

Trinity University

Digital Commons @ Trinity

Engineering Senior Design Reports

Engineering Science Department

5-7-2021

K-12 Security System Final Project Report

Josh Izi

Trinity University

T.J. Raniszkeski

Trinity University

Brittany Taylor

Trinity University

Follow this and additional works at: https://digitalcommons.trinity.edu/engine_designreports

Repository Citation

Izi, Josh; Raniszkeski, T.J.; and Taylor, Brittany, "K-12 Security System Final Project Report" (2021).
Engineering Senior Design Reports. 46.

https://digitalcommons.trinity.edu/engine_designreports/46

This Restricted Campus Only is brought to you for free and open access by the Engineering Science Department at Digital Commons @ Trinity. It has been accepted for inclusion in Engineering Senior Design Reports by an authorized administrator of Digital Commons @ Trinity. For more information, please contact jcostanz@trinity.edu.

K-12 Security System Final Project Report

The Security Team:

Josh Izi

TJ Ranieszski

Brittany Taylor

May 7, 2021

1. Executive Summary

Current emergency technology for school shootings is very basic, relying largely on faculty training and basic alarm technology. Alarms are usually activated via emergency buttons arranged throughout the school or online programs requiring a log-in. These methods are cheap and easy to install, however running to an emergency button or logging into a computer program takes time and current school alarms provide very little information about the location of the threat. The K-12 Security System Team proposes the design and implementation of a system to aid school authorities to minimize the risk posed to students and staff in case of gun threats in K-12 schools.

The system must be quick and unobtrusive to activate and should be able to locate and communicate the source of the activation. The system must also have two levels of alarm- a subtle alarm for pre-shooter situations where de-escalation may be possible and a blaring alarm for active shooter situations. The preliminary system design should also be cheap enough for a school to purchase, which our advisor recommended to be a total budget of around \$3,200. Since this is a proof of concept, our prototype stayed well under that constraint as well as Trinity's own \$1200 budget cap. The system also needs to be active for the entirety of the school day, so any mobile components should have a battery that lasts at least a semester or a rechargeable battery.

To satisfy the requirements the security team broke the project up into 3 subsystems. These subsystems include the central computing unit (CCU), portal beacon, and handheld fob device. Splitting up the project in this manner allowed for us to better track our progress and ensure that each subsystem can run on its own.

The team experienced time production slowdowns due to the pandemic keeping one of our three members off campus, winter storm delivery delays, and several faulty Arduino systems. Despite these limitations, the K-12 security team was able to build the fob, portal beacon, and CCU subsystems and successfully update a database using signals sent from a phone-sized, handheld fob. The prototype has multiple alarm levels and stores both the alarm level and ID number of the fob.

2. Introduction

In the US, there have been a total of 1,316 school shooting incidents since 1970. 18% of school shootings have taken place since the tragedy at Sandy Hook Elementary School in December of 2012 ^[1]. 2018 brought us the most prolific year of school shootings with a total of 116 incidents in K-12 schools across the US ^[2]. As this unfortunate trend seems to be on the rise a more effective security solution is needed now more than ever. Partnering with Beckwith Electronic Engineering, K-12 schools seek a low-cost communication system that allows teachers to discreetly communicate the level of a threat along with the exact location of said threat to local authorities.

The numbers mentioned above are not all active shooter situations. Some of these incidents are ones where a student brought a gun into the building with the intention to do harm but was successfully neutralized before any harm was done. For example, of the 116 reported incidents in 2018, 105 incidents were of non-active gunmen ^[2]. Knowing this, a more effective way of discreetly communicating potential threats before they turn into active threats is a key aspect of the design problem.

The currently used security systems rely primarily on faculty training and basic alarm technology. The most common security systems include emergency buttons located on walls or websites that require a log-in for alarm activation. These systems are cheap and easy to install, but also inconvenient and take time to reach or access. The Security Team aims to create a security system that is quick and unobtrusive to activate and can locate and communicate the source of the alarm's activation.

2.1 Constraints

The key aspect of this project is to design, implement, and test an improved security device that allows teachers to trigger an alarm in the event of a gun threat in a school building. Our project constraints include remaining under the \$1,200 budget provided by the Trinity University Engineering Department. While building our prototype it was also required that we remain conscious of a realistic budget that would be used by schools in order to implement a system such as this school wide with an approximate budget of \$3,215 per school.

Our system must also be unobtrusive to the user and ideally discreet so that it is nearly undetectable from someone standing six feet away. A system that is discreet and unobtrusive from a small device on their person would allow for quicker and safer alarm activation. We also want to create a system that can locate a threat within room-level accuracy and has two different levels of alarm (one that is loud and one that is more subtle but can still effectively evacuate the building). We also want to limit accidental alarm activation. Finally, our system needs to be active during all hours when there are people in the school building. Any components of our system that use a battery need to either last for half a semester or last a day and be rechargeable.

2.2 Design Subsystems

We divided our project into three major systems with different goals: the alarm activation system, the location determination system, and the alarm itself. The primary goal of the alarm activation system was to create an unobtrusive mobile alarm activation method using fobs and RF signals. The primary goal of the location determination system was to determine the location where the alarm was activated with room-level accuracy. Our third system, the alarm itself, aimed to create a two-level alarm response, the first being a subtle alarm undetectable by the threat for pre-shooter situations and the second being a blaring alarm for active shooter situations.

We also divided the design into 3 major components: the central computer unit (CCU), portal beacon, and fob as shown in Fig. 2. The CCU is the heart of the design where all the data is stored and the alarms are initiated. It receives information from the portal beacons and fobs, stores the information in its database, and, when an alarm is called for, carries out the alarm response. Portal beacons are the devices responsible for sending information via Ethernet to the CCU. They are positioned in every room in the school and will serve as signal receivers for the handheld fobs. We plan for every staff member to receive a fob and wear it on their person at all times. Fobs constantly send out a Bluetooth signal to keep track of the user's location in the school building. The two buttons on the fob will trigger the security system's two different alarms.

