Project Code: MQP SMJ AAIO

Campus Safety System

A Report
Submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degrees of Bachelor of Science by

James Beaulieu
Natasha Bonina
Lauren Lewis
Phyo Thinzar
Electrical and Computer Engineering
Major Qualifying Project

Advised by

Professor Susan Jarvis

Approved:

_____

May 1st, 2014

Abstract

Situated in an often dangerous neighborhood, Worcester Polytechnic Institute is disturbed by frequent crime, generating safety concerns among the campus population. This Major Qualifying Project designed a system to address this problem comprising of GPS-capable handheld devices, self-routing alarm boxes, and an intuitive interface for police. The system provides a one-button solution and mesh network infrastructure as a vital tool which transmits real-time location data from the victim to the police in emergency situations.

Acknowledgements

Professor Susan M. Jarvis
Professor Stephen J. Bitar
Robert M. Boisse
OSH Park, especially James “Laen” Neal & Dan Sheadel
Texas Instruments, especially Alexandria Davis
Technology Transfer Office, especially Todd S. Keiller
Amy Beth P. Laythe
WPI Campus Police, especially Sgt. Michael L. Jacobs & Off. Brian F. Lavallee
Melissa K. Rodas
Kenneth Flaherty
Shannon Cotter
Attie Grande

Authorship

All group members participated in data collection, project design, prototype development and the completion of this report. Throughout the project, each individual's responsibilities fluctuated and members often paired off in groups of two for work on major components of the project. An arrow symbolizes a transition of responsibility for the component.

Major Component	**Group members**
Interviews	All
MCU	James and Lauren
Xbee	Lauren and Natasha
GPS	Phyo and James
NFC	Natasha → James
Solar Panel	Phyo and Natasha
Code	James and Lauren
Dispenser Display	James
Alarm System	Natasha and Phyo
PCB Design	Lauren and James
Police User Interface	Natasha → James and Lauren
Power	Phyo
Casings	Natasha and Phyo

Executive Summary

This project focused on the problem of the safety and security of the Worcester Polytechnic Institute (WPI) campus. This group outlined the aspects of an ideal system and identified technology that would help to implement the system.

The Problem

Geographically speaking, WPI's campus as well as many off-campus apartments are located in one of the more dangerous neighborhoods in Worcester [1]. This poses a security problem to the WPI community, an issue which is made more apparent by the number of safety notifications from campus police. In an interview with campus police, it became apparent that the current system on campus is non-intuitive and archaic. This system uses a series of telephones as a way to contact campus police; however, the emergency button is not always apparent to the user. If the user is not near one of these phones, there is no simple way to alert campus police.

This creates a perception of vulnerability felt around the campus. Whether students only feel unsafe at night or just off campus, a little less than half of students surveyed reported feeling this way. Given the number of companies that release products aimed at this problem, it may be felt on other campuses across the country as well.

A Solution

This Major Qualifying Project sought to address the outlined problem with an easy to use handheld device for members of the WPI community. It is meant to aid them if caught in an emergency situation. This would have to be simple, robust, and it would need to have a low barrier of entry. It would also need to assist police in responding to emergency situations.

From these product requirements, a system was created which can signal police to the scene of a crime where a handheld device is activated very quickly. It does this by sending GPS coordinates from the handheld to the computer screen of the police dispatcher. This gives police an accurate location to investigate in real time. Additionally, a nearby alarm box should help discourage an assailant from continuing his/her attack. What sets this system apart, however, is that the users only need worry about the handheld when they feel unsafe and want the security in their hand. Otherwise, the handhelds are maintained and stored in on-campus receptacles. Rather than forcing the user to pay for use of the handhelds, through either an annual fee or one-time cost, the user can borrow a handheld for the duration of any occasion that they feel it necessary. During this time it is registered to their campus account and they simply return it later the next day, at their convenience. This gives users a simple system and provides police with location data to respond more quickly than if they had to respond to a phone call.

The Why

This system was designed to address the problem while filling in important gaps left by the competition. Rather than buying and maintaining a device which they are expected to always have on them, this project gives students and staff access to the system when they feel they want to use it. There is no need for a user to charge the handheld device; the receptacle takes care of that. There is no need to register the device with campus police, the system tracks

which student has what device in case of activation or misplacement. Additionally with this system, there is no wire installation to extend the reach of the network to a new location, as the alarm boxes are completely wireless. This allows for expansion beyond campus limits, to local businesses or between campuses, without the need for extensive wiring. Rather, all that is required is a small weatherproof box on the side of the building. The key benefits are modularity and ease of use.

In the state that it is in, the project lacks some of the robustness needed for a consumer level product of this type. However, the base features of the desired system exist in the current prototype and allow GPS coordinates of the handheld to be sent along a small network to a simple user interface. This demonstrates the concept is possible and even preferable over competing products that rely on smartphones or key chains the user owns. There are a number of features that are still necessary for a final consumer product, as well as extensive stress testing to assure reliability. However, this system has the potential to assist campus police in reducing campus crime, here at WPI and elsewhere.

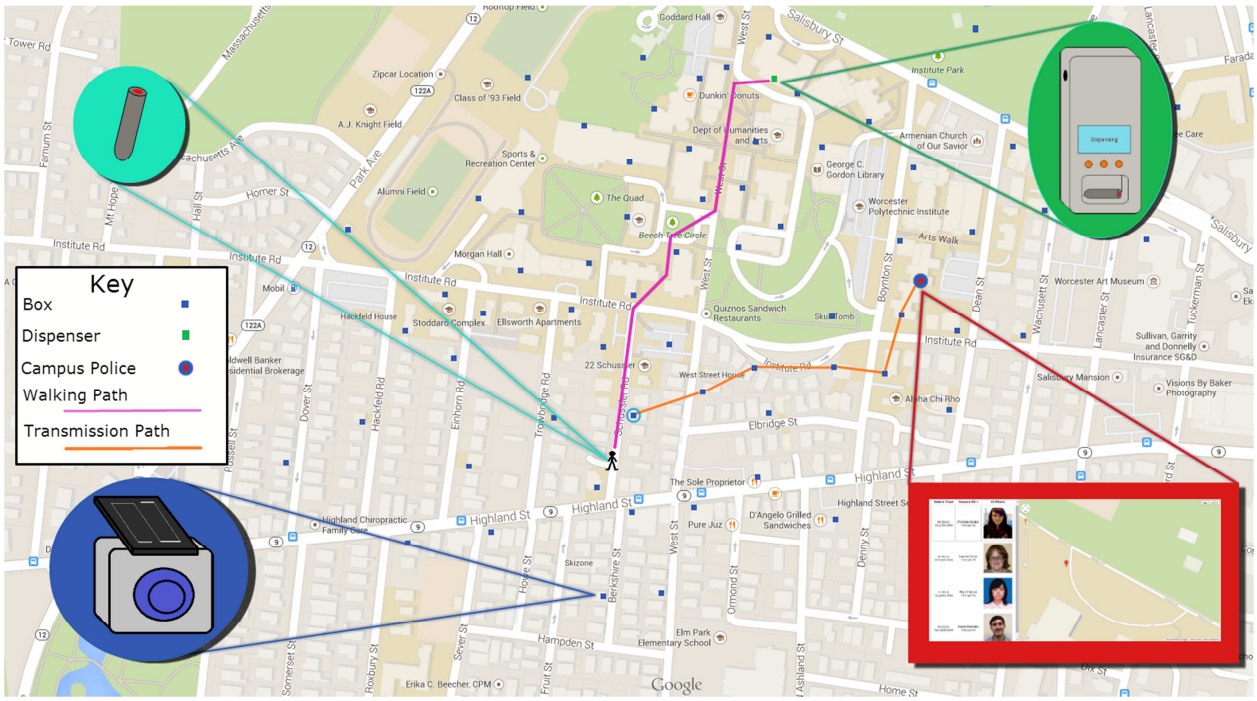


Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Problem Statement

The city of Worcester, like any large city, is bound to encounter a higher crime rate than the smaller cities and towns of Massachusetts. Threats to civilians range from thefts to murders and in recent years the crime rate in the area has been on the rise. This can be seen in the Table 1, which was derived from City-Data's website and portrays five years of crimes from 2007 to 2011. The final row of the table, about overall crime rate, is a unit-less metric calculated by City-Data [2].

