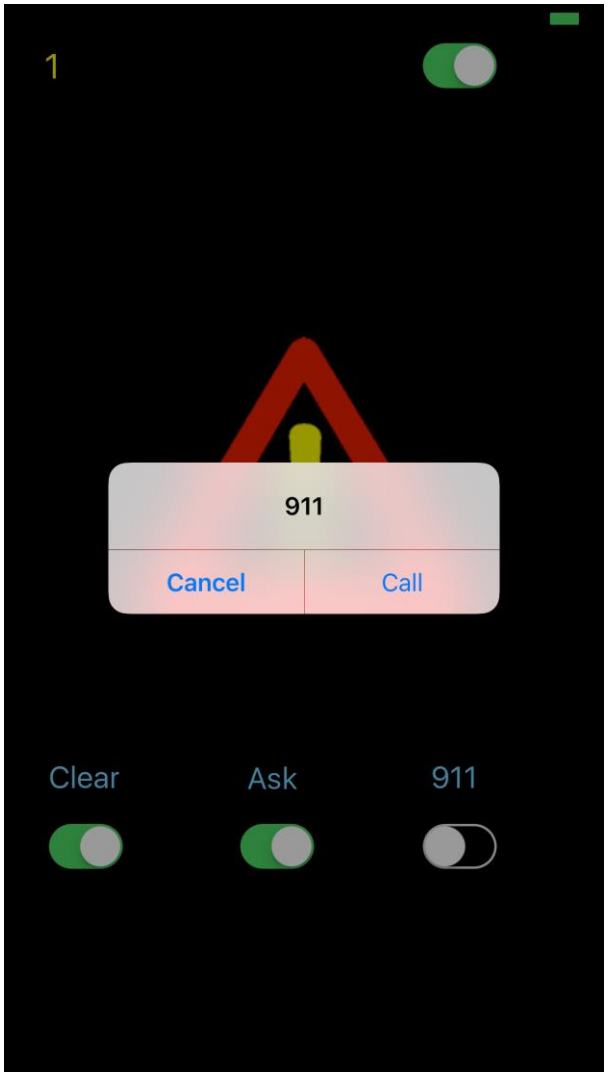


Figure 34: when ask or 911 pressed

Push notifications

Push notifications will show within 2 seconds of the emergency situation if the operator is using the application as the Emergency Contact. This is shown below in figure 35.

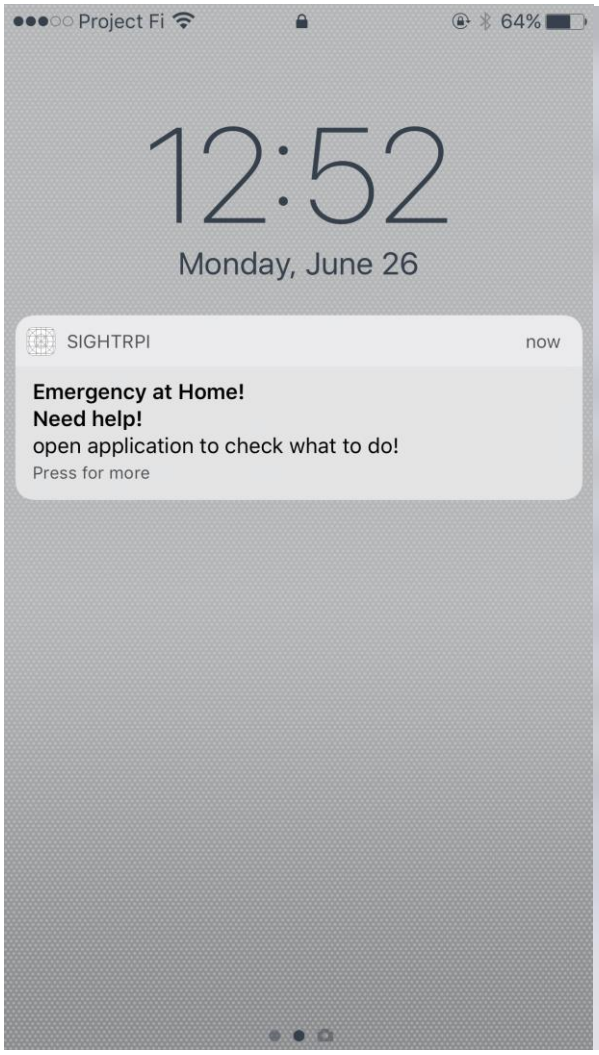


Figure 35: Notification at Lock Screen

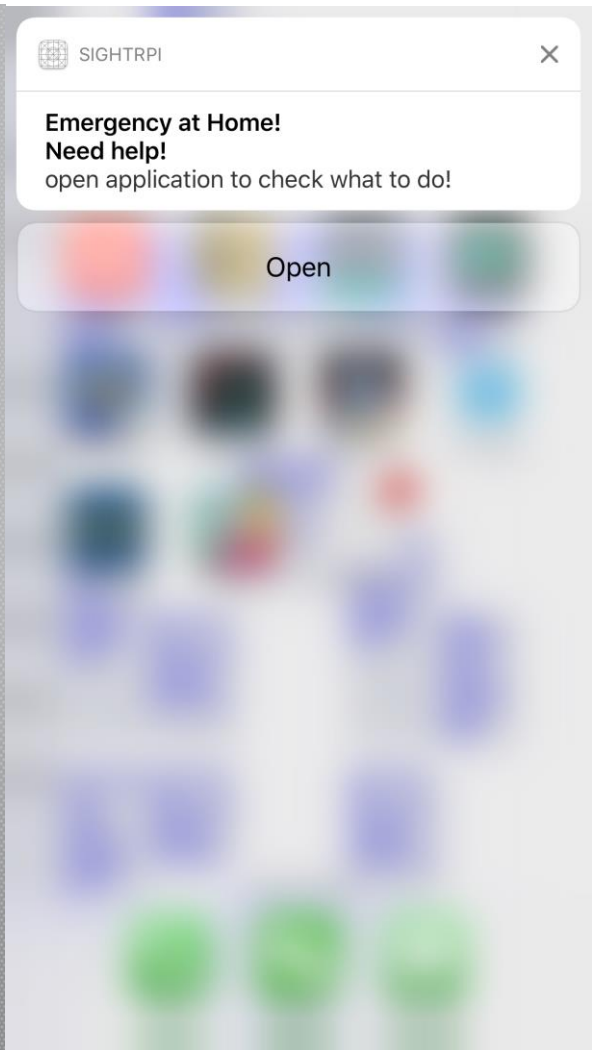


Figure 36: Notification at Main Screen

Results and Discussion

Results

Each subsystem met the technical specifications that were set forth prior to demonstration day. We performed many tests ahead of demo day in order to ensure that all of our hardware was functional. Additionally, this allowed us to debug all of our code.

One of the problems we encountered was reading a false positive when Mike was standing underneath the range sensor. We solved this by modifying our code to count the number of readings that are within the range of laying on the ground. If the number of readings is greater than the amount we specified, then a fall will be detected. One of the other problems that appeared was that the human figure must be within a very specific area to be detected by our system. This was solved by practicing moving human figures within the room and making sure that our hands were not disrupting the sensor.

The demo room met all parts of the technical specifications since all measurements (door, ceiling height, model heights) were within the ranges specified. The circuit design subsystem met all parts of the technical specifications since the light displayed all of the correct colors when prompted, the serial monitor printed all of the correct outputs that we specified, and the range sensor read the correct value when no one was in the room. The wireless module subsystem also met all of the technical specifications since the data was printed within <100 ms, and there was zero noisy data within 10 seconds during running. The confirmation subsystem met all of the technical specifications since our system read the correct room occupancy in <1 second, detected a fall within 5 seconds of the fall, and the light changed in <1 second. Finally, the smartphone application subsystem met all of the technical specifications. While in user mode, the alarm sound was played in <1 second after the fall and data was sent in <1 second when the button was pressed. While in emergency mode, the image was changed to emergency state in <1 second and the three options were displayed on the phone screen, the correct phone number was called, and a push notification appeared within <2 seconds of the emergency being detected.

For future iterations of our system, we are absolutely considering adding more sensors to cover a larger area of one room. One of the ways we are thinking about implementing this is angling the sensors in order to cover more area per sensor. If we had been granted more time for this project, this is something that we could have potentially implemented. We also considered using a different wireless module, but for the sake of

this proof of concept, we selected Bluetooth due to time constraints. For future iterations, we will be considering a VPS since it is more secure and has a larger range.