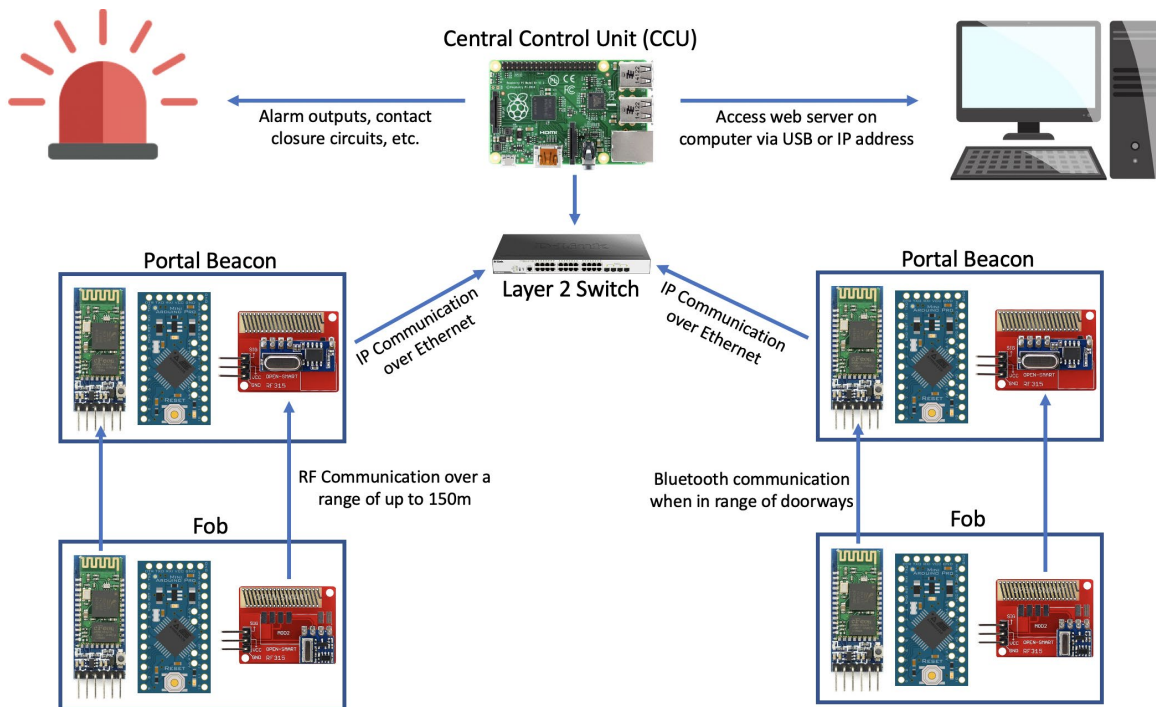


Figure 1. Diagram demonstrating how the subsystems communicate with each other

2.3 Codes and Standards

One standard applicable to our project is the Americans with Disabilities Act or ADA. Title I of the amended Americans with Disabilities Act (ADA) pertains to employees with disabilities and is relevant to our project because faculty and staff in K-12 schools may have disabilities. In section 12112, the ADA states that it is unlawful to fail to provide “reasonable accommodations to the known physical or mental limitations of an otherwise qualified individual with a disability who is an applicant or employee” [3]. Our security system, including components such as the fob activator and the faculty alert system, must be accessible to faculty and staff with disabilities such as visual impairments, hearing impairments, and movement impairments. This means that fobs should have large and easily pressed buttons with both visual and touch-based indicators. Braille and other textured additions have already been considered and will be included in the design process of the fobs. For the prototype tests, we seek to involve individuals who have never seen the fobs before and have them attempt to use them in a variety of different situations, including with their eyes closed. Unless we find a visually impaired volunteer, this would not test the effectiveness of Braille specifically, but it can test how easy it is to find the buttons with touch only.

Both licensed and unlicensed devices using RF signals are bound by The Federal Communication Commission (FCC) laws. In order to satisfy FCC rule part 2.1091, which is RF exposure compliance, Raspberry Pi 4B must be used at a distance of at least 20 cm from all persons. This means that when the Pi is operating, we must keep the device at appropriate distances. Following the FCC standards, our end product must comply with KDB Publication 996369 for modules and module certification. This step will most likely be taken care of during the final steps of the design project once all prototypes are finished.

We plan on using RF signals in the 260-470 MHz band in our fobs. The 260-470 MHz band is contained in Part 15.231 A-D of the FCC [4]. According to the law, ID codes and control or command signals are allowed transmission types. We will only use RF signals for the emergency alert signal, which will include an ID number and alarm level, so our design is legal under FCC law. In the case of a manually activated RF signal, the transmission must stop within 5 seconds.

Our design will have components that communicate via Bluetooth. Bluetooth communication falls under the IEEE standard 802.15.1-2002. This specification outlines the industry standard for short range RF communication on portable personal electronics. Luckily, the work is done for us. The devices we use all have Bluetooth capability and this standard is held by the manufacturer when making this device. Fig. 2 is a representation of the steps they take to assure the standard is met.

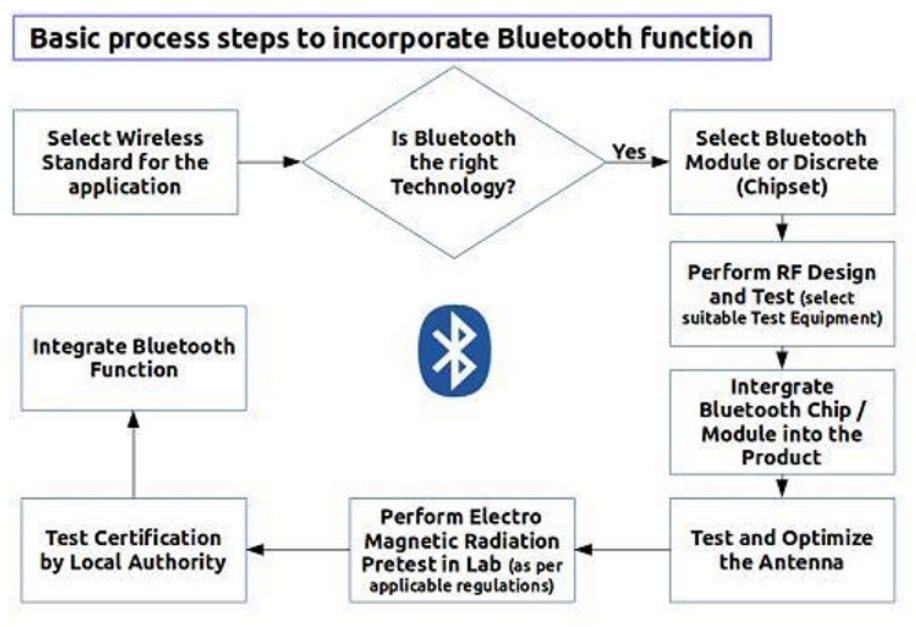


Figure 2. Process to establish Bluetooth connectivity on a device

As mentioned, all the required testing is completed by the manufacturer. This allows our group to seamlessly connect our components and communicate the necessary information for our design.

3. Overview of Final Design

Our proposed design primarily consists of a fob, CCU, and the school's existing alarm infrastructure. These systems interact to form the three subsystems mentioned previously: the alarm activation system, the location determination system, and the alarm response system. The fob and CCU are involved in multiple systems, while the school's alarm infrastructure is involved primarily in the alarm response system.

Beginning with the alarm activation system, the fob will have multiple buttons that will function as the interface of the alarm activation system. The fob will function similarly to a garage door opener, so it will be a radio transmitter [5]. Pressing either of the buttons will cause an RF signal to be transmitted, but the transmitter will be inactive otherwise, thus preserving battery life for the fob's other functions. Signals transmitted from the fob will feature a set of numbers coinciding with the ID number of the fob followed by a final digit that will coincide with the level of alarm being activated. For example, a fob with ID number 123 would send the signal 1231 when one button is pressed and 1232 when the other button is pressed.

The RF receivers are located in the portal beacons located at doorways and choke points. The receiver is always powered, but since it is stationary, unlike the fob, it can use the school's electricity. The range of each receiver depends on each individual school's layout; however the complete system of RF beacons must cover the school building in its entirety. Wireless range is calculated using (1) where maximum path loss is (2) [6]. f is signal frequency in MHz.

$$PL [dB] = 10 \frac{4\pi f d - 32.44 - 20 \log(d)}{20} \quad (1)$$

$$\begin{aligned} PL [dB] &= 10 \log \left(\frac{4\pi f d}{c} \right)^2 - 32.44 - 20 \log(d) \\ &+ 20 \log(f) - 20 \log(d) \end{aligned} \quad (2)$$

Signal losses vary from room to room due to walls and other obstacles present in the school building, so for maximum power efficiency, the transmitter power will need to be altered on a room-by-room basis.

Portal beacons located at doorways and choke points throughout the school receive RF signals from the fobs and send them to the CCU. The fob must have a recognizable ID number for the signal to be accepted, and the CCU can use its database in order to identify the owner of the fob and the location where they were most recently detected.

The location determination system, like the activation system, uses the fob, portal beacons, and CCU, however the system differs from the activation system in several ways. The location determination system uses Bluetooth signals instead of RF and restricts its range to doorways and chokepoints in a building.

Each fob has an RF and Bluetooth transmitter while each portal beacon has an RF and Bluetooth receiver. The Bluetooth range on the portal beacons is significantly smaller than the RF range. The Bluetooth range primarily covers the doorway itself. Once the Bluetooth signal of a fob is within range of the portal beacon, the portal beacon sends the contents of the signal to the CCU. At the CCU, the ID of the fob is read and a table within the web server's database is updated with the ID's new most recent location and the time of detection. When an alarm is activated, this information is used to report location of the threat.

When the CCU receives the alarm activation signal, it divides the signal into the ID number and the requested alarm level. Using the ID number, the web server accesses its database and identifies which Portal Beacon most recently detected the ID number. The location of this portal beacon is considered the location where the alarm was activated. Next, if the alarm is a level 1 (subtle) alarm, the CCU sends out an email alert to all faculty explaining that there is a gun threat and including the location of the threat. If the alarm is a level 2 (blaring) alarm, the CCU activates the school's alarm and sends an email as previously explained.

3.1 [FOB](#)

The fobs consist of two buttons, an HC-05 Bluetooth module, 315MHz RF transmitter, Arduino Nano, and a 5V rechargeable battery pack. The full prototype is 6in x 3in, roughly the size of a phone. The alarm activation portion of the fob includes two buttons, one to activate each level of alarm, and the RF transmitter to send the alarm signal. The location portion of the alarm includes

an HC-05. Before entering the HC-05's RX pin, the voltage must be converted to 3.3V from 5V [7]. The original design for the fob featured an Arduino Pro Mini instead of an Arduino Nano due to the Mini's smaller size. We lost several weeks to debugging the three Pro Minis we had and finally determined that all three were faulty. As a result, we changed the design to Arduino Nanos.