Table 1: Worcester Crimes, 2007 to 2011

Type	**2007**	**2008**	**2009**	**2010**	**2011**
Murders	6	6	7	7	11
Rapes	93	42	25	19	36
Robberies	371	373	414	368	411
Assaults	1061	1297	1344	1350	1342
Burglaries	1308	1701	1485	1910	2066
Thefts	3815	3983	4522	3687	3481
Auto thefts	902	672	684	593	530
Arson	18	10	12	10	10
Crime rate (US average = 305.7)	422.0	432.0	438.9	414.7	431.3

Worcester is also home to ten college campuses, which form the Worcester Consortium. Colleges of the Worcester Consortium include Anna Maria College, Assumption College, Becker College, Clark University, the College of the Holy Cross, Mass College of Pharmacy and Health Sciences (MCPHS), Quinsigamond Community College, UMASS Medical, Worcester Polytechnic Institute (WPI), and Worcester State University [3]. While the rates of crime in the areas of these schools vary, the students at any one could be exposed to the crime present in Worcester. In fact, some safety notifications received by WPI students are about incidents which occurred at nearby universities, as these incidents can sometimes affect neighboring schools.

The image below was originally a map of Worcester, obtained from Neighborhood Scout's website, which divided up the city by neighborhoods and compared the crime rates of each neighborhood [1]. The lighter the shading of the map, the less safe the neighborhood area. Google Maps was used as a reference so that the Colleges of the Worcester Consortium could be pinpointed and mapped onto the image. As seen in this image, of all the colleges in the Consortium, WPI and Becker both lie in the neighborhood that is the least safe. Off-campus

apartments that WPI students tend to occupy also fall within this neighborhood. With this insight, it is not surprising that WPI students receive multiple safety notifications from Campus Police each term and occasionally receive notices in regards to neighboring campus incidents.

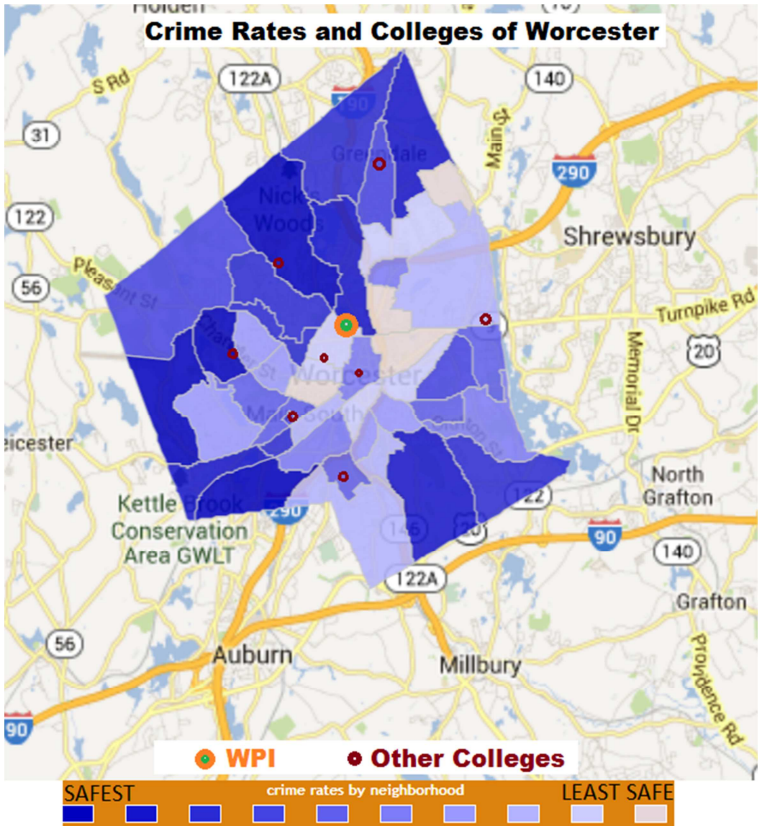


Figure 1: Crime Rates and Colleges of Worcester [1]

Table 2: WPI Crimes

Crime	Number of Times Reported (2010 - 2012)
Robbery	11
Aggravated Assault	5
Burglaries	22
Motor Vehicle Theft	5
Sex Offenses (Forcible)	7

Taken from WPI Campus Police's Annual Security and Fire Safety Report for 2013, the statistics in Table 2 show the crimes that occurred on the WPI campus from 2010-2012 [4]. These statistics show that there have been incidents on the WPI campus that threatened the safety and security of the WPI community. These numbers include only the incidents reported to the Campus Police on WPI's campus. It is possible that there have been additional unreported incidents on campus or in the surrounding area.

A Solution

As WPI students, this MQP group has been made aware of this safety problem in the form of security notifications from Campus Police, accounts from other students, and even experiencing the uncomfortable feeling of walking around off campus at night. It is for these reasons that the group wanted to satisfy their senior project requirement of demonstrating their abilities in electrical and computer engineering through the design of a product which would make the campus safer.

Building off of a project from an undergraduate design class, a remote bracelet device that students could wear to alert campus police of an emergency situation, the group designed a Remote Access Trigger System [ReAcTS]. From early research, it was found that the student perception of Campus Police and safety on campus was relatively low. The system designed is a solution based around the idea of providing students, faculty and staff with fast and simple remote access to WPI Campus Police. The remote access device can be rented from dispensers on campus and used to alert Campus Police in an emergency situation with the press of a button. This device, when activated, would send GPS location data via a modular relay system to Campus Police Headquarters. That information would not only bypass the need to use an on-campus security phone or one's own cell phone to dial the police, but it would also provide useful data to the dispatcher, thereby helping first responders get to the scene of a potential crime quickly.

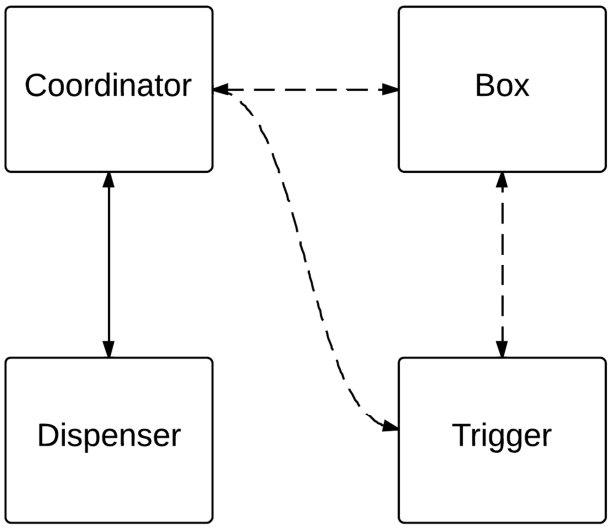


Figure 2: System block diagram

The system is comprised of the following parts: a Trigger, a Box, a Coordinator, and a Dispenser that retrieves and provides information to a user rental database. The handheld devices would inform campus police of the user's location at the press of the button. Since the user simply borrows the device, there would be no maintenance required on the user side. The relay network of Boxes would receive the emergency signal from the wireless module of the user's device and transmit it along the shortest route possible to Campus Police Headquarters. In order to have current location data, the handheld device would utilize satellites to capture the user's most recent location if he or she fled from the scene. The information sent along the network would converge at police headquarters through the Coordinator. This would be a receiver similar to the Boxes and an accompanying program which handles incoming emergency transmissions and does maintenance on the system. Finally, there would need to be a series of units to dispense the handheld devices to users. These dispensing units would be capable of keeping track of the devices rented and maintaining them when they are returned.