Significant Technical Accomplishments

One of the most significant technical accomplishments for our team was implementing an Arduino and successfully using it. While nearly all of us have had some coding experience previously, none of us had experience with writing a program for an Arduino. The Arduino is one of the most important parts of our system and many of our system's other components rely on it. This made debugging difficult initially but after many trials and a lot of research, Ding was able to design a program that is functional and allows for our system to work as a whole.

Another accomplishment of this project is our smartphone application. Ziyi had some experience previously with iOS but was not proficient. By the end of this first development cycle, Ziyi was able to create a working iOS application that we were able to install on an iPhone. The application can open and can communicate with our system as well as other applications on the phone (such as the Phone application for calling emergency contacts or 911).

These two components of our system are two of the most important ones to the entire system's function, but they both have met the technical specifications and work well within our system.

Conclusions

The goal of our project is to design a fall detection system to automatically detect falling of users and address such emergency cases. Our system uses a range sensor to detect if a person falls down and an IR break beam sensor to enable auto-switch of light aiming at saving energy. We implement Bluetooth to enable communication between Arduino UNO and smartphone. For demonstration of our proof of concept, we also built a demo room and implemented some model human figures to simulate the real life.

During researching on customer requirements, because we do not have large survey target group, we obtained data from websites of existing products and many organizations. For benchmarking, we tried to do research on every similar existing product and summarize their features. To design and build a competitive product, we looked into development of technology and what we can implement into our own product, which is not available when other existing products were introduced to market.

The whole team works together seven days a week cohesively. After we all finished our own subsystems, we combined them together and did further design and debugged together because we share common values towards our project. When encountering conflict, each of us is willing to listen to others' suggestions and learn from others. Therefore, we can reach all our goals promised in planning phase after six weeks' hard work.

Considering the testing results, because we have combined our own subsystems together and done further debug, our system passes every test cases and even all of the marginal cases. With the goal not only to finish IED class but also to make a fantastic project, we achieved a number of accomplishments and learned much new knowledge that we have not touched before.

As time passed, we followed Gantt Chart in planning phase and devoted all our efforts to finish every task in time. Some tasks could not be finished before deadline once set, but we then worked together to chase the progress.

Here are the conclusions and further improvements (if applicable) for each subsystem,

- For demo room,
    - Foamboard is the best material for creating a testing environment since it will not disrupt our sensors and it is easy to manipulate.
- For circuits part,

    - Range sensor we use (VL53L0X) can measure the whole demo room, but the reading values are not very accurate and stable. For further improvements, we will use ultrasonic sensor to gather more accurate data.
    - IR Break Beam Sensor (5mm) works as desired and is very sensitive
    - RGB LED will be replaced by high-power and more aesthetic lights and will be separated into normal light and alarm light. Alarm light will be equipped with a speaker and a battery to ensure emergency mode can be turned on at any time necessary.
    - Bluetooth has up to 60 meters radius covering area. According to tests, there is very little noisy data, so as a way to transmit data, Bluetooth is a reasonable choice.
    - Arduino UNO, as a core processor, is very easy to use and inexpensive. The only defect is that it only has 13 pins, so in further improvement, we may need to switch to Arduino Mega or Due to implement speaker and microphone.
- For smartphone application,
    - Our goal is to create a working application that is user-friendly (easy to use) and can wirelessly communicate with our hardware device.
- For software part,
    - We need to design the code to process the data from the sensors and the logic that can translate the data into action.

Acknowledgements

It is a pleasant task of acknowledging the help we have received in developing this project work.

We are thankful to **Professor Kenneth A. Connor** for guiding us throughout this project which has really helped us to develop our work perseverance.

We also would like to thank **Professor Casey T. Jakubowski** for giving us his valued advice to develop our work discipline and public speech techniques.

We heartily thankful to **Professor Hisham Mohamed** for being a constant source of inspiration.

Thank you all for everything.

SightRPI Group Members

References

[1] ADT. “ADT Medical Alert Systems”. *ADT.* ADT LLC. Web. 28 June 2017.

[2] Alford, Debbie. "Aging in Place Facts & Statistics." *Aging in Place Facts--Data.* In Your Home, n.d. Web. 08 June 2017.

[3] Byrne, Sue. "GreatCall and Other Mobile Medical Alerts Promise Help When You're Out or at Home." Consumer Reports. Consumer Reports, 17 Sept. 2015. Web. 08 June 2017.

[4] Center for Disease Control. “Body Measurements.” *National Center for Health Statistics.* U.S. Department of Health & Human Services. Web. 09 June 2017.

[5] Consumer Reports. "What to Look for in a Medical Alert System." Medical Alert Systems Comparison - Consumer Reports. Consumer Reports, July 2015. Web. 08 June 2017.

[6] (COR), International Code Council. "Section 1208 Interior Space Dimensions." International Building Code, 2012. Country Club Hills, IL: International Code Council, 2011. N. pag. Print.

[7] Emrath, Paul. “Spaces in New Homes”. *National Association of Home Builders,* October 2013. Web. 08 June 2017. (NAHB)

[8] Greatcall. Greatcall Lively Wearable device. Digital image. Greatcall. Greatcall, n.d. Web. 28 June 2017.

[9] GreatCall. "Wearable Medical Alert Device." *GreatCall*. N.p., n.d. Web. 28 June 2017.

[10] Ha, Thu-Huong. “A majority of Americans would be okay with facial recognition security at work.” *Quartz Media.* Quartz Media LLC, 14 January 2017. Web. 28 June 2017.

[11] MetLife Mature Market Institute. Falls and Falls Prevention. New York: MetLife Mature Market Institute, 2013. Print.

[12] Mombrea, Matthew. “How to build a rack-mounted server from scratch.” *IT World.* IDG Communications, 10 May 2013. Web. 28 June 2017.

[13] OMRON. “Proximity Sensors”. *OMRON - Industrial Advantage.* OMRON Corporation. Web. 08 June 2017.

[14] Orsini, Lauren. “Arduino vs. Raspberry Pi: Which Is Right For You?” *Readwrite.* Readwrite, 7 May 2014. Web. 28 June 2017.

[15] Philips. "HomeSafe-AutoAlert." Philips. Koninklijke Philips, 2017. Web. 08 June 2017.

[16] Philips. LifeLine with AutoAlert System. Digital image. Consumer Affairs. Consumers Unified, LLC, 08 June 2017. Web. 8 June 2017.

[17] Stein, Loren. "Medical Alert Systems and Personal Emergency Responders." AARP. AARP, 11 Nov. 2010. Web. 08 June 2017.

[18] Toys R Us. Happy Together Family. Digital image. Happy Together. Toys R Us, n.d. Web. 20 June 2017.

Appendix A: Selection of Team Project

The selection of team projects was done by brainstorming as a class and subsequently breaking up into respective groups based on interest in an idea. Our group was interested in this team project rather than the other because all of us were interested in the topic and realize that it is a big issue in today's society.

Fall detection of elderly has been under spotlight of society considering the safety of those who spend aging time in their houses alone. According to AARP [2], 90% of Americans wish to stay in their homes as they grow older. The MetLife Mature Market Institute (MMI) has found that one in three older adults will experience a fall this year, and nearly half of older adults cannot get up on their own [11]. CDC also released the data that 2.8 million older adults are treated for fall-related injuries in emergency rooms each year [4]. On top of all of that, most falls occur in the home [3]. Therefore, the problem identified is serious and ongoing.

Considering the market for fall detection system, according to AARP, 70% of elderly made modifications to their home due to safety concerns. Moreover, 60% of elderly made those modifications to their home to increase the chances that they will be able to live there independently [2]. Obviously, there is a large market for this kind of system and, because use of smart lighting is still not applied widely, our team project is competitive compared with existing products.

Appendix B: Customer Requirements and Technical Specs

Our system has one primary customer: elderly living in their homes. Some other groups that may be interested in our system are:

- Adult children/caregivers
- Nursing homes
- Hospitals

Stakeholders in this project include SightRPI team members, IED instructors, and competing medical alert companies.

As mentioned in Appendix A, research has shown that our ideal customer is an elderly person who is living in their home.

Detailed customer needs and technical specifications are detailed below in Table 15.