3.2 [Portal Beacon](#)

The portal beacons, unlike the fobs, will not be mobile and will instead be attached at doorways and choke points throughout the school building. They will each consist of an HC-05 Bluetooth module, 315MHz RF receiver, Arduino Nano, and Ethernet cable. The Arduino serves as the microcontroller of the portal beacon, while the Bluetooth and RF receivers receive different signals from the fobs. We included a temporary LED connected to port 5 for testing purposes. Also included on the portal beacon is a RobotDyn - W5500 Nano V3 Ethernet Network Shield. This ethernet module allows the portal beacon to have internet capabilities to send incoming data to the online database.

3.3 [CCU](#)

The CCU contains an Apache web server that is run through XAMPP which is an open-source cross platform web server. The programming language that is used for the databases is MySQL. Set up and usage of the CCU is explained step by step in section 6.1 of the appendix.

Design Evaluation

This section documents how we tested parts of our system to determine if they meet our requirements. We also discuss the outcomes of these tests and what we did to fix failures.

Testing of the Connectivity Between Fob and Portal (Bluetooth & RF Signals)

For our prototype to be able to determine the location of an alarm with room-level accuracy and send alarm data, the portal beacons and fobs first need to be able to communicate. For the location system, this communication occurs over Bluetooth and for the alarm system this communication occurs over RF. As a result, our first test was to determine that our portal beacon and fob could connect over both signal types.

- **Objectives:**

- a. Verify that the fob and portal beacon can connect over Bluetooth
- b. Verify that the fob and portal beacon can connect over RF

- **Features Evaluated:**

The main features evaluated in this test is the connectivity between the Bluetooth and RF modules located on the fob and portal beacons. For RF, communication will occur between the transmitter on the fob and the receiver on the portal beacon. For Bluetooth, communication will occur between an HC-05 module set to master mode on the fob and an HC-05 module set to servant on the portal beacon.

- **Scope:**

This test consists of creating a serial connection between two HC-05's and two RF modules. Libraries will need to be obtained for both Bluetooth and RF serial communication.

- **Test Plan:**

For the Bluetooth connection, the modules come built in with LED's that will light up in a certain pattern when the modules make a successful connection. We will need to power the modules properly and wait for this light pattern to determine if they have connected. For the RF communication, we will add two buttons to the fob that will cause a signal to be sent over RF when pressed. On the portal beacon, we will attach an LED temporarily that will be programmed to light up when either of the buttons on the fob are pressed.

- **Acceptance Criteria:**

The Bluetooth test will be successful if the LEDs on both HC-05 modules begin to blink in tandem. The RF test will be considered successful if the LED lights up signaling a proper serial connection.

- **Test Results and Evaluation:**

The Bluetooth tests were successful. When the modules were powered on, they initially blinked rapidly, but after several seconds their LEDs went dark briefly before beginning to blink out a pattern in tandem that consisted of a long pause followed by two rapid blinks. This indicated that they had connected.

The RF test was proven to be successful because the LED on the portal beacon circuit lit up when one of the buttons on the fob Arduino was pressed, indicating that the command to light

up the LED was being sent via RF signal. Fig. 3 shows the LED lighting up when either button is pressed.

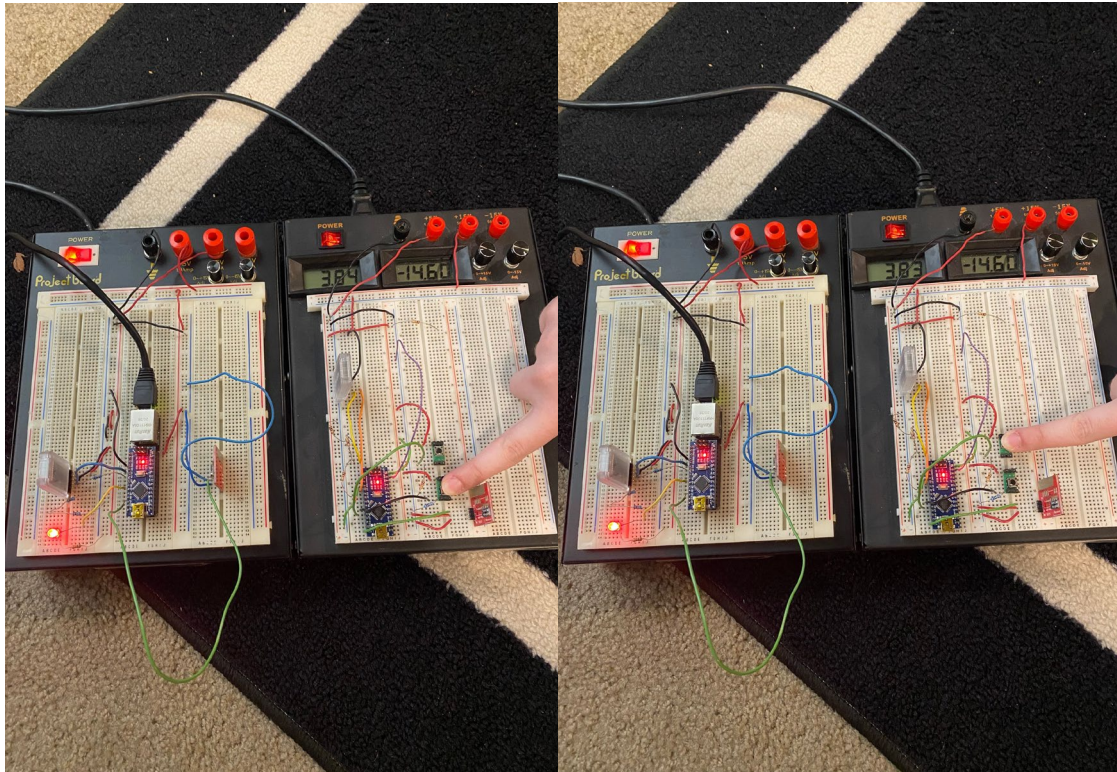


Figure 3. Shows LED lighting up in 2 different cases

Bluetooth Range Testing

Our security system is required to be able to determine location of alarm activation with room-level accuracy. During the design phase, we decided to accomplish this by putting portal beacons in doorways and chokepoints and that the range of these beacons should cover the area of the doorway (about 8ft). This test will determine what the Bluetooth range is and will involve adjusting the power to the Bluetooth transmitter in order to achieve our 8ft signal range goal.

- **Objectives:**
 - a. Verify that the Bluetooth range is 8ft.
 - b. Verify that the Bluetooth modules pair within a second of entering the 8ft range.
- **Features Evaluated:**

This test will evaluate the HC-05 Bluetooth transmitter on the fob and HC-05 Bluetooth receiver on the portal beacon.

- **Scope:**

This test will be completed at a variety of ranges, from where there's over 8 feet between the HC-05 modules to where there is only a couple of inches between them. During the testing, we will also be adjusting the power flow to the HC-05 modules to decrease the range, and thus power usage, of the Bluetooth modules so that the range is as close as possible to the 8 ft we have decided upon. The test will also determine if the portal beacons and fobs pair in a timely manner. In order to do this, the amount of time it takes for the HC-05 modules to pair will be recorded at each distance.

- **Test Plan:**

This test will require the breadboard-based prototypes of a fob and portal beacons we have constructed that have HC-05 Bluetooth modules and Arduino pro minis. We will need a measuring stick to measure distance and a stopwatch to measure pairing time. We will also need a multimeter to aid in adjusting the power flow to the HC-05 modules. The test will involve attempting to pair the HC-05 modules beginning with the modules placed besides each other. We will then increase the range to 8ft and attempt to pair the modules. If the range of the Bluetooth is too small, we will increase the voltage going to the HC-05 modules, and if the range is too large, we will decrease it. In the case of the range being too large, we will adjust the placement of the fob breadboard to be 12ft away from the portal beacon and will adjust the range of the Bluetooth using this positioning until the range drops below 12ft. We will then return to the 8ft separation and ensure that the range is still larger than 8ft.

- **Acceptance Criteria:**

We will consider this test successful if the fob and portal beacon Bluetooth modules pair within a second of being in range of each other, and if the range of the portal beacons is not less than 8ft or more than 12ft.