To meet the requirements of the system, a number of technologies were necessary. The Triggers contain GPS technology, and therefore also contain a low power microcontroller to interpret GPS data coming in as well as all of the data being sent out to the network. The handheld device, the relay boxes, and the Coordinator use wireless communication technology to create a network for sending and receiving packets of information. Similarly, information transfer between the dispensing units and the handheld units would be essential to track which student is using which handheld. In order for the handheld units to be maintained by the dispensing units, the dispensers were given the ability to charge the handheld. The Coordinator would need access to information from the dispensing units as well as the wireless network to present campus police with usable data. In order to allow the relay network to exist both on and off campus, the relay boxes were designed to be entirely wireless and largely independent of the power grid. In order to accomplish these goals, the group looked into technologies such as Bluetooth, Near Field Communication, and alternative energy sources. The group also researched what other companies were making in the way of similar products. Through research it was found that a product like this project would be very novel, in that no other system would be as user-friendly, modular, and inexpensive.

Chapter 2: Research

Current State of the Art

A number of other companies are working on similar solutions to this problem, and some are more recent than others. There are systems based around existing emergency tower systems, similar to the towers already on campus at WPI. Some systems are based on proprietary network infrastructure, some are based on smartphone apps, and some of these also use GPS.

Figure 3: An Emergency Tower on WPI Campus

The first product researched comes from Code Blue, a company working on the development of the “Circle of Safety System”, which provides remote access to the Code Blue emergency towers on some campuses [5]. This system includes a “panic button” pendant on the user side which would send a signal to the nearest emergency tower from up to 200 feet away when pressed. Upon receiving the emergency signal, the campus police can locate the activated tower and respond to the call. It also provides the police with identification of the user and links to his or her personal file. This system is currently in use at Butler University in Indianapolis as a test site and the panic button devices have been given to approximately 300 students. Code Blue targets university, corporate and medical campuses as its market. Since this system would be costly if the university or medical campuses decided to adopt it, one solution is to let individuals subscribe to the system at the rate of $75 for the pendants and $50 for the annual activation fee. This system provides a viable way to access the campus security remotely, but the user has to maintain the device themselves and additional cost may be too much of an additional burden on students who are already paying to be at university.

A product trying to address this problem through software is one called Campus Guardian. Rather than use dedicated hardware at all, this system uses an application for smartphones which provides a specific button to place an emergency call to campus police or 911. It provides “computer-aided dispatching” for campus police, and it uses a smartphone’s GPS radio to help inform responders. This is another robust solution, but one that is dependent on students owning smartphones and, more importantly, actually keeping them handy when an

emergency may happen, rather than in a purse or pocket. This also exposes the smartphone for theft in the emergency situation [6]. There is another, similar product called TapShield, which also uses an application on mobile phones to request help and communicate with campus police [7].

Another product on the market that is used for campus safety is called RAVENalert, which sends emergency notifications to students with a brief account of any emergency incidents [8]. RAVENalert devices are the size of a keychain and use wireless technology to send alerts to the students by means of voice, vibration and emergency messages. This device serves its purpose of notifying the students of emergency cases but it lacks the ability to let the students communicate with the campus security.

Figure 4: RAVENalert keychain [8]

Another newer system aiming to tackle the issues with campus security is called Campus Shield, by National Protective Systems, Inc. This system is probably the most similar to this project in its solution. It utilizes an infrastructure of receivers throughout a campus to listen for signals from key fob-like devices students can put on their keychain. Upon pressing and holding the button on the key fob, a signal is sent to the nearest receivers, and the system uses a “unique software algorithm” to determine the location of the user from the signals received. This is sent to a graphical interface for campus police to use to dispatch help to the scene. However, unlike this project, Campus Shield’s receivers have to be hardwired together to large switches, which then relay info to the police, and this will require potentially significant installation and power usage. Additionally, the user is again responsible for keeping a trigger on their person and potentially maintaining it himself, as opposed to a rental arrangement like with this project [9].

Figure 5: Campus Shield [9]

Campus Survey

To ascertain the opinions of WPI students regarding campus safety, a survey was conducted. In this survey, students were asked questions relating to security and safety on and around the WPI campus. One of the questions on the survey was about whether there are any apparent security problems at WPI.

A junior living in an off campus apartment said,

"I live extremely close to campus, and yet I don't feel safe walking from East to Honey Farms."

Another student writes,

"When I am on the campus I feel completely safe. There are students around and I don't feel likely to encounter danger, however, the second I step off campus at night, even on institute and Boynton, I feel less secure."

A sophomore comments,

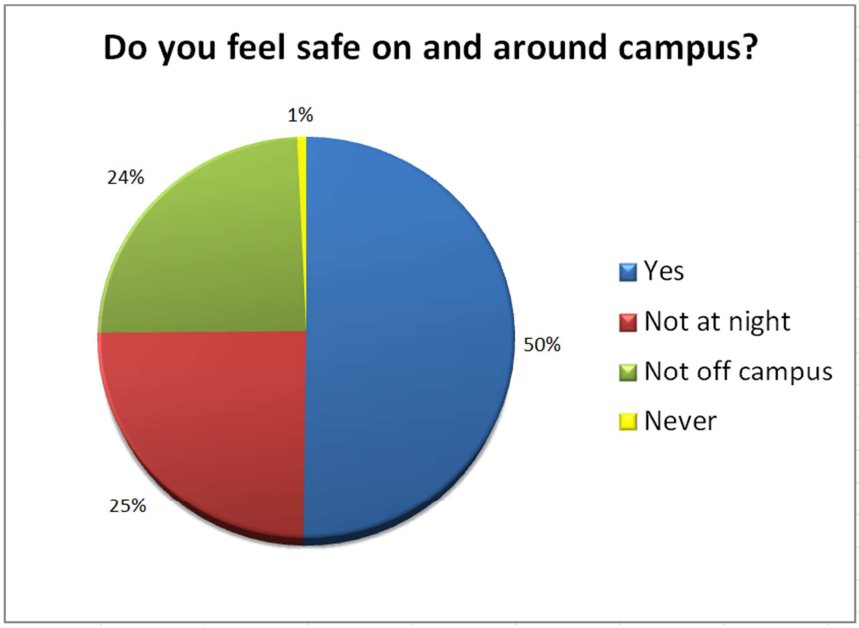
"I'm afraid of walking home to my apartment. However I know it's closer than a lot of 'on campus' housing. For example, I would never live in faraday."

A senior observes,

"WPI campus itself is pretty safe, especially because of the blue light program. The problem is once you are a block away there are no more blue lights[.]"

Based on these responses, students generally feel safe on the WPI campus, but when they leave the campus, especially at night, students no longer feel safe.

An additional question asked was to inquire if the students felt safe on and around campus. The results from this question showed that around half of the respondents do not feel safe on and around campus at all times. The results from this question are shown in Figure 6 below.



Yes	**346**	50%
Not at night	**170**	25%
Not off campus	**168**	24%
Never	**5**	1%

Figure 6: Chart showing Student Responses

Based on the results from the survey, students are looking for another solution to solve the issue of crime around WPI. The last question that was asked of the students was relating to suggestions for security improvement on and around campus. Around thirty percent of students were interested in either more blue light towers or a way to remotely access blue light towers. The results from this question are shown below.

Improvements for security/safety on and around campus?

Additional blue light towers	**210**	15%
Additional lighting	**456**	32%
Remote access to blue light towers	**216**	15%
Easier access to Campus Police	**233**	17%
Opt-in safety initiatives	**134**	10%
None	**78**	6%
Other	**78**	6%

(To see the rest of the results from the survey, refer to Appendix A)

System and Component Research

To implement this idea, there needed to be a wireless network between the Triggers (handheld units) and the Boxes (relay units). For this project, three different wireless networks were researched. The first wireless network considered was Bluetooth. This tech is an inexpensive solution to this project and it uses less power than other wireless technologies [10]. The main issue with Bluetooth is that all of the devices need to be paired with each other. This means that a Bluetooth network would not scale well into a larger network [11]. Based on this, Bluetooth was not the ideal wireless network for this project.