Table 15

Customer Needs	**Technical Specification**	
	Metric	**Target Value / Range of Values**
I need the light to be soft and doesn't harm my eyes.	Brightness	100-200 lux
I want to know as soon as possible if the one at home is in danger.	(Sensitivity of Fall Detection system) Time between reading all sensors	<100 milliseconds
If I am in my office, I may still want to control.	Wireless Module covering distance	0-∞ meter
Can it cover all area in my home?	Effective area	8 cm^2/sensor (in demo room)
Can it detect a child?	Age	3-10 years old

	height	>2.92 inches (on 1:12 scale)
I don't want my dog to trigger annoying alarm by mistake.	height	>2 inches (on 1:12 scale)
Important. I must receive notification at once.	Communication Latency	<2 seconds
I sometimes lie on the ground to think about things.	(Allow Deliberate Behavior when User is Safe) Confirmation System	# of speaker(s): 1 # of microphone(s): 1
Do not want to call 911 on accident.	Option to call 9-1-1	1 button (to call 9-1-1 required on smartphone application)
	Electronic Voice recorded in smartphone application	1 piece of pre-recorded electronic voice stored in system
I want to use my phone to control whole system when I am in my room.	Remote control	RGB range: 0-255 On/Off button
I don't want to see a huge thing sticking on wall or ceiling.	Appearance	Sensor size: 1.5-2.0
I like blue light when I am falling asleep. And soft yellow when I wake up.	Color change	(R,G,B) range: (0,0,0)-(255,255,255)
If I purchase such a system, I don't need a physical switch anymore.	# of wearable device(s)	0 (zero)

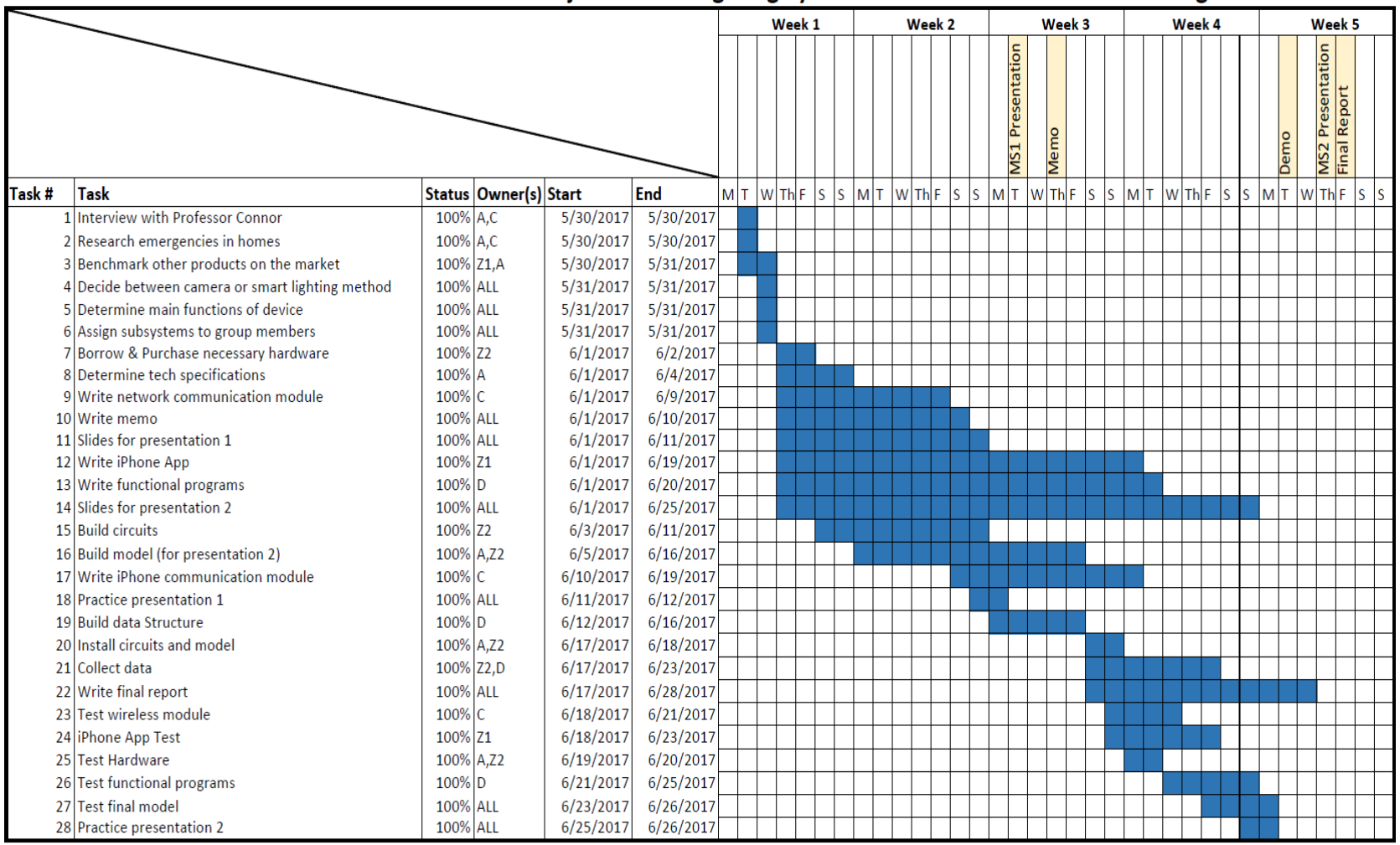
Low amount of false positives	Confirmation system	1 speaker and 1 microphone (to detect the reply from the user)
I don’t want to switch battery when battery runs out.	Connected to house’s power directly	Input: 100-240 v (varies in different countries and areas)

Appendix C: Gantt Chart

The Gantt Chart (Figure 37) shown below outlines every task and person responsible for the task. The status in Gantt Chart is the percentage of finish up to June 29th, 2017. Cutline for name abbreviations is under the table. The starting week is the second week of summer session due to group members' determination. The Gantt Chart is classified by topics, such as hardware, software and general concept decision. Important dates have been marked into yellow under the week row. Note that this is the chart for task distribution rather than subsystem distribution.

Figure 37: Gantt Chart

IED - ENGR 2050 **Project - Smart Lighting System** **Team - SightRPI**



Task #	Task	Status	Owner(s)	Start	End
1	Interview with Professor Connor	100%	A,C	5/30/2017	5/30/2017
2	Research emergencies in homes	100%	A,C	5/30/2017	5/30/2017
3	Benchmark other products on the market	100%	Z1,A	5/30/2017	5/31/2017
4	Decide between camera or smart lighting method	100%	ALL	5/31/2017	5/31/2017
5	Determine main functions of device	100%	ALL	5/31/2017	5/31/2017
6	Assign subsystems to group members	100%	ALL	5/31/2017	5/31/2017
7	Borrow & Purchase necessary hardware	100%	Z2	6/1/2017	6/2/2017
8	Determine tech specifications	100%	A	6/1/2017	6/4/2017
9	Write network communication module	100%	C	6/1/2017	6/9/2017
10	Write memo	100%	ALL	6/1/2017	6/10/2017
11	Slides for presentation 1	100%	ALL	6/1/2017	6/11/2017
12	Write iPhone App	100%	Z1	6/1/2017	6/19/2017
13	Write functional programs	100%	D	6/1/2017	6/20/2017
14	Slides for presentation 2	100%	ALL	6/1/2017	6/25/2017
15	Build circuits	100%	Z2	6/3/2017	6/11/2017
16	Build model (for presentation 2)	100%	A,Z2	6/5/2017	6/16/2017
17	Write iPhone communication module	100%	C	6/10/2017	6/19/2017
18	Practice presentation 1	100%	ALL	6/11/2017	6/12/2017
19	Build data Structure	100%	D	6/12/2017	6/16/2017
20	Install circuits and model	100%	A,Z2	6/17/2017	6/18/2017
21	Collect data	100%	Z2,D	6/17/2017	6/23/2017
22	Write final report	100%	ALL	6/17/2017	6/28/2017
23	Test wireless module	100%	C	6/18/2017	6/21/2017
24	iPhone App Test	100%	Z1	6/18/2017	6/23/2017
25	Test Hardware	100%	A,Z2	6/19/2017	6/20/2017
26	Test functional programs	100%	D	6/21/2017	6/25/2017
27	Test final model	100%	ALL	6/23/2017	6/26/2017
28	Practice presentation 2	100%	ALL	6/25/2017	6/26/2017

Alex	A
Cornelius	C
Ding	D
Ziyi	Z1
Zhen	Z2
Everyone	ALL

Appendix D: Expense Report

Table 16: Expenses

Item	**Quantity**	**Unit Price**	**Subtotal**	**Shipping**
Arduino kit	1	$34.99	$34.99	$0.00
Lux distance sensor	1	$13.49	$13.49	$0.00
IR break beam sensor	2	$3.90	$7.80	$15.00
Foam board	2	$6.99	$13.98	$0.00
Miscellaneous hardware	-	various	$6.38	$0.00
Bluetooth Module HM-10	1	$11.99	$11.99	$0.00
Figure set(Human)	1	$11.99	$11.99	$0.00
Figure(Dog)	1	$2.00	$2.00	$0.00
Total			$117.62	

To improve our project, we could have spent additional funding on more lux distance sensors in order to increase the amount of area covered. Additionally, we could have spent additional funding on a speaker for our system to demonstrate the confirmation method.