- **Test Results and Evaluation:**

When the Bluetooth modules were paired, they went from blinking constantly to blinking in a pattern of one long pause followed by two rapid blinks. This made observing their pairing easy. During the range testing, the Bluetooth modules successfully paired while 8, 12, 20, 30, and 35 feet apart. From 20ft onwards, the connection was inconsistent, and we only got the devices to connect once with 35 feet of separation. With a wall between them, the modules could connect from 20ft apart but did not connect successfully any farther than that.

This 20ft Bluetooth range was almost twice as large as our intended range, so we adjusted the voltage to the Bluetooth module. We initially tried 3.3V on one device and observed no change in the pairing range. The voltage requirement of the Bluetooth modules is 3.3V to 6V, so decreasing the voltage beyond 3.3V caused the devices to switch off. We concluded that adjusting the range of the Bluetooth module would require adjusting the circuit on the module itself. However, a 20ft range can still be used to achieve room-level accuracy for the security system if the portal beacon is placed inside the room instead of in the doorway. Our usage plans will change to reflect this.

Our time of connection test for the Bluetooth resulted in connection times of less than 5 seconds up to the 20ft range. This fulfills our connection time requirement, which was 5 seconds. We used the double blink from the connection light pattern to mark when to stop the stopwatch, but since this double blink is the end of the connection pattern, the connection time is likely even less than 5 seconds.

With the noted change in usage plan, our tests were a success.

Table 1. Time Until Bluetooth Connection at a Range of Distances

Distance	Time Until Connection (first time lights blink 2 times rapidly)			
	Trial 1 [s]	Trial 2 [s]	Trial 3 [s]	Average [s]
0ft	4.65	4.74	5.45	4.95
10ft	4.80	4.76	4.87	4.81
20ft	7.99	6.39	11.30	8.56
25ft	5.43	7.41	6.02	6.29

RF Range Testing

In order to be an effective security system, our system needs to be able to receive emergency signals from anywhere in the school building. As a result, our RF signal, which handles the alarm activation, needs to be large enough to ensure that a portal beacon is always within signal range. Depending on the signal length, portal beacons would need to be scattered throughout the

school accordingly, however we decided that an RF signal would only be acceptable if it were greater than 20ft. For this test, we will determine the maximum RF signal range between the fob and portal beacon to ensure that the range is greater than 20ft.

- **Objectives:**

- a. Verify that the Portal Beacon can receive information over RF from the fob
- b. Verify that the portal beacon can detect the fob RF signal from a range of greater than 20ft away
- c. Verify that the RF signal can be detected through walls
- d. Measure the total range of the RF signal

- **Features Evaluated:**

This test will evaluate the 315MHz RF transmitter on the fob and 315MHz RF receiver on the portal beacon as well as the two buttons on the fob that are used to activate the transmitter.

- **Scope:**

This test will be completed at a variety of ranges, between 0 and 20+ feet. If the range of the RF signal is larger than 20 ft, we will continue testing until we determine its full range. If the range is less than 20 ft, we will adjust the power flow to the RF modules to increase the range, and thus power usage, of the RF modules. The test will also determine if the buttons can activate the RF signal when pressed. Finally, the effect of wall interference will be tested by separating the RF transmitter and receiver with several layers of walls and reattempting the previous tests.

- **Test Plan:**

This test will require breadboard prototypes for the fob and portal beacon. These prototypes use an Arduino Pro Minis and 315MHz RF transmitter and receiver. We will need a measuring stick to measure distance. We will also need a multimeter to aid in adjusting the power flow to the HC-05 modules. This test should be attempted twice, once in a large open area and once in a building with multiple rooms. This will allow us to see how wall interference affects the range of the RF device. In both cases, the test will begin with the fob and portal beacon close together. If the LED's light up, indicating that the RF signal has been received, we will move the devices further apart and try again. This will continue until the devices no longer pair and at that point the distance between them will be measured.

- **Acceptance Criteria:**

This test will be considered successful if the range of the RF transmission is at least 20ft even when walls are in the way. If this criteria is not met, the 5V input to the RF transmitter will be increased and the test will be attempted again.

- **Test Results and Evaluation:**

Using an LED, we were able to show that the RF was transmitting between Arduinos by pressing one of two buttons and seeing that the LED lit up. During range testing, the LED lit up in response to a button press from a maximum distance of 30ft away. This range was much smaller than we expected based on the product description of the RF transmitter and receiver. It is still a larger value than our minimum 20ft however, so the range test can be considered a success.

The time of connection tests resulted in values much smaller than the Bluetooth connection time, which is ideal for an emergency signal. Table 2 below shows the time between pressing the button and the LED lighting up, however since the values were so small, they are subject to error from the speed of human reflexes. The average time until connection varies only slightly as the distance between modules increases. The largest time gap, which occurred between 0 and 10ft, was only 0.2s.

Table 2. Time Until Bluetooth Connection at a Range of Distances

Distance	Time Until Connection (first time lights blink 2 times rapidly)					
	Trial 1 [s]	Trial 2 [s]	Trial 3 [s]	Trial 4 [s]	Trial 5 [s]	Average [s]
0ft	1.13	0.62	0.64	0.56	0.71	0.73
10ft	0.81	0.78	1.12	0.94	1.00	0.93
20ft	0.92	0.68	1.22	1.03	0.97	0.96

[Sending Messages Over RF](#)

In order to fulfill our system’s requirement of activating different alarm levels and determining alarm activation location, our fob needs to send both ID and alarm level information over an RF signal. In this test we sent both ID and alarm level data over RF and used the Arduino serial monitors in order to determine if our system fulfills these requirements.

- **Objectives:**

- a. Verify that messages can be sent over RF
- b. Verify that messages can be received over RF
- c. Verify that the message can be divided into two integers (ID number and alarm level) in order to trigger a response

- **Features Evaluated:**

In this test we evaluated the way messages are sent with our chosen RF library (VirtualWire.h) to determine if the ID numbers and alarm level can be passed successfully between fobs and portal beacons.

- **Scope:**

This test utilizes the VirtualWire.h library to send messages between a 315 MHz RF transmitter and receiver each connected to an Arduino Nano. We utilized the serial monitor on both devices in order to determine the success of the test. We also attached two buttons to the Arduino on the fob in order to generate two different RF signals.

- **Test Plan:**

Our testing will begin on the fob end. We will use the Serial monitor in order to ensure that our buttons can generate two different number sequences based on which button is being pressed. The sequences will consist of the fob's assigned ID (which was initially 1234), which will be multiplied by 10 and have a digit added to it that corresponds to the alarm level. So, when button 1 is pressed, the number 12341 should display on the serial monitor. If button 2 is pressed, 12342 will display.

Once we have determined that the signal is being generated properly, we will turn to focus on the portal beacon and its serial monitor. We will first print the entire signal that the portal beacon receives when a button is pressed on the fob to ensure that it is either 12341 or 12342. After that, we'll test if we can successfully divide the signal back into the fob ID and alarm level and print these values on the monitor.

- **Acceptance Criteria:**

In order to be a successful test, the portal beacon needs to be able to receive both the fob ID and alarm level whenever a button on the fob is pressed.

- **Test Results and Evaluation:**

We were able to successfully print 12341 and 12342 on the fob's serial monitor, however the portal beacon was printing the values 62 and 63 instead. We determined that this was because VirtualWire.h can only send 8-bit messages between RF modules. By shortening the sent values

to 121 and 122, the portal beacon was able to successfully receive and print these values as well as store them under the variables *alarmlvl* and *ID_rf*. The 8-bit limitation greatly reduces the number of fobs that can be incorporated into this system, making it inconvenient for schools with over 99 staff members. This indicates that VirtualWire.h would not be an ideal library for a complete security system, but it still works for a proof of concept consisting of one fob.

Despite the discovered limit to the number of fobs that can be incorporated into our system, our prototype can still send ID numbers and alarm levels over RF, so this test can be considered a success.

Sending Messages Over Bluetooth

For our system to determine alarm activation location, we must be able to send fob ID information over Bluetooth indicating when fobs are detected within range of a portal beacon. In this test we will determine if our prototype can fulfill this requirement.

- **Objectives:**
 - a. Verify that messages can be sent over Bluetooth
 - b. Verify that messages can be received over Bluetooth
- **Features Evaluated:**

In this test we evaluated the way messages are sent with our chosen Bluetooth library (SoftwareSerial.h) to determine if the ID number can be passed successfully between fobs and portal beacons.