The second wireless network considered for this project was Wi-Fi. Some positives of Wi-Fi are that it supports fast transmission speeds. For this project, speed is not an issue since the data being sent through the wireless network is small. A con of using Wi-Fi as the wireless network is that it consumes a lot of power. Wi-Fi also requires a lot of software support, such as a much more powerful CPU [10]. Another con for using Wi-Fi is that it is more expensive than the next wireless network, Zigbee [11].

The last wireless network researched, and the wireless network chosen for this project, was Zigbee. Zigbee is not as fast as Wi-Fi, since the data rate is usually 250KB/s in comparison to the data rate for Wi-Fi which is about 54 Mb/s. For this project, the data being sent between the Trigger and each Box is small enough so that 250KB/s is more than enough speed [11]. Another positive for Zigbee is that it can support thousands of nodes in a single network. A Zigbee network can be configured in a point-to-point network or in a mesh network. An example of a point-to-point network can be seen in Figure 6. In a point-to-point network, all of the nodes are able to communicate with each other, and send data to each node. For this project, this network is not the best option, since there is a superior option available. The other network that Zigbee modules can be configured as is a mesh network. An example of a mesh network is shown in Figure 7. A mesh network uses a different approach to sending data throughout the network. In each mesh network, there is one coordinator that maintains the network by

managing the associated devices and routes. Routers send data between nodes that cannot communicate directly due to distance. Endpoints are the last part of the network, which are connected to controllers and sensors. In this project, there would be one coordinator in the network, which is part of the Coordinator at police headquarters. Routers would be used in the Boxes, and Triggers would be endpoints [12].

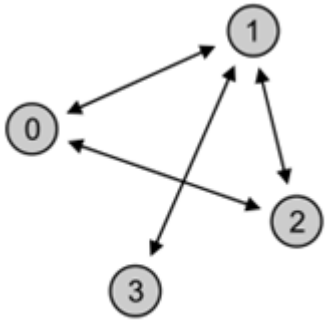


Figure 7: Point-To-Point Network [12]

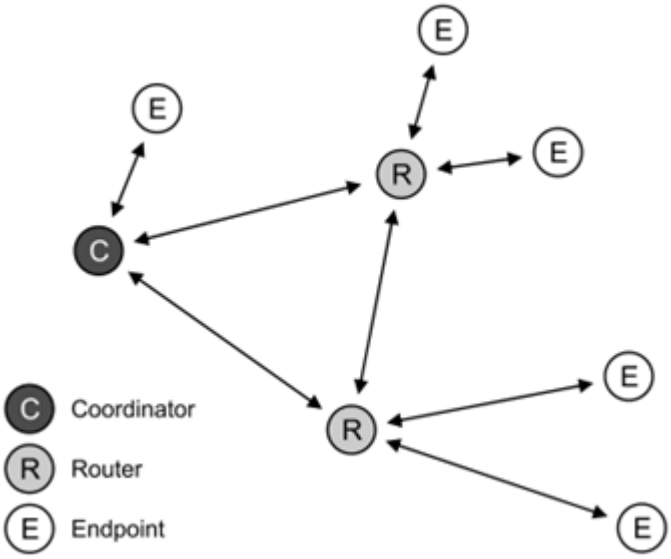


Figure 8: Mesh Network [12]

With today's wide market of GPS receivers, GPS provides a convenient and inexpensive solution to track location. Research was done on the types of GPS receivers to ensure that it would be a good fit for the project. Receivers get information from the GPS satellites orbiting around the earth and use triangulation to calculate the user's location. The information from the satellites gives three-dimensional data – longitude, latitude and altitude. There are different types of GPS receivers on the market. The standalone systems are not applicable for this project. However GPS modules, which can be embedded into a device, are a good match. The output of the module is a string of ASCII characters which can be converted into longitude, latitude and altitude data by using the NMEA protocol, a standard protocol for GPS systems.

The prices of GPS modules range from about $15 to $60 and they are much less expensive than the complete stand-alone receivers. The GPS modules are small in size, have an update rate of about 5 to 10Hz, have average power requirement of 30 mA at 3.3 V, and an accuracy of up to 3m depending the number of channels of the GPS module. Besides these specifications, there is also a wide availability of software to convert GPS data into maps such as those used for Google Maps or Google Earth [13]. For example, software called GPS Visualizer can convert the GPS data into a readable format that gives longitude and latitude information. The GPS data files can then be imported into this program and mapped into Google Maps. Based on this information, we decided to include a GPS module into the trigger because the trigger needs to provide accurate location data, have reasonable power consumption and update rate, and software to easily recover the GPS data into the server.

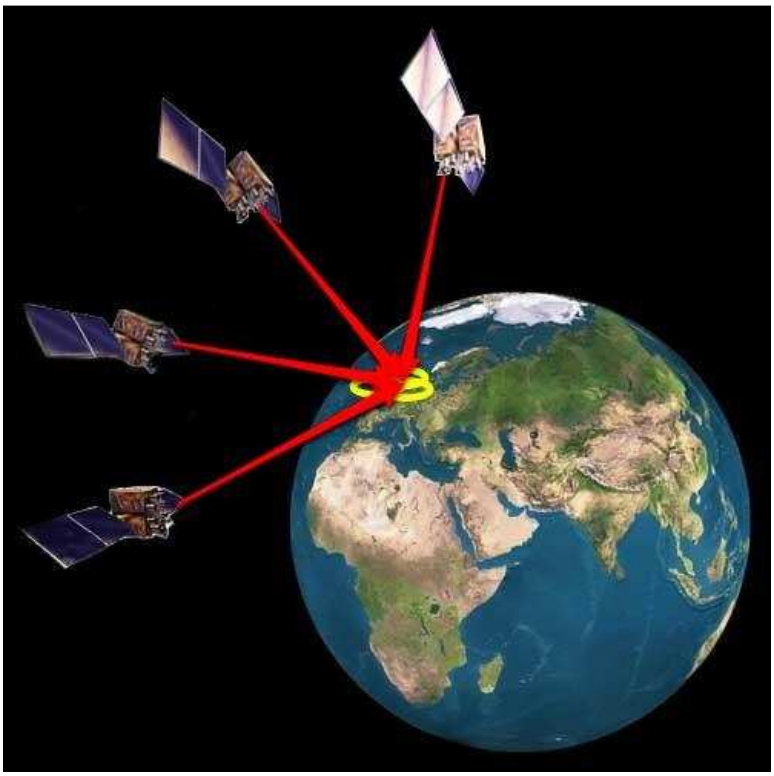


Figure 9: Concept diagram of GPS receiving data from satellites [14]

Because the Zigbee and GPS modules both require power to operate, the trigger must have a way of charging. Wireless power is a relatively new, up and coming technology that has the potential to replace existing wired solutions. One appeal of wireless power is the elimination of cables and ports to allow for a sleeker and more durable design. The origins of wireless power can be traced all the way back to a patent Nikola Tesla filed in 1902. Today devices ranging from toothbrushes to smartphones are gradually gaining the capability of wireless charging [15].

There are several means by which one can charge a device wirelessly, the most explored and popular of which is the Qi standard developed by the Wireless Power Consortium (WPC) [16]. This standard uses two magnetic coils to transfer power from a charging pad to a device [17]. These magnetic coils are created by introducing a coil of wire to a magnetic field of

alternating current, allowing for the induction of electricity [15]. Although a charging pad is a more efficient methods of wireless power, its range is not very great and can only send over distances of 5 mm or less. When this standard was updated in 2012, the transmitter was then able to send up to 5 watts of power 40 mm [18]. The resonant frequency of the WPC can be selected between the ranges of 100 KHz up to 205 kHz. This technology tends to use either a half-bridge or a full bridge inverter in the transmitter's circuit for DC to AC conversion [19].

The Power Matters Alliance (PMA) is like the WPC in that it is based on inductive charging technology. However, the PMA focuses less on charging pads and accessories and more on the availability of hot spots in company's stores [16]. Basically, this standards organization aims to make wireless charging a publicly available feature through partnership with a range of companies [20]. One day, perhaps this would allow for a user to seamlessly transition from one charging hotspot to another with no need to actually think about the fact that wireless charging was occurring. The PMA standard has a frequency range from 277 kHz to 357 kHz [19].