Appendix E: Team Members and their Contributions

Team Member 1: Alexandra Fearn

My subsystem was all about creating a room that we could use to demo our proof of concept. This included choosing a scale that would best fit our need of a feasible testing environment. I researched the average height of humans and how large the average American common room was in order to choose the dimensions of the room. Additionally, I also constructed our room based on standards set forth by the International Building Code. I was also responsible for collaborating with Zhen to place all of our breadboards and hardware. I also edited our presentations, first memo, and this memo to ensure that our writing sounded as unified and as clear as possible.

Team Member 2: Hanyuan (Cornelius) Xiao

For my subsystem, I researched possible options for *Wireless Module* subsystem and have done concept selection for the subsystem (read *concept selection matrix* for more details). Since I have chosen BLE for our prototype (reasons have been stated in previous paragraphs), I select HM-10 as the Bluetooth Module. HM-10 works perfectly with iPhone for our demo, because there have been available libraries and sample code online for open source (e.g. EVO Things, BLECentral, Hangar42, GitHub, etc.). Besides my subsystem, I also have helped build and debug hardwares and softwares. As the one responsible for communication with other professors, I maintain the communication between our team and Professor Connor via face-to-face meetings and emails.

Team Member 3: Junjie Ding

In the beginning, I help my teammates to test the electric component that used in our project. In my subsystem, I designed the fundamental coding logic for the fall detection and occupancy detection. I did all the codes on the Arduino IDE, which is the best coding platform for writing code that uploaded into Arduino UNO. For the occupancy detection, I created a complicated function to tell whether if a person is entering or leaving the demo room, so that the number of people in the room can be recorded. For the fall detection, I designed a function that can measure the distance between the person and ceiling, and tell the person if is in emergency based on my measurement of distance and the confirmation from the person from himself/herself. In addition to these two functions, auto light switch and remote color change are also included in my code.

Team Member 4: Ziyi Lu

My subsystem in this project was smartphone application. I started with research for the best smartphone operating system that we should launch. I looked into the available operating systems and found the best OS will be Andriod-based on the popularity. However, with my limited knowledge base we chose to start with the second most popular system: Apple iOS system. Then, I created the GUI (graphical user interface), which allows the user easily understand what our application is capable of. After this, I worked closely with the other two group members: Junjie Ding and Hanyuan (Cornelius) Xiao to let the application successfully communicate with our hardware device. I designed the logic that triggers the change (view, sound, notification) of application and preventing any false information sent or received through user's smartphone.

Team Member 5: Zhen Chen

In our Fall Detection System, my subsystem is to design the circuits and test cases. This means I need to cooperate with all the group member closely. When deciding the electrical devices, I need to discuss with Ding, Ziyi and Cornelius whether the electrical devices matches our knowledge base. In the final Demo Room building, I have to work with Alex to find the proper place to put the breadboards, wires and hardwares. In the beginning, I need to find the most suitable devices in every small systems of our project. Additionally, I need to ensure the power support, decide the Arduino Port use and avoid the hardware conflict. Since there are always bugs as the project processes, I need to figure out the hardware bugs. After the Demo room finished, I tested all the cases we thought of and check whether they passed the requirements.

Appendix F: Statement of Work

The only project for this semester of IED was to work through the entire product development cycle by creating a proof of concept within a team of students. Groups were determined based on interest in different ideas brainstormed in class. Initially, our group wanted to create a smart lighting system for the home. However, we agreed that this was too broad and not something that would yield a subsystem for each team member. We arrived at an emergency detection system after brainstorming further. The skills demanded by this project were the following: CAD/Adobe Creative Cloud software, circuit design and testing, materials properties, iOS programming, Arduino programming, Bluetooth module programming and usage, and model building. A Gantt chart was created in order to keep track of what tasks needed to be done and in what order. Additionally, the group unanimously agreed at the beginning of the project that all items for major deadlines would be completed at least 2 days before.

Appendix G: Professional Development - Lessons Learned

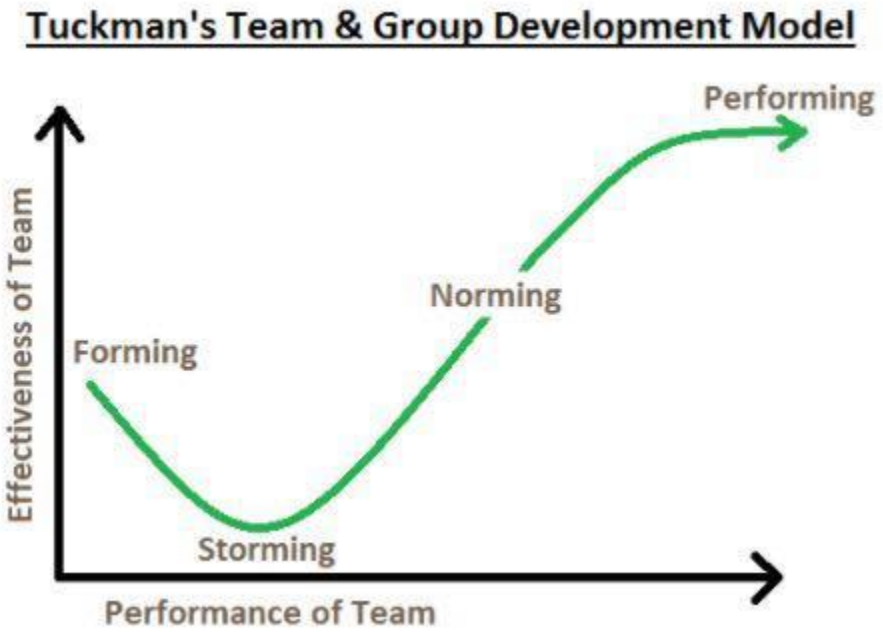


Figure 38: Tuckman's Stages

There are five team members in team SightRPI with different cultural and educational backgrounds. When we first came to this course to form a team, we shared different values and had conflicts. However, we were all willing to listen and learn from others to not only aim at finishing IED course and project, but also improve ourselves in knowledge and social network. This helped us resolve some of those conflicts.

When it comes to the planning phase of project design, we learned to design a Gantt Chart to manage time and designed subsystem distribution to manage work. When there is a deadline set for a specific task, our efficiency and productivity can be driven to maximum to ensure the whole project is in correct pace. Because we share a common value and have the goal to design a proof of concept, we seldom encountered conflict and always collaborated with other team members.

Because the time in summer session is very short, we can barely do survey on customer to gather customer requirements. Hence, we learned to do research and obtain data from websites of existing products and some organizations related. Sometimes, the data obtained was more accurate and more convincing, so that our prototype shown in demonstration can be more consistent with customer requirements. RPI has smart lighting center (LESA) and we obtained data from Professor Connor

there. Suggestions from Professor Connor are very helpful and thorough; some of which even helped us identify some potential problems during project design and development. Our team shows great respect to all professors who have offered instructions and support to our project. The special thanks and acknowledgements are included in Acknowledgements section.

In a few of our meetings, we always interrupted each other when someone else is talking. We have five people in our group, and these interruptions make our conversation a mess, and we can not communicate with each other efficiently. With the suggestions of Professor Casey, we learned how to become good listeners when others are talking.

Since there are four team members with English as secondary language in our team, the public speaking and presentation was a big problem for our team. We would get nervous and forget what to say in the presentation. After several practices of presentations, the encouragement from two professors and help from Alex (native speaker in our team), we had more confidence when we do presentation. In order to make our presentations good, we would spend a lot of time on practicing it.

After six weeks of study, IED has changed us a lot in using language - from using verbal words to the engineering words. When describing things, we are now more likely to use numbers and datas to make our view more reliable. We sometimes might find our points of view wrong when doing research to prove our opinions. This reminds us the necessity of doing enough research and study before making a conclusion.

Our project is done by five people and through weeks of work, we recognized the importance of merging working products and ideas into one. For example, there are bugs when we are making progression in our projects and we need to figure out whether the bugs come from the hardwares and softwares. We need to check step by step and communicate with each other to know their opinions about the bugs. Through our group meeting, we learned how to cooperate to solve problems.

We spend a lot of time when our group get into the storming stage. The problem we had is we were not clear about our goals. Every member in our group had different skills and saw our project in different aspects. The need to understand what goals we will be addressed and which one needs to be compromised on is the most important thing we worked through as a team in this stage. To reduce the confusion and conflict in the future, we made a clear rank to our goals and implemented accordingly.