- **Scope:**

This test utilizes the SoftwareSerial.h library to send messages between an HC-05 master module and HC-05 servant module each connected to an Arduino Nano. We utilized the serial monitor on both devices in order to determine the success of the test.

- **Test Plan:**

Like in the RF testing, the serial monitors of the Arduinos were used to determine if the Bluetooth modules can successfully send a signal consisting of a variable. This variable is the ID number.

- **Acceptance Criteria:**

If the ID number of the fob can be printed on the serial monitor of the portal beacon, we will consider this test a success.

- **Test Results and Evaluation:**

The test initially failed, and the portal beacon only received values of -1. The number we were actually sending was 12, so we concluded that the message was being scrambled sometime in the Bluetooth communication process. Due to time constraints we were not able to resolve this communication issue. Our next course of action would be to return to the instructions about the SoftwareSerial.h library and try to find what we might be missing in our current code.

As it is currently, our prototype does not fulfill the location detection requirement since that was covered by the Bluetooth message sending and the Bluetooth message test was a failure.

Updating Database Table

Though our previous tests examined our prototype's ability to receive information about fob locations and alarm levels, this information still needs to be sent to the CCU in order to be processed and used. The CCU's role is to activate the alarm response and begin the staff notification protocols, so the project's ability to perform as a successful alarm system relies on the CCU. The following test will determine whether the Raspberry Pi (the CCU itself) can receive information from the portal beacon and update a database accordingly.

- **Objectives:**

- a. Verify that the location table updates the location ID in the correct fob ID row.
- b. Verify that the time updated is recorded in the database.

- **Features Evaluated:**

The test will evaluate the Raspberry Pi database and observe how it receives information from the Portal Beacon.

- **Scope:**

This test will evaluate the project's ability to store the location, identification credentials, date and time into an online database that has been created on the CCU. This test will also be completed alongside the connectivity between Arduino mini and HC-05 because the Arduino sketch created on the Arduino will also be sending information to the .js file on the CCU.

- **Test Plan:**

The first step of this test will consist of the creation of a database that will hold the information for ID number, date and time of entry, and the type of alarm that has been triggered. Once the database is created, a serial connection between the CCU and the portal beacon must be

established. This will consist of creating a php script to perform the updating action to the database.

- **Acceptance Criteria:**

This test will be considered successful if the database can update information as actions are being inputted to the portal beacon.

- **Test Results and Evaluation:**

Our initial test plan for updating the database table within the CCU proved to be inconclusive due to the Raspberry Pi's (RPI) inability to maintain a static IP. This caused fatal error messages whenever the .php script was run. After doing more research it was found that for the purposes of our project and testing objectives it makes more sense to host the database on a free 3rd party hosting server that will have all the same capabilities as the RPI. The new database that we created is run on a cross-platform server, XAMPP, that contains Apache and MariaDB packages. Since we had a solid background in creating tables from our work with the RPI, making the switch to the XAMPP server was seamless.

The next portion of the test consisted of creating the Arduino sketch that will push the Bluetooth ID and alarm level to the database. To accomplish this, the portal beacon must be able to connect to the internet. For this our group chose to use a RobotDyn - W5500 Nano V3 Ethernet Network Shield that is compatible with our Arduino nano.

Due to the change in components of the CCU, we decided to switch to a more modular form of testing that involves first, testing the ability of the new server to connect to the created database and second, testing the updating of values from our fob. To accomplish this our desired variables were changed in the php script to match the ones on the portal beacon. Once these changes were made our prototype was able to update the database corresponding to the button presses on the fob. This test proved that our tables can receive data from the portal beacon. Fig. 4 shows the outcome of the tests, displaying the server, database, and database table in the maroon box and the values retrieved from the portal beacon in the red box.