The Alliance for Wireless Power (A4WP) is a standardization organization that aims for "spatial freedom for charging of electrical devices." This standard organization allows several devices to be charged simultaneously and at greater distances [21]. This standard uses magnetic resonance [22]. This type of charging, unlike typical magnetic induction, only transfers power at a particular resonant frequency and a power amplifier is used to induce current. To achieve power transfer in WPC and PMA, the transmitter will periodically search for the receiver. However, in A4WP this same technique cannot be utilized and amplitude, power, or current modulation might be used instead if circumstances were correct. Generally, Bluetooth or Zigbee are better options for A4WP communications [19].

Inductive charging uses electromagnetic fields to transfer energy between the transmitter and receiver. An oscillating current at the primary coil produces varying electromagnetic fields which in turn induces current in the secondary coil. The secondary coil is connected to the load to transfer energy [23]. In order to maximize the efficiency of the energy transfer using the coils, the resonant inductive coupling should be used [24]. At the resonant frequency, the system oscillates with the highest amplitude. In considering the efficiency of wireless charging, one should be aware that the coupling coefficient of the air is very small resulting in low efficiency. The coupling coefficient is the fraction of the electromagnetic flux transferred from one coil to another [25]. With normal inductive charging, the efficiency of the system will be low because of the small coupling coefficient of air. Therefore, for wireless charging between the trigger and dispenser, resonant inductive charging would be the best technique. This would be ideal for this project since several triggers would need to be charged in the dispenser using wireless charging. With resonant charging, each trigger can be configured with unique resonant frequency. At the transmitter side, this can be done by using a variable capacitor to adjust the resonant frequency [26].

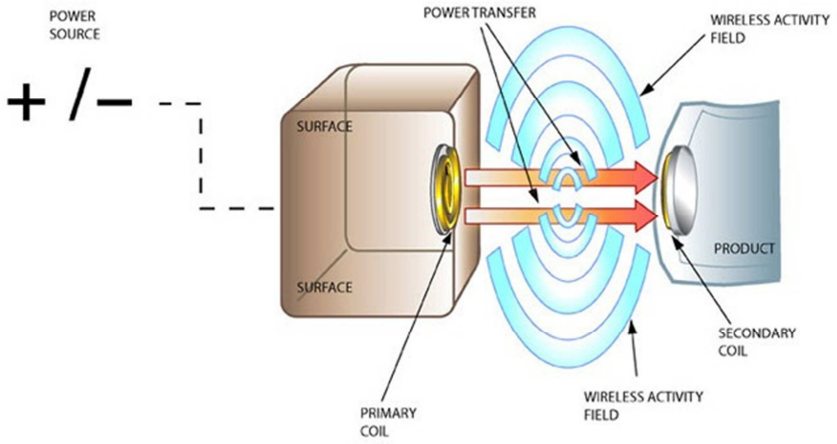


Figure 10: Inductive Power Transfer [27]

One of the most important requirements set in our goals for this project was that the system, in particular the relay boxes, used renewable energy. This is for multiple reasons, starting with the importance of modularity. In order for the system to be easily expandable, especially off campus, there can't be reliance of hardline power or data lines. Since these relay boxes are relying on wireless data transfer, should not be powered from the grid, which means there needs to be another way to deliver power to the boxes. Additionally, with the vast expansion of the idea of being "green," there is no reason not to try less utilization of traditional energy when possible.

Given that the system would be relatively small, solar was prioritized over other alternative sources like wind power. Sometimes intermittent and unreliable, wind power would not suit the reliability necessary of a safety system. While larger farms of turbines would be incredibly reliable, as the impact of each individual turbine is small enough so as not to disrupt power generation of the system as a whole, the relay box units would be far too small to mount enough turbines upon it. Additionally, wind is not as easy to predict as several other power sources, namely solar.

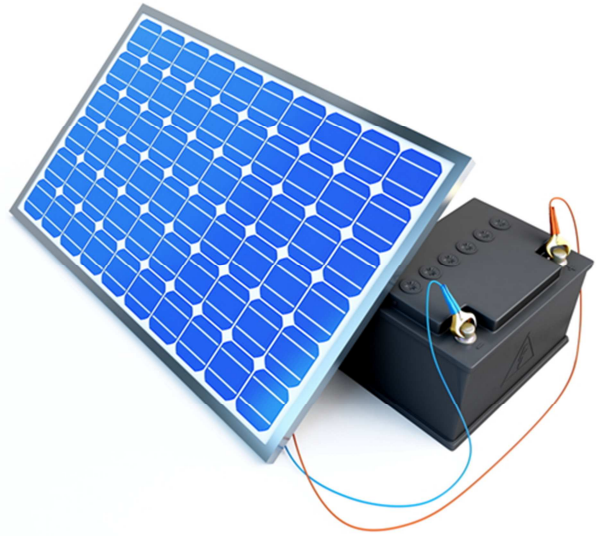


Figure 11: Solar panel charging battery [28]

With the hours of sunlight being largely predictable, it would be significantly easier to use solar power in this project. Combined with an appropriately sized battery bank, solar power can be relied on for both day and night operations. There are three main types of crystal panels used to harvest solar power: monocrystalline, polycrystalline, and amorphous. The first is made from a single crystal and is the most efficient. However, the cost of that efficiency is how incredibly fragile it is. The polycrystalline is better on that front without too much of a performance drop. Finally there is amorphous, which is a non-crystalline form of silicon, makes up for a reduced efficiency with far more durability. Given the parameters of the project, it is likely amorphous cells will be used, as durability is our priority with the relay boxes. It's unlikely much more in the way of efficiency, as the relay boxes will be fairly low power.

As with all electronic devices not tethered to a wall for main power, the trigger and box need batteries to store energy. Research into what batteries would be suitable for these devices began with a few requirements, namely size, capacity, and nominal voltage. Given that both of these devices have specific charging methods, the batteries would also need to be rechargeable and ease of charging was most desirable. A few types of rechargeable batteries with various chemistries were considered, specifically NiCd, NiMH, NiZn and Li-ion. Nickel cadmium batteries are fairly inexpensive and are good for high and low energy drains, but they're hard to find due to the environmental risk posed by the cadmium. Additionally, they tend to experience a "memory effect", which reduces capacity over time. NiMH are similar to NiCd in a lot of ways, but they do not contain environmentally harmful heavy metals, so they are easy to obtain and inexpensive. They come in a variety of shapes, sizes, and capacities, but they are difficult to monitor to prevent against overcharging if using faster charging methods. NiZn batteries are a newer type that have a lot to offer, including low self-discharge rates and ease of recharging quickly, as well as the ability to fully discharge and be recharged without issue. However, they can't be trickle charged, they're more expensive than some alternatives, and they are less readily available. The final consideration for the battery was Lithium Ion. These have high energy density and very low self-discharge. However, they aren't usually available in more compact sizes that would be suitable for the trigger, they're expensive and volatile, and the nominal voltage is unstable. Additionally, charging Li-ion requires a lot of sensing and regulating to prevent damage to the battery as well as for the safety of the rest of the device and user. It was decided that the best balance of desirable traits would be NiMH, which are the least expensive and most readily available, and they can easily be trickle charged. Additionally, their capacity is excellent, and they perform well under high drain conditions. This should allow them to last long enough for users to have overnight between charges and easily switch into emergency mode without depleting the battery quickly.