When we comes to writing, the team worked together very well. Even though we all have experience in technical writing, the language barrier that four of us carried imposed great challenge to a smooth writing process. In this circumstance, Alex stepped up to take on the responsibility of proofreading and editing the memo. With each team member contributing to their designated sections, the memo is finally done.

Appendix H: User Manual

How System Works – At home

The solution that cares all about your safety and convenience.

Let us install for you!

Please cooperate with our work staff to install and test system embedded inside wall and ceiling.

SightRPI

How System Works – On Your Phone

The solution that cares all about your safety and convenience.

2

Download our App from app store.

- If you use **Android** phone, you can download it from **Google Play** store or from our website.
- If you use **iOS** devices, you can download it from **App Store**.

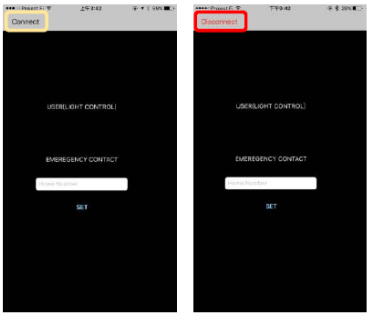
GET IT ON Google Play

Download on the App Store

3

Connect Bluetooth.

- Turn on Bluetooth on your phone and click on “Connect” button circled by yellow rectangle.
- App will automatically search for Bluetooth available for connection. Choose our system Bluetooth “SightRPI”.
- “Connect” will change to red “Disconnect” (as circled in red circle). If not, please check Bluetooth setting or contact us.



SightRPI

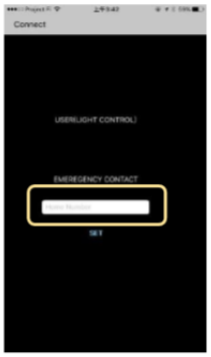
If you are user's **emergency contact**, skip to

4

On user's phone,

Add emergency contact.

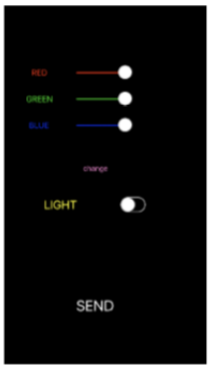
- Type in your emergency contact phone number in the slot circled by yellow rectangle.
- Press "SET".



5

Start safe life!

- Click on user mode "USER(LIGHT CONTROL)".
- To adjust light color
  - Adjust Red, Green, Blue attributes.
  - Click "change" right below.
- To switch ON/OFF light
  - Swipe "LIGHT" button.
  - Click "SEND" below.



SightRPI

6

On emergency contact's phone,

Free Mode

- Our system works correctly and there is no danger happening now.

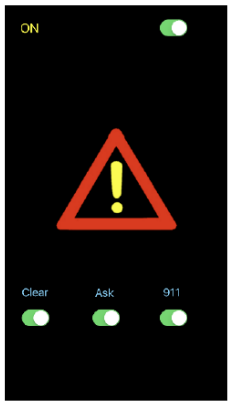
Emergency Mode

If there is emergency happens, your phone will pop up notification.

- To clear signal after you ensure no one is in danger at home, press "Clear".

  NOTE: Please contact the user at home then clear signal if appropriate.

- To call the one at home, press "Ask".
- To call EMS immediately, press "911".



SightRPI

Appendix I: All Code

Pseudo Code for Arduino UNO

```
// Send data from Arduino UNO
/* Data Type
      For debug only: ("number of people: %d, range: %d", numberofpeople,
range)
      For normal transmission: ("%d", emergencyHappens) -- only send 101 when
emergency happens, nothing during normal run
                              ("%d", switchStatus) -- 103 if ON, 104 if OFF
*/

// Pseudocode
#include <BluetoothSoftware>

int a = 0;                          // can use sbit (i.e. flag) instead

// in main function
while (in loop) {
      ...
      if (lightOn) {
            sendBluetooth(103);     // refresh switch ON/OFF on smartphone
app
      }
      else {
            sendBluetooth(104);
      }
      ...
      a = 0;
      a = emergencyHappens();       // Range sensor, Homemade function
      if (a == 1) {                 // emergency happens
            sendBluetooth(101);     // Homemade function
      }
      ...
}

// in function sendBluetooth
int/void sendBluetooth(int data) {  // int if need error checking -- see
"Hints for Final Report" on Google Drive
      sendData(data);
      delayForBluetooth(t1);        // delay for Bluetooth
}

// Receive data at Arduino UNO end
/* Data Type
      For Trial_1: parseInt() may not meet requirement.
      During free status:
```

```
            Trial_0: ("201" -- free status) + delay + ("%d", R) + delay +
("%d", G) + delay + ("%d", B) + delay + ("%d", switchOnOrOff) + delay
            Trial_1: ("201 %d %d %d %d", R, G, B, switchOnOrOff) + delay
      During emergency status:
            Trial_0: ("202" -- emergency status) + delay + ("100" -- SOS OR
"101" -- cleared) + delay
            Trial_1: ("202 100" OR "202 101") + delay
*/

// Pseudocode for Trial_0
#include <BluetoothSoftware>

int emergencyStatus = 0;                  // can use sbit (i.e. flag) instead

// in main function
while (in loop) {
      ...
      if (emergency happens) {
            ...
            confirmation system call;
            ...
            if (not OK) {
                  while (true) {
                        int tmp = 1;
                        tmp = readBluetooth();
                        delay();
                        if (tmp == 101) {
                              break;
                        }
                  }
            }
            ...
      }
      ...
}

// in function sendBluetooth
// DO NOT give control of light to users -- except: 1) user clears, 2) object
(i.e. fallen down person) is removed
// If user says OK, ask another OK until the range gets back to normal and
then into emergency case again.
// -- Idle mode in System flowchart
int readBluetooth() {
      int data = 0;
      if (available())
            readData();
      delayForBluetooth(t1);        // delay for Bluetooth
      return data;
}
```

```
// other functions may be required
```