127.0.0.1 / 127.0.0.1 / test / data | phpMyAdmin 5.0.3

127.0.0.1/phpmyadmin/sql.php?server=1&db=test&table=data&pos=0

Server: 127.0.0.1 » Database: test » Table: data

Showing rows 0 - 6 (7 total, Query took 0.0005 seconds.)

```
SELECT * FROM `data`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: All | Filter rows: Search this table | Sort by key: None

+ Options							
				id	ID_rf	alarmlvl	Time
<input type="checkbox"/>	Edit	Copy	Delete	2	12	2	2021-05-07 14:10:00
<input type="checkbox"/>	Edit	Copy	Delete	3	12	2	2021-05-07 14:10:03
<input type="checkbox"/>	Edit	Copy	Delete	4	12	1	2021-05-07 14:10:05
<input type="checkbox"/>	Edit	Copy	Delete	5	12	2	2021-05-07 14:10:07
<input type="checkbox"/>	Edit	Copy	Delete	6	12	1	2021-05-07 14:10:09
<input type="checkbox"/>	Edit	Copy	Delete	7	12	2	2021-05-07 14:10:11
<input type="checkbox"/>	Edit	Copy	Delete	8	12	1	2021-05-07 14:10:14

Show all | Number of rows: All | Filter rows: Search this table | Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Label: Let every user access this bookmark

Bookmark this SQL query

Figure 4. Updated database values

Test Result Analysis & Future Steps

Through our tests we determined the following:

- Our fobs and portal beacons can successfully connect over RF and Bluetooth.
- The maximum signal range for Bluetooth is 20ft.
- The maximum signal range for RF is 30ft.
- We can send ID number and alarm level information to the portal beacons using RF signals.
- We cannot currently send usable data to the portal beacon using Bluetooth signals.
- Our CCU can receive data from the portal beacon and update a table with it.

From these results we know that our system fulfills one of our requirements by being able to correctly recognize two levels of alarm. We also have shown that our database can be updated with real-time information originally sent from the fobs. However, our system also needs to be able to determine where the alarm was activated from with room level accuracy. In order to do this, we planned to have fobs send their ID values via Bluetooth nonstop. Whenever a portal beacon detects an ID, it would then update a database with both the fob ID and the ID of the portal beacon that detected it. Then, when an RF-based alarm signal is sent, the CCU could take the ID of the fob that sent the emergency signal, cross reference database, and determine which portal beacon most recently detected said fob. That portal beacon's room assignment would be considered the location of the alarm. Despite our tests determining that both the RF and database updating are functioning, our prototype fails to meet all its requirements because the Bluetooth signals are currently bugged.

We believe the Bluetooth issue is caused by the fact that Arduino Nanos only have one serial output port. Since we're sending both Bluetooth and RF signals from the fob, these signals are interfering and resulting in incorrect values being sent over Bluetooth. As a result, one possible fix would be finding a different processor for our fob.

One requirement that we have not been able to test yet is our battery requirement. Since we went with a design where fobs are used to activate the alarm, these fobs need to either have batteries that last a whole semester or have rechargeable batteries that can last 12 hours. In our current design, we're using a rechargeable phone battery that takes 5 hours to recharge. This means that it can be successfully recharged overnight and fulfills part of our requirement, but we do not currently know how long the battery will last when hooked up to our circuit and run for a whole day.

5. Conclusions

The K-12 security system was designed to improve the response of school to active and potential shooter situations by being a handheld alarm activation system with the ability to determine the threat's location and send two different levels of alarm. The team was limited by only having two group members on campus due to the pandemic, faulty Arduinos, and the winter storm causing delivery delays. These delays and limitations resulted in our inability to fix a bug with our Bluetooth connection that resulted in the incorrect messages being sent, preventing us from fully testing our prototype. However, our final prototype still features a XAMPP database, Raspberry Pi-based central computer unit, Arduino-based portal beacon, and handheld Arduino-based fob. By holding a button on the fob, the prototype can communicate the fob ID and button alarm value from the fob to the CCU where it is input into a database.

For future work, our group recommends testing the battery life of the fob's battery pack and fixing the Bluetooth message error. We would also recommend showing a proof of concept of the alarm notification system using Temboo, an email notification service that allows for emails to be sent from Arduino.

6. Appendices

6.1 Fob Usage Instructions

The fob can be used to trigger either a subtle alarm or a blaring alarm. Which alarm the staff member chooses to trigger is up to them, however we describe two possible situations below.

When to use a subtle alarm (button 1):

- If an individual claims to have a gun but has made no move yet to retrieve it

When to use a blaring alarm (button 2):

- If an individual is holding a gun
- If you hear a gunshot

In order to send out an alarm signal, hold down either button on the fob for more than 4 seconds before releasing.

Make sure to charge the fob overnight.

6.2 Database Usage Instructions

Step 1 Download XAMPP

Step 2 Ensure that the proper packages have been initialized as seen in Fig. 5

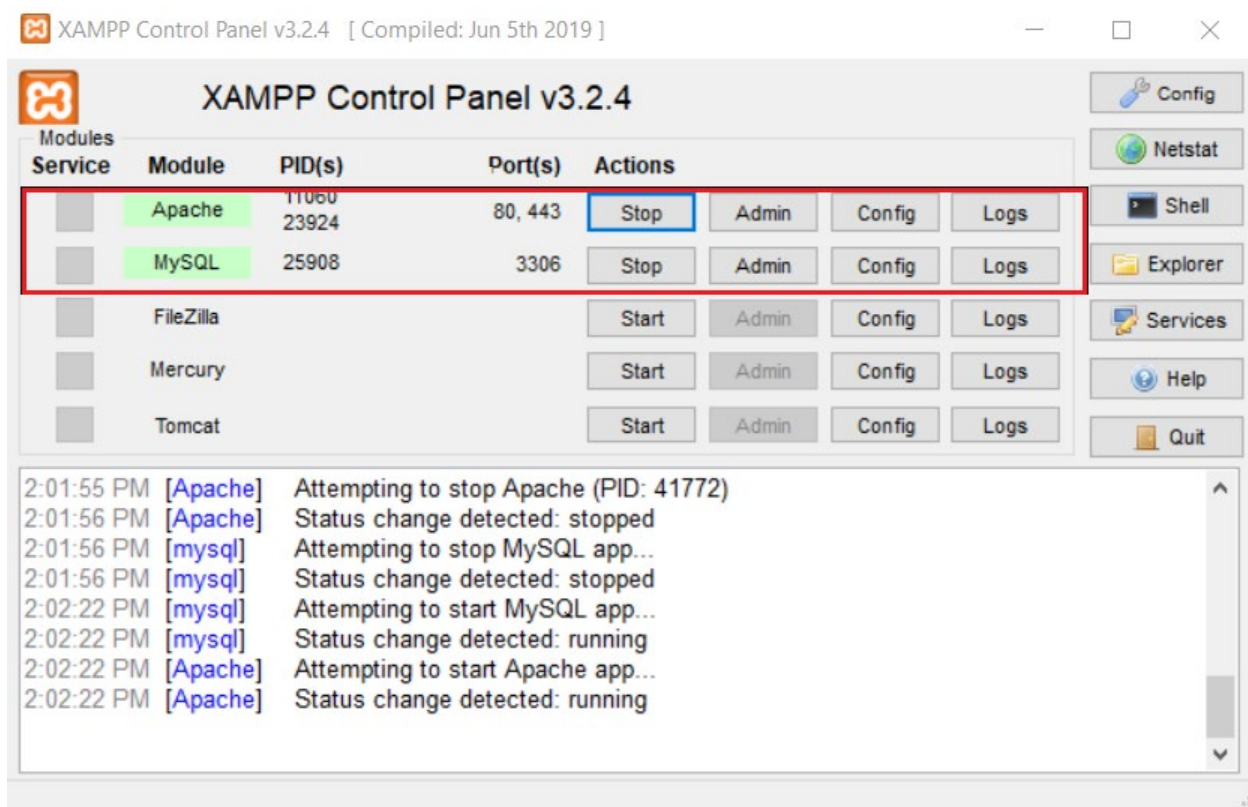


Figure 5. XAMPP control panel

Step 3 In your preferred browser type in your local IP address followed by /phpmyadmin to access the databases

Step 4 Create a new database and database table.

Step 5 Adjust the variables in the table to match the variables being occupied in the portal beacon

Step 6 Using the notepad create a new file named Write_data.php and add the php script at the end of this document to the file.

Step 7 Once the php script is completed you will need to save it to the htdocs file located in the xampp libraries in your file explorer

6.3 Fob and Portal Beacon Prototype Assembly Instructions

List of materials:

- Fob x1:
 - HC-05
 - 315 MHz RF transmitter
 - Arduino nano
 - Wires
 - 5x 1k resistors
 - 2x push button (normally open)
 - Half breadboard
 - Miady 10000mAh Dual USB Portable Charger
 - Portal beacon x1:
 - HC-05
 - 315 MHz RF receiver
 - Arduino nano
 - Wires
 - 3x 1k resistors
 - Powered breadboard
 - RobotDyn - W5500 Nano V3 Ethernet Network Shield
1. Upload required libraries to the Arduino IDE
 - a. Libraries are SoftwareSerial.h, VirtualWire.h, Ethernet2.h, and SPI.h
 2. Configure the HC-05's
 - a. Power up one of the Arduino nanos and connect it to a computer
 - b. Open Arduino IDE and upload a blank sketch
 - c. Unplug off the Arduino from power
 - d. Connect Portal Beacon HC-05 to Arduino
 - i. TX to TX, RX to RX, GND to GND, 5V to 5V
 - e. Power the Arduino on while holding the reset button on the HC-05
 - i. This should put the HC-05 into AT command mode (marked by the HC-05's long, slow blinking)
 - f. Enter the serial monitor on Arduino IDE and make sure the baud rate is 38400 and "both NL and CR" is selected.
 - g. HC-05 modules begin in servant mode, which is the mode the Portal Beacon should be in
 - h. Get the address of the servant (portal beacon) module
 - i. Type "AT" into the serial monitor until it returns with "OK" (first time may output an error instead).

- ii. Type “AT+ROLE?” to see the module’s role (0 is servant and 1 is master). For the portal beacon fob, should be 0.
 - iii. Type “AT+ADDR?” to see the module’s address. Copy it down and change colons to commas. This address will be given to the master HC-05 module to ensure that it will only pair with recognized HC-05 servant modules.
- i. Configure the master (fob) module
 - i. Open serial monitor and type “AT+ROLE=1”
 - ii. Type “AT+ROLE?” to confirm the new role of the HC-05
 - iii. Type “AT+CMODE?” to check the connection mode
 1. CMODE = 1 is connect to any
 2. CMODE = 0 is connect to one
 - iv. Type “AT+CMODE=0” to set connection mode to one
 - v. Type “AT+BIND=” and fill in the portion after the equal sign with the address of the servant module
3. Construct the circuits as shown in Fig. 6 and Fig. 7 below keeping in mind that the master HC-05 is the transmitter and the servant HC-05 is the receiver.

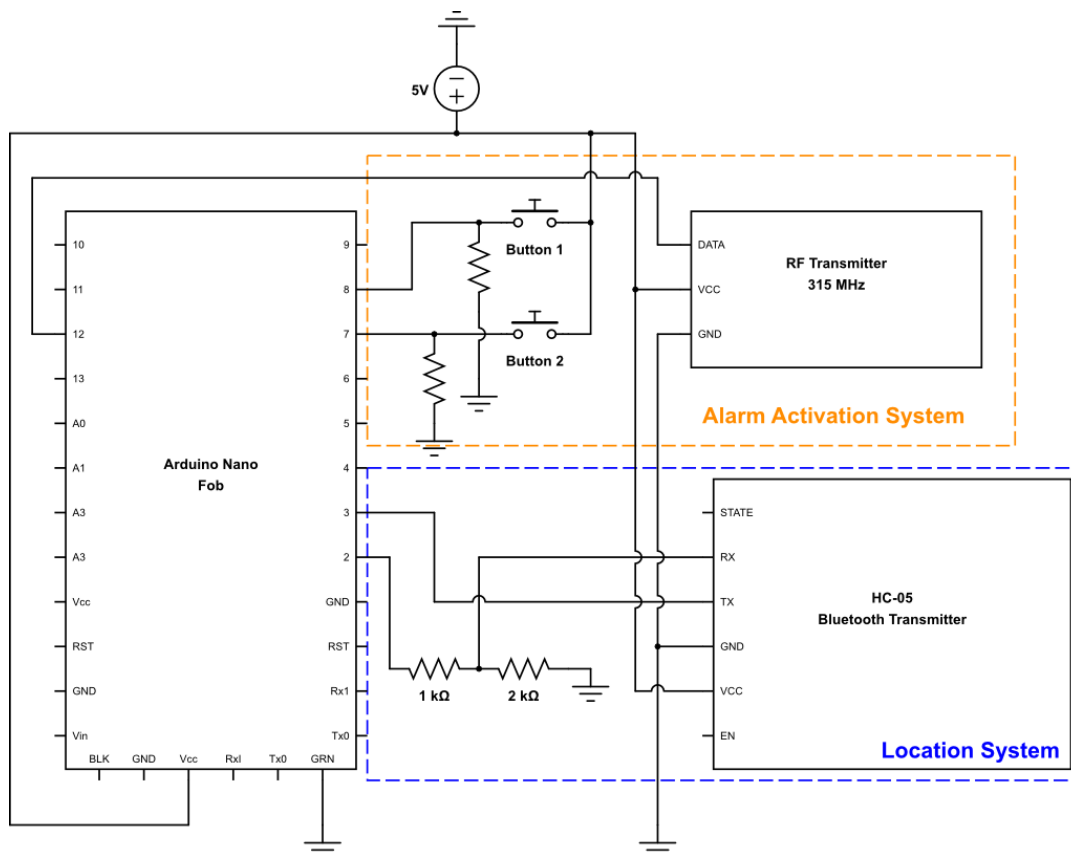


Figure 6. Circuit design of the fob

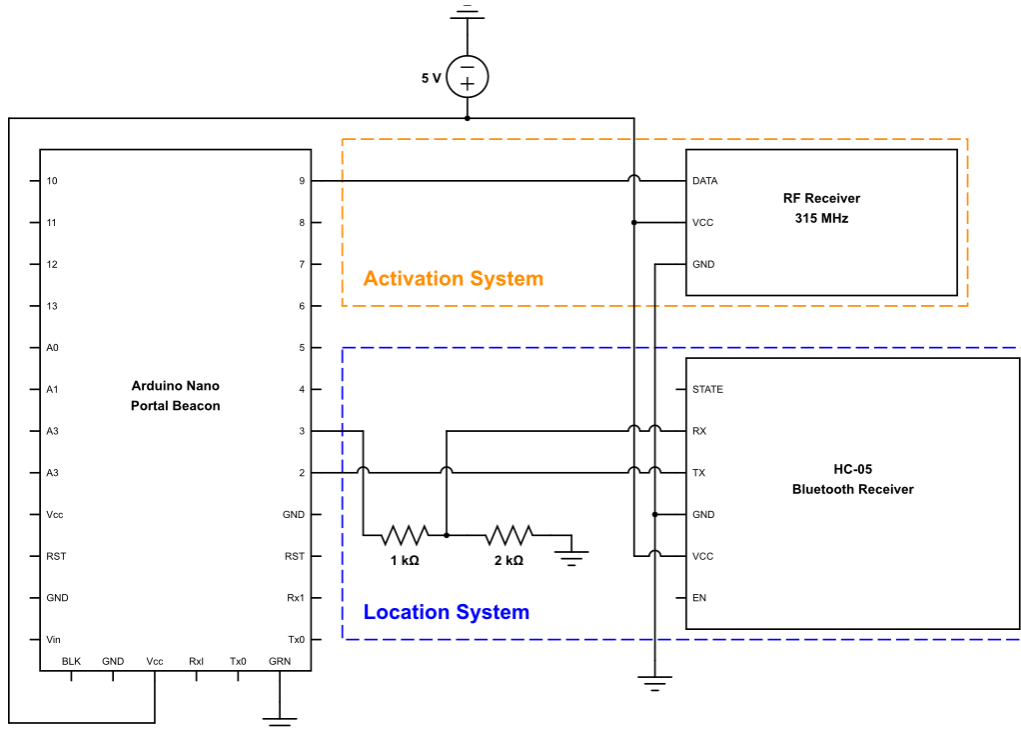


Figure 7. Circuit design of the portal beacon

4. With both HC-05's powered on, the modules should pair after several seconds and begin blinking in tandem
5. Upload the codes in Appendices 6.4 and 6.5 to the fob and portal beacon Arduinos respectively.
6. Connecting Arduino to the Internet
 - a. To connect arduino to the internet you will need to find the IP address of the Ethernet Shield and the computer that will hold the database.
 - i. The IP address of the Ethernet shield can be obtained through the DHCP address printer sketch that can be found in the example section of arduino.
 - ii. The IPv4 address can be found in your computer's hardware settings