Table 3: Important Battery Characteristics [29]

Type	NiCd	NiMH	NiZn	Li-ion
Nominal Voltage (V)	1.2	1.2	1.65	3.7
Discharge (%/mon)	10	8	13	5
Cycle life (#)	1500	1000	100-500	500-1000
Cost [30]	$7.52	$9.60	$13.77	$10
Ease of charging	Medium	Easy	Medium	Difficult

In addition to the essential hardware research for the system, knowledge of how the ID system on campus operates was needed. One feature of the project would be to allow the dispenser to read an individual trigger's ID when a student or other member of the WPI community attempted to rent it. When this trigger's ID was read, the dispensing unit would then need the capability to associate the student ID and trigger ID in a database. That way, when an emergency signal was sent via the Zigbee module, the unique identification of the trigger that the module provides would allow the Campus Police to then determine which student was in trouble. Providing identifying information such as a name and picture in addition to location data would allow the first responders to the scene to quickly find the potential victim and come to his or her aid. In order to gain understanding of how WPI runs the campus ID system, Amy Beth Polonsky Laythe, associate director of operations, was referenced. From the conversation with her, it was learned that the system WPI uses is called banner and it makes use of an Oracle database. Branching off of banner are services such as Lenel for access to buildings and Sbord as the transaction software. She also informed that IT is in control of the database and it is kept very secure so accessing it would potentially require jumping through hoops. However, the system need not know every detail about a WPI campus member. Rather, it could be one of the branches off of banner that only keeps track of an individual's name, picture, ID number, and how many triggers they currently have checked out. From the conversation with Amy Beth, it was determined that this was a possibility down the road [31]. For the scope of this project, a sample database with simple information to prove that this system would work would be used.

Figure 12: HID RFID readers on WPI campus [32]

When designing a product, it is essential to get feedback from potential customers so that the design will fulfill their needs without making blind decisions. Since this product is meant for use by the campus police, an interview with Sergeant Jacobs would be helpful. He has been part of the WPI police department for the past three years and is in his 20th year of law enforcement. He informed that as far as campus police departments go, WPI's department is rated as one of the higher ones.

As for the current state of safety and security on campus, WPI is "a lot better off than...some of [its] counterparts in the consortium [because it has] a very progressive department." In fact, the department has put together a "rapid response team," which is essentially like a SWAT team, according to Sergeant Jacobs. This team consists of six members of the department who have been trained to a higher level and are held accountable to run teams in the event that a large emergency situation arises on or around campus. This is excellent news for the project, because it is apparent the campus police are open minded to advancement and actually making efforts towards improvement.

Figure 13: WPI Campus Police Department [33]

When it was inquired what the Sergeant thought of the current system of emergency towers on campus, he said he was not a fan of them because they are dated. He went on to say that an actual emergency occurred where a girl pressed the wrong button to contact campus police because the indistinctness of the two call buttons on the towers. For this reason, she panicked and instead dialed 911, which actually calls the state police and prolongs the process of getting help to an individual in immediate danger. Clearly this situation is far from ideal and he reinforced the idea that a safety system needs the simplicity of one button and the lowest time delay possible between becoming aware of an emergency and responding to it.

One aspect of this project is the potential need for maintenance of the boxes and dispensers, so it was also excellent to hear that the department takes the time to audit and maintain the current system. Upon explaining this project's concept to Sergeant Jacobs, he said that it would be preferable to provide a simple and clear map or street location as opposed to a location name because occasionally the Worcester police department does details and they wouldn't be familiar with the names of campus buildings. As far as identifying a student on the headquarters side, he said that their system would just need a student's name and ID to access the rest of that student's information, as it would be most helpful for the responding officer to have the student's picture and know who to look for on a potentially populated street. Sergeant Jacobs was also of the opinion that the trigger and system should be discrete, so that the police could catch the criminal in the act as opposed to the criminal having a chance to escape and attack again at a different time.

Another new piece of information that was learned from the Sergeant was that the police cruisers have laptops in them. Therefore it would be highly beneficial for the system to pop up the student's name, image and location on the cruiser laptops, as it would eliminate the extra time spent when the dispatcher has to contact the cruiser. He was very willing to talk to his chief about this project, as he saw it as a step in the right direction [34]. Overall, speaking with Sergeant Jacobs was very educational and beneficial because with the information he provided, we could better imagine how to best design and implement this project.

Chapter 3: Design

Concept of Operations

The system would ideally operate in two main modes: Emergency Mode and Standby Mode. When the button on the handheld, or Trigger, is pressed, the system enters Emergency Mode. At this point, the XBee ZB and GPS radios are activated in the Trigger to begin receiving position data from satellites and communicating with the closest relay, or Box. Once communication with a Box is established, identifying information about the Trigger is sent to this Box to be relayed to Campus Police, along with GPS position information. At the Box level, information received is immediately routed to the next closest Box that is on the way to the Coordinator Box at campus police headquarters. At that point, this Box would have a hardwire connection to an interface at the dispatcher's desk.

In Standby Mode, the Boxes continue to look for incoming transmissions from Triggers, but are otherwise inactive. Standby Mode is also included in the way the Dispensers function, specifically how they distribute Triggers to users. As illustrated below, the user would begin by scanning their Campus ID on the RFID reader. They would then be prompted with commands on the Dispenser's screen to Borrow or Exit. If they choose to Borrow, a Trigger is dispensed. As it is dispensed, it is activated and any charging that may be occurring stops. To return a Trigger, the user would place the Trigger in the indicated slot. The returned Trigger is then deactivated and charging begins. When activating or deactivating the Triggers, the Dispenser checks in with the campus ID server and updates a database accessible by the police department with who has rented which trigger.

In Figure 8 below, the system in action is displayed, with a user in trouble sending GPS data from a Trigger they borrowed from Atwater Kent.

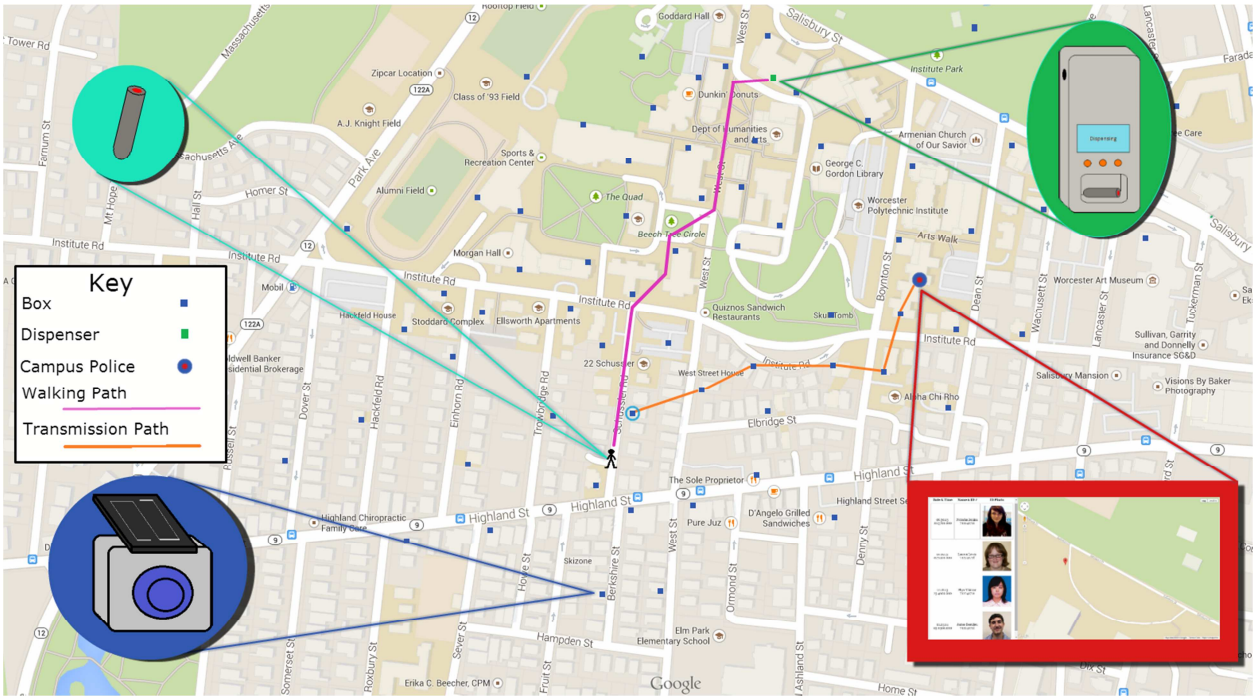


Figure 14: Basic concept of operations

Design Outline

The Trigger is a one button device that communicates with the campus police by sending a signal through the relay of boxes. The system is designed to be durable, easy to use, accessible and cost effective. It has emergency notification, tracking, wireless charging and user identification capabilities. The size of the Trigger is small enough so that the user can hold it with one hand and easily press the button when the emergency occurs. It has hard plastic for its outer casing.