Code for Arduino UNO

```
#include "Adafruit_VL53L0X.h"
#include <SoftwareSerial.h>
#define IRsensor1 9 // define the outside IR break beam sensor
#define IRsensor2 10// define the inner IR break beam sensor
/*"301 200" indicates that lights on "301 201" indicates that lights off */
/*********************************************************
 * all the functions I have created in the smart lighting:
 * void setup()   *void loop()      *void lightup()
 * void lightdown() *void blinklight() *void setColor
 * LoEroom()      *int readvalue()   *void sendvalue()
 * void iflighton() *void checkclear()  *void SFLM()
 * void colorchange()
 *********************************************************/

const int pingnum=12;
SoftwareSerial mySerial(4,5);
int counter;
int counter2;
int IRstate1=0;
int laststate1=0;
int IRstate2=0;
int laststate2=0;
int time1;
int time2=0;
int IRflag1;
int IRflag2;
int numofpeople=0; //the number of people inside the room
int lightonflag=0;// the flag reflects that if the flag is on
```

```
Adafruit_VL53L0X rs = Adafruit_VL53L0X();// the range sensor
int rs_range=0;// the distance between the ceiling and the object
int breaksignal;
int red;
int green;
int blue;
int red_store;
int green_store;
int blue_store;
int redpin=6;
int bluepin=8;
int greenpin=7;
int i=0;
int pushbuttom=18;
int pushbuttom2=19;
int YES=0;
int NO=0;
int falling=0;
char sendingdata;
int time_differ;
int startsignal;
int endsignal;
boolean clearing;
boolean mobileon;
boolean fall=false;
boolean sent=false;

void setup() {

  // put your setup code here, to run once:
  Serial.begin(9600);
  mySerial.begin(9600);
  if (!rs.begin()) {
      Serial.println(F("Failed to boot VL53L0X"));
```

```
      while(1);
  }
  pinMode(IRsensor1,INPUT);
  pinMode(IRsensor2,INPUT);
  pinMode(pushbuttom,INPUT);
  digitalWrite(IRsensor1,HIGH);//set high impedance
  digitalWrite(IRsensor2,HIGH);//set high impedance
  red_store = 255;
  green_store = 255;
  blue_store = 255;
}

void loop() {
  VL53L0X_RangingMeasurementData_t measure;// the variable from the  VL530L0X
liabrary

  // put your main code here, to run repeatedly:
  YES=digitalRead(pushbuttom);
  LoEroom();
  startsignal=0;
  startsignal=readvalue();
  if (startsignal == 257)
      SLFM();
  if (startsignal == 250)
      colorchange();

  while (numofpeople>0){
      LoEroom();
      iflighton();
      Serial.print("num of people : ");
      Serial.print(numofpeople);
      Serial.print("\n");
      Serial.print("Time:");
      Serial.print(millis());
```

```
      Serial.print("\n");
      if(mobileon==false){
      lightup();
      mySerial.print("301 300");
      }
      startsignal=0;
      startsignal=readvalue();
      if (startsignal == 257)
      SLFM();
      if (startsignal == 250)
      colorchange();

      while(1){
      fall= false;
      LoEroom();
      if(numofpeople==0){
      red_store = 255;
      green_store = 255;
      blue_store = 255;
      break;
      }
      startsignal=0;
      startsignal=readvalue();
      if (startsignal == 257)
      SLFM();
      if (startsignal == 250)
      colorchange();
      //rs_range=measure.RangeMilliMeter;
      rs.rangingTest(&measure, false);
      if (measure.RangeStatus != 4) {  // phase failures have incorrect data
            //  Serial.print("Distance (mm): ")
;Serial.println(measure.RangeMilliMeter);
            rs_range=measure.RangeMilliMeter;
            LoEroom();
```

```
      }
      Serial.print("num of people : ");
      Serial.print(numofpeople);
      LoEroom();
      if((rs_range>200)&&(rs_range<230)){
            counter=0;
            LoEroom();
          while((rs_range>215)&&(rs_range<227)){
            rs.rangingTest(&measure, false);
            rs_range=measure.RangeMilliMeter;
            counter++;
            Serial.print("Distance (mm): ")
;Serial.println(measure.RangeMilliMeter);
            Serial.print(counter);
            Serial.print("\n");
            if(counter==10 ) {

            fall=true;
            break;
            }
            }
      if(counter==10)break;



      }
      }



      if(fall==true){
      boolean nonemergency=false;
      clearing=false;
      LoEroom();
      startsignal=0;
      Serial.print("\n");
```

```
      Serial.print("Time:");
      Serial.print(millis());
      Serial.print("\n");
      Serial.print("Distance (mm): ")
;Serial.println(measure.RangeMilliMeter);
      Serial.print("Are you OK\n");
      mySerial.print("Are you ok");
      YES=digitalRead(pushbuttom);
      time1=millis();
      time2=millis();
      time_differ=time2-time1;
      while(time_differ<5000){
      time2=millis();
      time_differ=time2-time1;
      YES=digitalRead(pushbuttom);
      NO=digitalRead(pushbuttom2);
      if(YES==LOW){
            nonemergency=true;
            while(1){
            rs.rangingTest(&measure, false);
            rs_range=measure.RangeMilliMeter;
            if(rs_range>232) break;

            }
            Serial.print("YES");
            //mySerial.print("YES");
            break;
            }
      if(NO==LOW){

            break;
            }
      }
      if(nonemergency==false){
```

```
            while(clearing==false){
            counter2=0;
            YES=digitalRead(pushbuttom);
            blinklight();
            checkclear();
            mySerial.print("201 200");
            rs.rangingTest(&measure, false);
            rs_range=measure.RangeMilliMeter;

            // Serial.print("Distance (mm): ")
;Serial.println(measure.RangeMilliMeter);
            if(rs_range>232){
            while(rs_range>232){
                  rs.rangingTest(&measure, false);
                  rs_range=measure.RangeMilliMeter;
                  counter2++;
                  if(counter2==5)break;
            }
            break;
            }
            }
      }
      mySerial.print("201 201");
      rs.rangingTest(&measure, false);
      rs_range=measure.RangeMilliMeter;
      }
      lightup();
  }
  LoEroom();
  if(numofpeople==0){
      lightdown();
      mySerial.print("301 301");
      delay(50);
  }
```

```
  Serial.print("nobody is in the room\n");

}

void lightup(){
  //turn on the light
  analogWrite(redpin,red_store);
  analogWrite(bluepin,blue_store);
  analogWrite(greenpin,green_store);
}

void lightdown(){
  //turn off the light
  analogWrite(redpin,0);
  analogWrite(bluepin,0);
  analogWrite(greenpin,0);
}

void blinklight(){
  //blink the bulb
  analogWrite(redpin,255);
  analogWrite(bluepin,0);
  analogWrite(greenpin,0);
  delay(100);
  analogWrite(redpin,0);
  analogWrite(bluepin,0);
  analogWrite(greenpin,0);
  delay(100);
  // Serial.print("******");
}

void setColor(int red, int green, int blue){
  // the function used to change the color the RGB
  analogWrite(redpin,red);
```

```
  analogWrite(bluepin,blue);
  analogWrite(greenpin,green);
  red_store = red;
  green_store = green;
  blue_store = blue;
}

void LoEroom(){
  //the function used to leave or enter the room
  IRstate1=digitalRead(IRsensor1);//4
  delay(50);
  IRstate2=digitalRead(IRsensor2);//5
  int happen=0;
  if((!IRstate1)&&(laststate1)&&(happen==0)){
      //outer sensor
      while(1){
      IRstate2=digitalRead(IRsensor2);//5
      if((!IRstate2)&&(laststate2)){
      numofpeople++;
      happen=1;
      break;
      }
      laststate2=IRstate2;
      }
  }

  if((!IRstate2)&&(laststate2)&&(happen==0)){
      while(1){
      //inner sensor
      IRstate1=digitalRead(IRsensor1);//4
      if((!IRstate1)&&(laststate1)){
      if(numofpeople==0){
      break;
      }
```

```
      numofpeople--;
      happen=1;
      break;
      }
      laststate1=IRstate1;
      }
      }
  laststate1=IRstate1;
  laststate2=IRstate2;

}

int readvalue(){
  int c;
  if(mySerial.available()){
      c=mySerial.parseInt();
      return c;
  }
}

void sendvalue(char data) {  // int if need error checking -- see "Hints for
Final Report" on Google Drive
   mySerial.print(data);
}

void iflighton(){
  if((red==0)&&(green==0)&&(blue==0)){
      mobileon=false;
  }
  else{
      mobileon=true;
  }
}
```

```
void checkclear(){
      if (mySerial.available())
      Serial.print("here");
      startsignal=readvalue();
      if(startsignal==256){
      endsignal=0;
      while(endsignal!=256){
      if (mySerial.available()){
            Serial.print(clearing);
            endsignal=readvalue();
            if (endsignal == 256) break;
            if(endsignal==128){
            clearing=true;
            }
            }
      }
   }
}

void SLFM(){//switch on / off lights from mobile app
      endsignal=0;
      while(endsignal!=257){
      if (mySerial.available()){
            Serial.print(clearing);
            endsignal=readvalue();
            if (endsignal == 257) break;
            if(endsignal==103){
            mobileon=true;

            lightup();
            }
            if(endsignal==104){
            mobileon=false;
            lightdown();
```

```
            }
            }
      }
   //}
}

void colorchange(){//the function used for color change
      //startsignal=0;
      //if (mySerial.available())
      //  startsignal=readvalue();
      Serial.print("***");
    Serial.print("getcolorchangesignal");
      Serial.print("***\n");
      //if(startsignal==250){
            red_store=readvalue();
            green_store=readvalue();
            blue_store=readvalue();
            readvalue();
      setColor(red_store,green_store,blue_store);
      //}
}
```

Phone Application Code

AppDelegate.swift

```
import UIKit
import UserNotifications

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?
```

```
    func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey:
Any]?) -> Bool {
        // Override point for customization after application launch.
        application.applicationIconBadgeNumber = 0;

        UNUserNotificationCenter.current().requestAuthorization(options:
[.alert, .sound, .badge]) {(authorized:Bool,error:Error?) in
            if !authorized{
                print("need to allowed noticification")
            }
        }

        let answerOne = UNNotificationAction(identifier: "answerOne", title:
"Open", options: [.foreground])
        let category = UNNotificationCategory(identifier: "category",
actions: [answerOne], intentIdentifiers: [], options: [])

UNUserNotificationCenter.current().setNotificationCategories([category])
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {

    }

    func applicationDidEnterBackground(_ application: UIApplication) {

    }

    func applicationWillEnterForeground(_ application: UIApplication) {

    }

    func applicationDidBecomeActive(_ application: UIApplication) {
          }

    func applicationWillTerminate(_ application: UIApplication) {
          }
}
```