```
DhcpAddressPrinter$
#include <Ethernet2.h>
byte mac[] = {
  0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02
};

EthernetClient client;

void setup() {
  Serial.begin(9600);
  // this check is only needed on the Leonardo:
  while (!Serial) {
    ;
  }

  // start the Ethernet connection:
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    for (;;)
      ;
  }
  // print your local IP address:
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    // print the value of each byte of the IP address:
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }
  Serial.println();
}
```

Figure 8. Test code used to obtain Ethernet Shields IP address.

6.4 Arduino Fob Code

```
// Fob
// Make sure the VirtualWire Library is downloaded and installed into Arduino
Libraries
// Has the Servant Bluetooth module and RF Transmitter

#include <SoftwareSerial.h> //For Bluetooth
#include <VirtualWire.h> //For RF

SoftwareSerial BTserial(2, 3); // RX | TX

// *****IDENTIFICATION #*****
int ID = 12;

int TransmitData = ID*10;
uint8_t TransmitDataLen = sizeof(TransmitData);

int button1 = 8;
int button2 = 7;
const int long_press_time = 4000; //4 seconds

//for extended button hold
int lastState_1 = HIGH; // the previous state from the input pin
int currentState_1; // the current reading from the input pin
unsigned long pressedTime_1 = 0;
unsigned long releasedTime_1 = 0;
int lastState_2 = LOW; // the previous state from the input pin
int currentState_2; // the current reading from the input pin
unsigned long pressedTime_2 = 0;
unsigned long releasedTime_2 = 0;

void setup()
{
// *****FOR BLUETOOTH*****
// start th serial communication with the host computer
Serial.begin(9600);
Serial.println("Arduino with HC-05 is ready");

// start communication with the HC-05 using 38400
BTserial.begin(38400);
Serial.println("BTserial started at 38400");

// *****FOR RF*****
//RF setup
vw_set_ptt_inverted(true); // Required for RF Link modules
vw_setup(300); // set data speed
```

```

    vw_set_tx_pin(12);          // RF data pin is connected here
    pinMode(button1, INPUT);
    pinMode(button2, INPUT);
}

void loop()
{
// *****FOR BLUETOOTH*****
    // Keep reading from Arduino Serial Monitor and send to HC-05
    if (Serial.available())
    {
        //Send ID into to Bluetooth modulee
        Serial.write(ID);
        BTserial.write(ID);
    }

// *****FOR RF*****

currentState_1 = digitalRead(button1);

    if(lastState_1 == LOW && currentState_1 == HIGH) {        // button is pressed
        pressedTime_1 = millis(); //save time when pressed
    }
    else if(lastState_1 == HIGH && currentState_1 == LOW) { // button is released
        releasedTime_1 = millis(); //save time when released

        long pressDuration_1 = releasedTime_1 - pressedTime_1;

        if( pressDuration_1 > long_press_time ) {
            TransmitData = TransmitData + 1;
            Serial.println(TransmitData);

            vw_send((uint8_t *)&TransmitData, &TransmitDataLen); // send the data out
to the world
            Serial.println(vw_send((uint8_t *)&TransmitData, &TransmitDataLen));
            TransmitData = TransmitData - 1;
            vw_wait_tx(); // wait a moment
            delay(200);
        }
    }

currentState_2 = digitalRead(button2);

    if(lastState_2 == LOW && currentState_2 == HIGH) {        // button is pressed

```

```

    pressedTime_2 = millis(); //save time when pressed
} else if(lastState_2 == HIGH && currentState_2 == LOW) { // button is released

    releasedTime_2 = millis(); //save time when released

    long pressDuration_2 = releasedTime_2 - pressedTime_2;

    if( pressDuration_2 > long_press_time ) {
        TransmitData = TransmitData + 2;
        Serial.println(TransmitData);

        vw_send((uint8_t *)&TransmitData, &TransmitDataLen); // send the data out
to the world
        Serial.println(vw_send((uint8_t *)&TransmitData, &TransmitDataLen));
        TransmitData = TransmitData - 2;
        vw_wait_tx(); // wait a moment
        delay(200);
    }
}