As illustrated in the block diagram Figure 15, the Trigger starts at the user interface level, which is simply a single button. This button, when pressed, activates the emergency program in the controller, which then communicates with the GPS receiver and XBee ZB radios. When in Standby Mode, only the XBee module remains active. The Trigger contains power storage in the form of rechargeable batteries, and an interface between that storage and the controller and its peripherals.

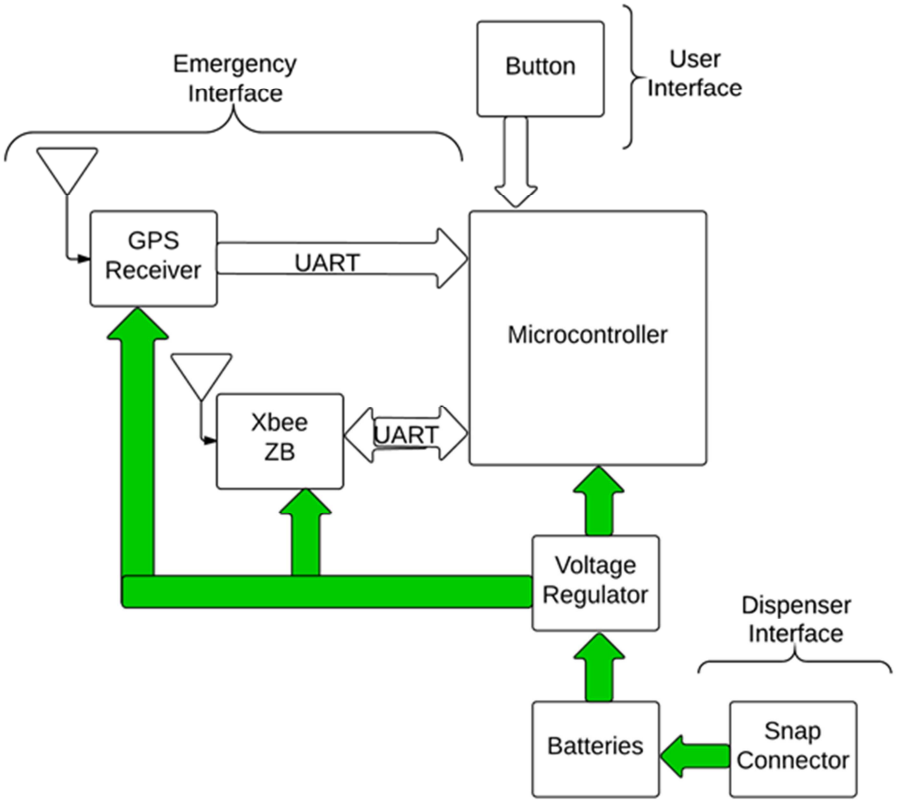


Figure 15: Block Diagram for Trigger

When the Trigger first turns on, it goes into Standby Mode. This mode is for when the Trigger is waiting inside of the Dispenser to be borrowed. When the device is checked out by a user, it is initialized. When it is in this mode, it waits to see if the emergency button has been pressed. If the emergency button is pressed on the Trigger, it simultaneously activates the GPS and the network radio inside. After it activates both of the radios, it sends its GPS data through the wireless network back to Campus Police. A flowchart for this device is shown in Figure 16.

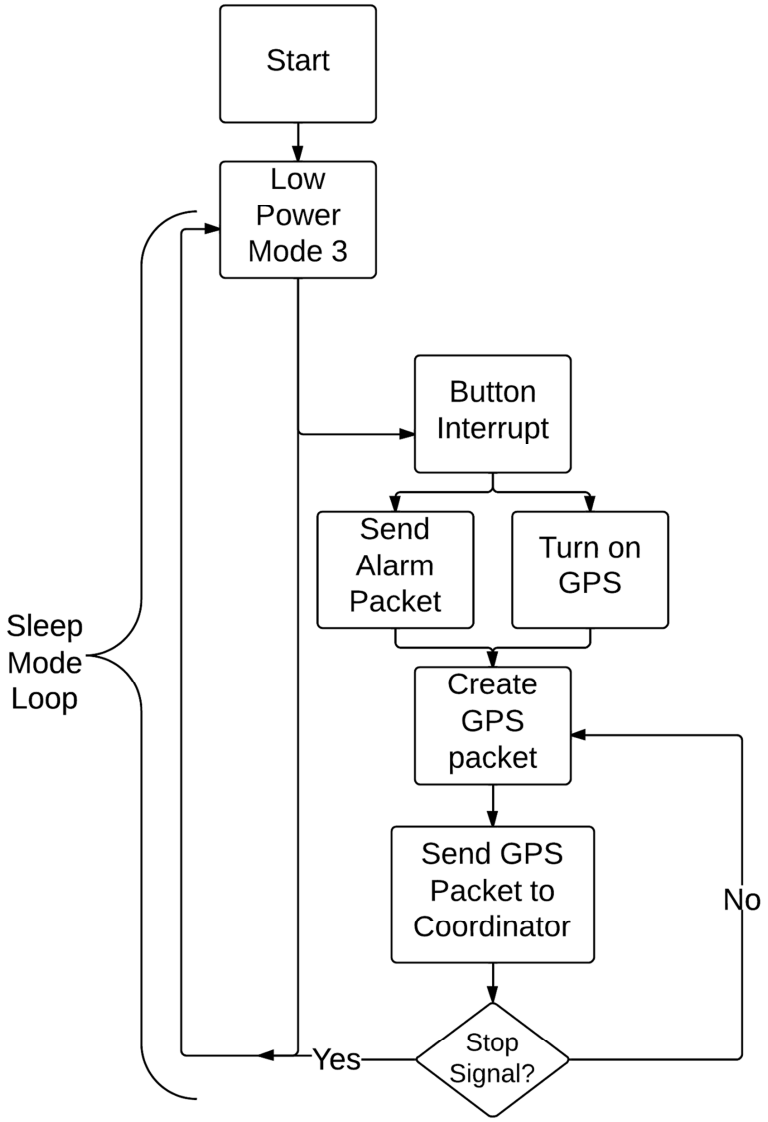


Figure 16: Software Flowchart for Trigger

The purpose of the Box is to relay the emergency signal, GPS coordinates, and identifying information about the user to the interface at Campus Police Headquarters. The Box nearest to an activated Trigger will receive data from and relay it to the next nearest Box, taking the shortest path possible, to the Headquarters Box, which interfaces with the dispatching officer. The Box is designed to be completely modular and wireless so that the infrastructure will be easily expandable. It also has an alarm system consisting of a speaker and a series of LEDs which will be activated at the first Box activated in the emergency. These boxes will be placed right outside of the buildings so that they will be easily visible. In keeping with a determination to be entirely wireless and green, the Box will use solar energy to charge its internal power storage. To be used in New England, the Box will also have water resistant metal for casing to accommodate the harsh weather, particularly during the winter.

As illustrated in Figure 17, the block diagram for the Box shows its basic operation.

Beginning with its power configuration, the storage can accept power from either the built in solar panels. Data flow will occur in Emergency Mode, with data received through the XBee ZB transceiver entering the controller to be interpreted, and routing information will be added. The controller will then output the data to the next Box in the relay path through the transceiver. Finally, on the right side of the block diagram is the alarm system, consisting of LEDs and a siren. This will only activate in the Box closest to the active Trigger.