FirstViewController.swift

```
import UIKit
import CoreBluetooth

var rcv = 0
var textGet = ""
```

```
var phoneNum = ""

class FirstViewController:
UIViewController,BluetoothSerialDelegate,UITextFieldDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()

        // init serial
        serial = BluetoothSerial(delegate: self)

        reloadView()

        NotificationCenter.default.addObserver(self, selector:
#selector(FirstViewController.reloadView), name:
NSNotification.Name(rawValue: "reloadStartViewController"), object: nil)

        // to dismiss the keyboard if the user taps outside the textField
while editing
        let tap = UITapGestureRecognizer(target: self, action:
#selector(FirstViewController.dismissKeyboard))
        tap.cancelsTouchesInView = false
        view.addGestureRecognizer(tap)

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()

    }

    @IBOutlet weak var bleConnect: UIBarButtonItem!
    @IBOutlet weak var phoneNumber: UITextField!

    @IBAction func setNumber(_ sender: Any) {
        phoneNum = phoneNumber.text!

        phoneNumber.text = ""
        phoneNumber.placeholder = "recorded!"

    }

    func reloadView() {
        // in case we're the visible view again
        serial.delegate = self

        if serial.isReady {

            bleConnect.title = "Disconnect"
```

```
            bleConnect.tintColor = UIColor.red
            bleConnect.isEnabled = true
        } else if serial.centralManager.state == .poweredOn {

            bleConnect.title = "Connect"
            bleConnect.tintColor = view.tintColor
            bleConnect.isEnabled = true
        } else {

            bleConnect.title = "Connect"
            bleConnect.tintColor = view.tintColor
            bleConnect.isEnabled = false
        }
    }
    func dismissKeyboard(){
        phoneNumber.resignFirstResponder()
    }

    //MARK: BluetoothSerialDelegate

    func serialDidReceiveString(_ message: String) {

        rcv = 1
        textGet = ""
        textGet += message

        if textGet == "301 300" {
            signal = 1
            NotificationCenter.default.post(name: Notification.Name(rawValue:
"FreeViewReload"), object: nil)
        }
        if textGet == "301 301"{
            signal = 0
            NotificationCenter.default.post(name:Notification.Name(rawValue:
"FreeViewReload"), object: nil)
        }
        if textGet == "201 200"{
            emerg = 1
            NotificationCenter.default.post(name:
Notification.Name(rawValue:"EmergViewReload"), object: nil)
            NotificationCenter.default.post(name:
Notification.Name(rawValue:"EmergFreeReload"), object: nil)
        }
        if textGet == "201 201"{
            emerg = 0
            NotificationCenter.default.post(name:
Notification.Name(rawValue:"EmergViewReload"),object:nil)
        }
        if textGet == "Are you ok"{
```

```
            NotificationCenter.default.post(name:
Notification.Name(rawValue:"AreYouOk"),object:nil)
        }
    }

    func serialDidDisconnect(_ peripheral: CBPeripheral, error: NSError?) {
        reloadView()

        let hud = MBProgressHUD.showAdded(to: view, animated: true)
        hud?.mode = MBProgressHUDMode.text
        hud?.labelText = "Disconnected"
        hud?.hide(true, afterDelay: 1.0)
    }

    func serialDidChangeState() {
        reloadView()
        if serial.centralManager.state != .poweredOn {

            let hud = MBProgressHUD.showAdded(to: view, animated: true)
            hud?.mode = MBProgressHUDMode.text
            hud?.labelText = "Bluetooth turned off"
            hud?.hide(true, afterDelay: 1.0)
        }
    }
}
```

ViewController.swift

```
import UIKit
import CoreBluetooth
import UserNotifications

var signal = 0
var emerg  = 0
var red = 255
var green = 255
var blue = 255
var flag = 0

class ViewController: UIViewController, BluetoothSerialDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()
        UNUserNotificationCenter.current().delegate = self
        refresh()
        NotificationCenter.default.addObserver(self, selector:
#selector(ViewController.refresh), name:
NSNotification.Name(rawValue:"EmergViewReload"), object: nil)
```

```
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}
@IBOutlet var lightOnOff: UILabel!
@IBOutlet var imageView: UIImageView!
@IBOutlet weak var clear: UILabel!
@IBOutlet weak var ask: UILabel!
@IBOutlet weak var askSwitch: UISwitch!
@IBOutlet weak var emergLable: UILabel!
@IBOutlet weak var emergSwitch: UISwitch!
@IBOutlet weak var testSwitch: UISwitch!

@IBOutlet weak var status: UISwitch!
@IBAction func lightChange(_ sender: UISwitch) {
    let image0 = UIImage(named:"Snip20170617_1.png")
    let image1 = UIImage(named:"eye.png")
    if status.isOn {
        emerg = 1
        self.lightOnOff.text = String(emerg)
        self.imageView.image = image0
    }
    else{
        emerg = 0
        self.lightOnOff.text = String(emerg)
        self.imageView.image = image1
        serial.sendMessageToDevice("256 128 256")
        refresh()
    }

}

func refresh(){
    if emerg == 1{
        flag+=1
        //show all the buttons
        testSwitch.isOn = true
        status.isHidden = false
        clear.isHidden = false
        ask.isHidden = false
        askSwitch.isHidden = false
        emergLable.isHidden = false
        emergSwitch.isHidden = false
        status.isOn = true
        askSwitch.isOn = true
        emergSwitch.isOn = true
       //change the text and the imageView
```

```
            self.lightOnOff.text = String(emerg)
            self.imageView.image = UIImage(named:"Snip20170617_1.png")
            if flag==1{
                popNotic()
            }
        }
        else{
            flag = 0
            testSwitch.isOn = false
            status.isHidden = true
            clear.isHidden = true
            ask.isHidden = true
            askSwitch.isHidden = true
            emergLable.isHidden = true
            emergSwitch.isHidden = true
            self.lightOnOff.text = String(emerg)
            self.imageView.image = UIImage(named:"eye.png")
        }
    }
    func popNotic()
    {
        //send noticifications to User
        let content = UNMutableNotificationContent()
        content.title = "Emergency at Home!"
        content.subtitle = "Need help!"
        content.body = "open application to check what to do!"
        content.badge = 1
        content.categoryIdentifier = "category"

        let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 1,
repeats: false)
        let requestIdentifier = "emergNote"
        let request = UNNotificationRequest(identifier: requestIdentifier,
content: content, trigger: trigger)
        UNUserNotificationCenter.current().add(request,
withCompletionHandler: {error in
            // handle error
        })
    }
    @IBAction func callHome(_ sender: Any) {
            let url = URL(string: "tel://"+phoneNum)!
            UIApplication.shared.open(url, options: [:], completionHandler:
nil)
    }
    @IBAction func callPolice(_ sender: Any) {
        let url = URL(string: "tel://911")!
        UIApplication.shared.open(url, options: [:], completionHandler: nil)
    }
    @IBAction func fakeEmerg(_ sender: Any) {
```

```
        if  testSwitch.isOn{
            emerg = 1
            refresh()
        }else{
            emerg = 0
            refresh()
        }
    }
    //MARK: BluetoothSerialDelegate
    func serialDidReceiveString(_ message: String) {


    }
    func serialDidDisconnect(_ peripheral: CBPeripheral, error: NSError?) {

    }

    func serialDidChangeState() {
    }
}

extension ViewController: UNUserNotificationCenterDelegate{
    func userNotificationCenter(_ center: UNUserNotificationCenter,
willPresent notification: UNNotification, withCompletionHandler
completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {
        completionHandler([.alert, .sound])
    }
    func userNotificationCenter(_ center: UNUserNotificationCenter,
didReceive response: UNNotificationResponse, withCompletionHandler
completionHandler: @escaping () -> Void) {
        switch response.actionIdentifier {
        case "Open":
            break
        default:
            break
        }
    }
}
```