// save the the last state
lastState_1 = currentState_1;

// save the the last state
lastState_2 = currentState_2;

}

```

6.5 Arduino Portal Beacon Code

```
// Portal Beacon
// Installed Libraries: Virtualwire
// Has the Master Bluetooth module and RF Receiver

#include <SPI.h>
#include <Ethernet2.h>
#include <SoftwareSerial.h>
#include <VirtualWire.h>

SoftwareSerial BTSerial(2, 3); // RX | TX
//EthernetClient client;
IPAddress ip(192,168,1,210);

int ReceivedData;
uint8_t ReceivedDataLen = sizeof(ReceivedData);

int ledpin = 5;

int ID_bt;           // ID of the fobs detected by the PB through
Bluetooth
int alarmlvl;       // Level of triggered alarm
int ID_rf;          // ID of the fobs detected by the PB through
RF
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {192, 168, 1, 210 }; //Enter the IP of ethernet shield
byte serv[] = {192, 168, 1, 130} ; //Enter the IPv4 address

char server[] = "192.168.1.130";

void setup()
{
// *****FOR BLUETOOTH*****

  Serial.println("Arduino with HC-06 is ready");

  // HC-06 default baud rate is 9600
  BTSerial.begin(9600);
  Serial.println("BTserial started at 9600");

// *****FOR RF*****
  ww_set_ptt_inverted(true); // Required for RF link modules
  ww_setup(300);
  ww_set_rx_pin(9);
  ww_rx_start();
```

```

pinMode(ledpin, OUTPUT);

// *****FOR ETHERNET*****
Serial.begin(9600);
if (Ethernet.begin(mac) == 0) {
  Serial.println("Failed to configure Ethernet using DHCP");
  // no point in carrying on, so do nothing forevermore:
  // try to configure using IP address instead of DHCP:
  Ethernet.begin(mac, ip);
}
Ethernet.begin(mac, ip);
}

void loop()
{
// *****FOR BLUETOOTH*****
  // Keep reading from HC-06 and send to Arduino Serial Monitor

  if (BTSerial.available())
    Serial.write(BTSerial.read());
    ID_bt = BTSerial.read(); //set ID to the value being sent by the Bluetooth
    Servant

// *****FOR RF*****
  if (vw_get_message((uint8_t *)&ReceivedData, &ReceivedDataLen)) //message is
  received as an array
  {
    alarmlvl = ReceivedData % 10;
    ID_rf = (ReceivedData - alarmlvl)/10;
    Serial.println(alarmlvl);

  }
// *****FOR ETHERNET*****

if (client.connect(server, 80)) {
  Serial.println("connected");
  client.print("GET /Write_dataWork.php?");
  client.print("ID_bt=");
  client.print(ID_bt);
  client.print("ID_rf=");
  client.print(ID_rf);
  client.print("alarmlvl=");

```

```
client.print(alarmlvl);

client.println(" HTTP/1.1"); // Part of the GET request
client.println("Host: 192.1.168.130"); // Computer Ip
client.println("Connection: close");
client.println(); // Empty line
client.println(); // Empty line
client.stop(); // Closing connection to server

}

else {
  // If Arduino can't connect to the server (your computer or web page)
  Serial.println("--> connection failed\n");

  delay(2000);
}
```

6.6 PHP Code (Write dataWork)

```
<?php
class data{
    public $link='';
    function __construct($ID_rf, $alarmlvl){
        $this->connect();
        $this->storeInDB($ID_rf, $alarmlvl);
    }

    function connect(){
        $this->link = mysqli_connect('localhost','root','') or die('Cannot connect to the DB');
        mysqli_select_db($this->link,'test') or die('Cannot select the DB');
    }

    function storeInDB($ID_rf, $alarmlvl){
        $query = "insert into data set ID_rf='".$ID_rf."', alarmlvl='".$alarmlvl.'";
        $result = mysqli_query($this->link,$query) or die('Errant query:  '.$query);
    }
}
if($_GET['ID_rf'] != '' and $_GET['alarmlvl'] != ''){
    $data=new data($_GET['ID_rf'],$_GET['alarmlvl']);
}

?>
```

6.7 Bill of Materials

Summary

	Cost of Items	Cost of Shipping	Total
Expenses	\$257.63	\$25.12	\$282.75
Current Available Balance	\$917.25		

	Expenses	Quantity	Cost of Item	Cost of Shipping	Total
Requisition Form #1	<i>Items total:</i>		\$35.00	\$0.00	\$35.00
	Raspberry Pi 4 Model B/2GB	1	\$35.00		\$35.00
Requisition Form #2	<i>Items total:</i>		\$25.88	\$0.00	\$25.88
	Micro HDMI to HDMI Cable	1	\$7.99		\$7.99
	32GB MicroSD Card	1	\$9.90		\$9.90
	MicroSD Card Reader	1	\$7.99		\$7.99
Requisition Form #3	<i>Items total:</i>		\$18.94	\$7.09	\$26.03
	Bluetooth Transmitter	1	\$8.99		\$8.99
	Arduino Pro Mini	1	\$9.95	\$7.09	\$17.04
Requisition Form #4	<i>Items total:</i>		\$80.72	\$7.09	\$87.81
	Bluetooth Transmitter	3	\$8.99		\$26.97
	Arduino Pro Mini	3	\$9.95	\$7.09	\$36.94
	RF Wireless Kit	2	\$11.95		\$23.90
Requisition Form #5	<i>Items total:</i>		\$10.96	\$10.94	\$21.90
	Raspberry Pi Case	1	\$5.00	\$7.95	\$12.95
	FTDI Programmer	1	\$5.96	\$2.99	\$8.95


Reimbursement Form #6	Items total:		\$11.00	\$0.00	\$11.00
	Real FTDI Programmer	1	\$11.00		\$11.00
Reimbursement Form #7	Items total:		\$30.98	\$0.00	\$30.98
	Nano board	1	\$15.99		\$15.99
	RobotDyn - W5500 Nano V3 Ethernet Network Shield	1	\$14.99		\$14.99
	Adapter for DIY USB Power Supply Breadboard Design	1	\$7.17		\$7.17
Requisition Form #8	Items total:		\$36.98	\$0.00	\$36.98
	Portable Battery Pack	1	\$25.99		\$25.99
	FOB Breadboard	1	\$10.99		\$10.99
Overall Totals	Items total:		\$257.63	\$25.12	\$282.75

Bibliography

- [1] Zimmerman, M., Carter, P. and Cunningham, R., 2019. *The Facts On Children And Teens By Guns*. [online] The Trace. Available at: <<https://www.thetrace.org/2019/08/children-teens-gun-deaths-data/>> [Accessed 19 November 2020].
- [2] K-12 School Shooting Database. 2020. *Charts & Graphs - K-12 School Shooting Database*. [online] Available at: <<https://www.chds.us/ssdb/charts-graphs>> [Accessed 20 November 2020].
- [3] *Americans with Disabilities Act of 1990, as Amended*. Available at: <<https://www.ada.gov/pubs/adastatute08.pdf>> [accessed Oct. 27, 2020].

- [4] “47 CFR § 15.231 - Periodic operation in the band 40.66-40.70 MHz and above 70 MHz.,”
 Legal Information Institute. [Online]. Available:
<https://www.law.cornell.edu/cfr/text/47/15.231>. [Accessed: 04-Dec-2020].
- [5] M. Brain, “How Remote Entry Works,” *HowStuffWorks*, Aug. 15, 2001.
 Available at: <<https://auto.howstuffworks.com/remote-entry.htm>> [accessed Nov. 16, 2020].
- [6] “Understanding Wireless Range Calculations,” *Electronic Design*, Apr. 03, 2013.
 Available at:
 <<https://www.electronicdesign.com/technologies/communications/article/21796484/understanding-wireless-range-calculations>> [accessed Nov. 16, 2020].
- [7] M. Currey, “Arduino to Arduino by Bluetooth,” *Martyn Currey*, Apr. 09, 2016.
 Available at: <<http://www.martyncurrey.com/arduino-to-arduino-by-bluetooth/>> [accessed Nov. 17, 2020].

Signatures

Signatures		
Project Name: Security Team		
The undersigned have reviewed and approved the final version of this document.		
	Date Received	Date Approved
Team Members: Timothy Raniszkeski Brittany Taylor Joshua Izi		5/7/2021
Team Advisor: 	5/7/2021	5/7/2021