ScannerViewController.swift

```
import UIKit
import CoreBluetooth

class ScannerViewController: UIViewController,UITableViewDataSource,
UITableViewDelegate, BluetoothSerialDelegate {

    @IBOutlet weak var tryAgainButton: UIBarButtonItem!
```

```
    @IBOutlet weak var tableView: UITableView!

    //MARK: Variables

    /// The peripherals that have been discovered (no duplicates and sorted
by asc RSSI)
    var peripherals: [(peripheral: CBPeripheral, RSSI: Float)] = []

    /// The peripheral the user has selected
    var selectedPeripheral: CBPeripheral?

    /// Progress hud shown
    var progressHUD: MBProgressHUD?


    override func viewDidLoad() {
        super.viewDidLoad()

        // tryAgainButton is only enabled when we've stopped scanning
        tryAgainButton.isEnabled = false

        // remove extra seperator insets (looks better imho)
        tableView.tableFooterView = UIView(frame: CGRect.zero)

        // tell the delegate to notificate US instead of the previous view if
something happens
        serial.delegate = self

        if serial.centralManager.state != .poweredOn {
            title = "Bluetooth not turned on"
            return
        }

        // start scanning and schedule the time out
        serial.startScan()
        Timer.scheduledTimer(timeInterval: 10, target: self, selector:
#selector(ScannerViewController.scanTimeOut), userInfo: nil, repeats: false)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()

    }

    /// Should be called 10s after we've begun scanning
    func scanTimeOut() {
        // timeout has occurred, stop scanning and give the user the option
to try again
        serial.stopScan()
```

```
        tryAgainButton.isEnabled = true
        title = "Done scanning"
    }

    /// Should be called 10s after we've begun connecting
    func connectTimeOut() {

        // don't if we've already connected
        if let _ = serial.connectedPeripheral {
            return
        }

        if let hud = progressHUD {
            hud.hide(false)
        }

        if let _ = selectedPeripheral {
            serial.disconnect()
            selectedPeripheral = nil
        }

        let hud = MBProgressHUD.showAdded(to: view, animated: true)
        hud?.mode = MBProgressHUDMode.text
        hud?.labelText = "Failed to connect"
        hud?.hide(true, afterDelay: 2)
    }

    //MARK: UITableViewDataSource

    func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        return peripherals.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {
        // return a cell with the peripheral name as text in the label
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell")!
        let label = cell.viewWithTag(1) as! UILabel!
        label?.text = peripherals[(indexPath as
NSIndexPath).row].peripheral.name
        return cell
    }
```

```
    //MARK: UITableViewDelegate

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath) {

        tableView.deselectRow(at: indexPath, animated: true)

        // the user has selected a peripheral, so stop scanning and proceed
to the next view
        serial.stopScan()
        selectedPeripheral = peripherals[(indexPath as
NSIndexPath).row].peripheral
        serial.connectToPeripheral(selectedPeripheral!)
        progressHUD = MBProgressHUD.showAdded(to: view, animated: true)
        progressHUD!.labelText = "Connecting"


        Timer.scheduledTimer(timeInterval: 10, target: self, selector:
#selector(ScannerViewController.connectTimeOut), userInfo: nil, repeats:
false)
    }


    //MARK: BluetoothSerialDelegate

    func serialDidDiscoverPeripheral(_ peripheral: CBPeripheral, RSSI:
NSNumber?) {
        // check whether it is a duplicate
        for exisiting in peripherals {
            if exisiting.peripheral.identifier == peripheral.identifier {
return }
        }
        // add to the array, next sort & reload
        let theRSSI = RSSI?.floatValue ?? 0.0
        peripherals.append(peripheral: peripheral, RSSI: theRSSI)
        peripherals.sort { $0.RSSI < $1.RSSI }
        tableView.reloadData()
    }

    func serialDidFailToConnect(_ peripheral: CBPeripheral, error: NSError?)
{
        if let hud = progressHUD {
            hud.hide(false)
        }

        tryAgainButton.isEnabled = true

        let hud = MBProgressHUD.showAdded(to: view, animated: true)
        hud?.mode = MBProgressHUDMode.text
```

```
        hud?.labelText = "Failed to connect"
        hud?.hide(true, afterDelay: 1.0)
    }

    func serialDidDisconnect(_ peripheral: CBPeripheral, error: NSError?) {
        if let hud = progressHUD {
            hud.hide(false)
        }

        tryAgainButton.isEnabled = true

        let hud = MBProgressHUD.showAdded(to: view, animated: true)
        hud?.mode = MBProgressHUDMode.text
        hud?.labelText = "Failed to connect"
        hud?.hide(true, afterDelay: 1.0)

    }

    func serialIsReady(_ peripheral: CBPeripheral) {
        if let hud = progressHUD {
            hud.hide(false)
        }

        NotificationCenter.default.post(name: Notification.Name(rawValue:
"reloadStartViewController"), object: self)
        dismiss(animated: true, completion: nil)
    }

    func serialDidChangeState() {
        if let hud = progressHUD {
            hud.hide(false)
        }

        if serial.centralManager.state != .poweredOn {
            NotificationCenter.default.post(name: Notification.Name(rawValue:
"reloadStartViewController"), object: self)
            dismiss(animated: true, completion: nil)
        }
    }


//MARK: IBActions

    @IBAction func cancel(_ sender: Any) {
        // go back
        serial.stopScan()
        dismiss(animated: true, completion: nil)
    }
```

```
    @IBAction func tryAgain(_ sender: Any) {
        // empty array an start again
        peripherals = []
        tableView.reloadData()
        tryAgainButton.isEnabled = false
        title = "Scanning ..."
        serial.startScan()
        Timer.scheduledTimer(timeInterval: 10, target: self, selector:
#selector(ScannerViewController.scanTimeOut), userInfo: nil, repeats: false)
    }
}
```

FreeModeViewController.swift

```
import UIKit
import CoreBluetooth
import AVFoundation
var msg = ""
var play = 0
class FreeModeViewController: UIViewController,BluetoothSerialDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()

        reloadV()
        NotificationCenter.default.addObserver(self, selector:
#selector(FreeModeViewController.reloadV), name:
NSNotification.Name(rawValue:"FreeViewReload"), object: nil)
        NotificationCenter.default.addObserver(self, selector:
#selector(FreeModeViewController.reloadV), name:
NSNotification.Name(rawValue:"EmergFreeReload"), object: nil)
        NotificationCenter.default.addObserver(self, selector:
#selector(FreeModeViewController.playSound), name:
NSNotification.Name(rawValue:"AreYouOk"), object: nil)
        Red.value = Float(red)
        Green.value = Float(green)
        Blue.value = Float(blue)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    @IBOutlet weak var Red: UISlider!
    @IBOutlet weak var Green: UISlider!
    @IBOutlet weak var Blue: UISlider!
    @IBOutlet weak var light: UISwitch!
    @IBOutlet weak var test: UILabel!
```

```
    var audioPlayer = AVAudioPlayer()

    func playSound(){
        //set the sound file name
        do{
            play = 1
            audioPlayer = try  AVAudioPlayer(contentsOf:
URL.init(fileURLWithPath:Bundle.main.path(forResource: "are_you_ok.mp3",
ofType: "mp3")!))
            audioPlayer.prepareToPlay()
            audioPlayer.play()
            play = 0
        }
        catch{
            print(error)
        }
    }

    func playAlarm(){
        do{
            audioPlayer = try  AVAudioPlayer(contentsOf:
URL.init(fileURLWithPath:Bundle.main.path(forResource: "Alarm", ofType:
"mp3")!))

            audioPlayer.play()
            play = 1
        }
        catch{
            print(error)
        }
        self.test.text = textGet
    }
    func reloadV(){
        if emerg == 1{
            if play == 0{
                playAlarm()
            }
        }
        if emerg == 0{
            if play == 1{
                audioPlayer.stop()
            }
            play = 0
        }
        if signal == 1{
            self.light.setOn(true, animated: true)
        }
        if signal == 0{
            self.light.setOn(false, animated: true)
```

```
        }

        Red.value = Float(red)
        Green.value = Float(green)
        Blue.value = Float(blue)

        self.test.text = textGet

    }
    @IBAction func testPlay(_ sender: Any) {
        playSound()
        Timer.scheduledTimer(timeInterval: 5, target: self, selector:
#selector(FreeModeViewController.playAlarm), userInfo: nil, repeats: false)
    }
    @IBAction func changeColor(_ sender: Any) {
        red = Int(Red.value)
        green = Int(Green.value)
        blue = Int(Blue.value)
        msg = ""
        msg += "250 "
        msg += String(red) + " " + String(green) + " " + String(blue)
        msg += " 250"
        self.test.text = msg
        serial.sendMessageToDevice(msg)
    }
    @IBAction func sendData(_ sender: Any) {

        if !serial.isReady {
            let alert = UIAlertController(title: "Not connected", message:
"What am I supposed to send this to?", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "Dismiss", style:
UIAlertActionStyle.default, handler: { action -> Void in
self.dismiss(animated: true, completion: nil) }))
            present(alert, animated: true, completion: nil)
        }

        msg = ""
        if signal == 1{
            msg = "257 103 257"
        }else{
            msg = "257 104 257"
        }
        serial.sendMessageToDevice(msg)
        self.test.text = "sent"
        if rcv == 1{
            self.test.text = textGet
        }
    }
```

```
    @IBAction func lightChange(_ sender: UISwitch ) {
        if light.isOn{
            signal = 1
        }else{
            signal = 0
        }
    }

    func serialDidReceiveString(_ message: String) {


    }

    func serialDidDisconnect(_ peripheral: CBPeripheral, error: NSError?) {

    }

    func serialDidChangeState() {
    }
}
```