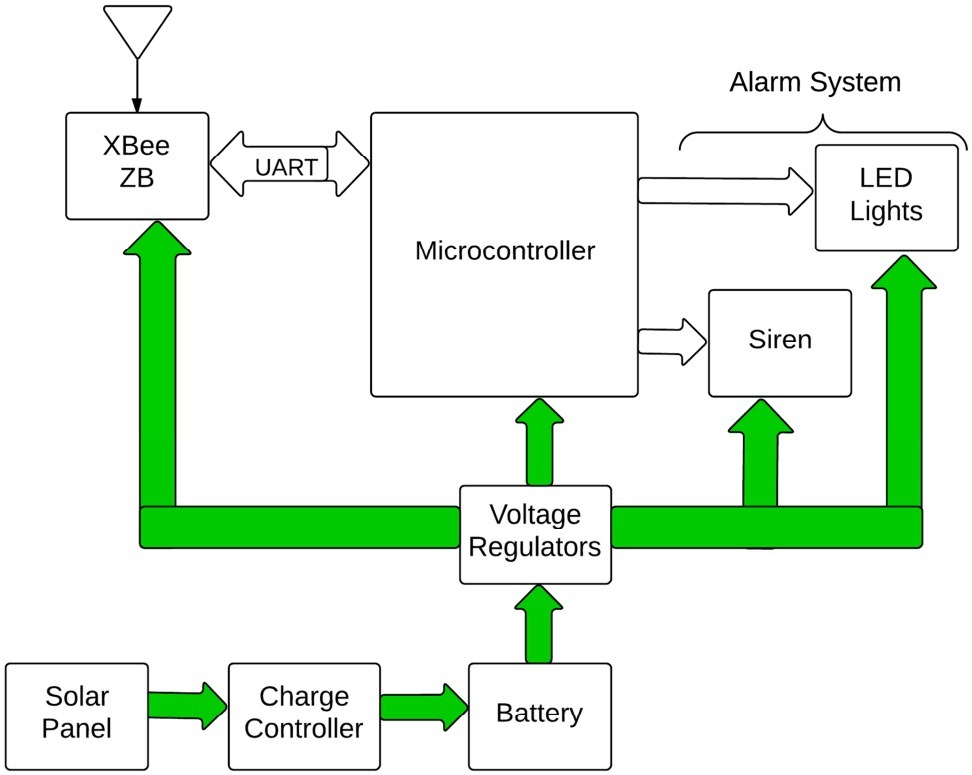


Figure 17: Block Diagram for Box

When the Box turns on for the first time, it goes into Standby mode. This is a low power mode where the Box will wait for a signal from a Trigger or another Box in the network. When an emergency signal is received, it checks to determine if the signal is received from a Trigger or another Box. If the signal is from a Trigger, the Box will concurrently send the signal to the Headquarters via the relay network and set off the alarm system. However, if the signal is from another Box, the Box will just forward the emergency signal to the Headquarters. This Box will also stay in emergency mode until the stop signal is received. This is illustrated in Figure 18.

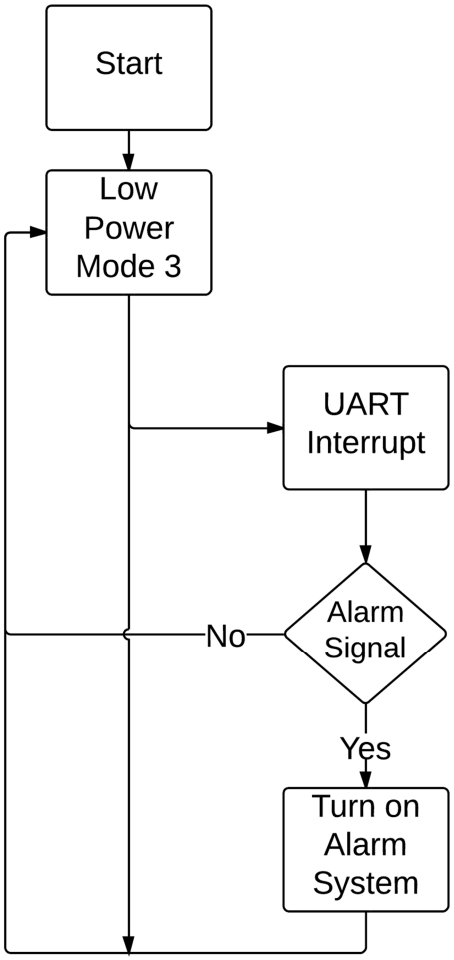


Figure 18: Software Flowchart for Box

The Dispenser contains, charge, and dispenses Triggers to users. It will allow users to check out Triggers using their Campus ID. It will check with the ID server and if the user has not borrowed more than the allotted number of Triggers, they will be allowed to check one out. When a Trigger is checked out, the Dispenser will activate it into Ready Mode. The Dispenser will also charge unused Triggers being stored in it via inductive charging. After the student is done with their Trigger, they can return it to a Dispenser. The dispensing units would be placed inside academic buildings and residence halls near the entrance, so that when user is leaving the building they can check out a Trigger if they feel they need one.

The block diagram seen in Figure 19 displays a system layout of the Dispenser. Starting with the user interface, the RFID reader communicates with the controller and ID server on campus to pull student data to tie to the dispensed Trigger. This action would prompt the screen to display options to the user, who can use one or more buttons to make a selection. The built in NFC radio communicates information with the Triggers inside to activate them and store important user information, as well as monitor the battery of the Triggers being charged. The Dispenser will be attached to the grid for power, which will not only power this module itself, but will also charge the Triggers.

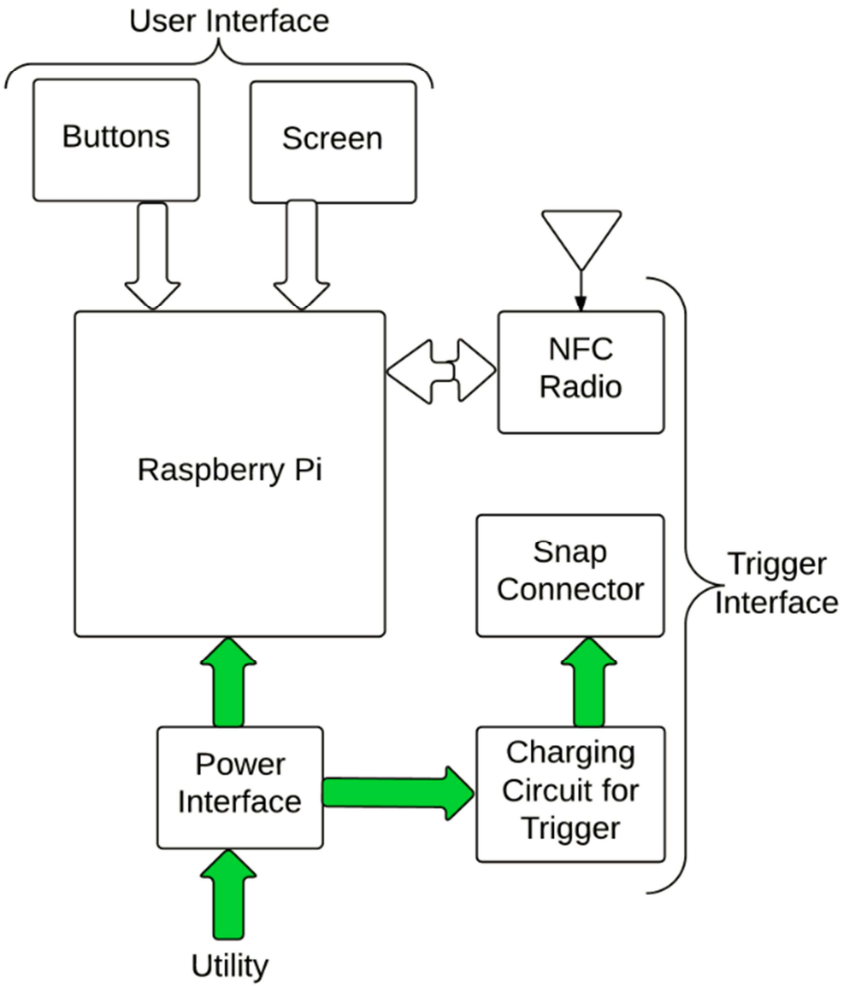


Figure 19: Block Diagram for Dispenser

When the Dispenser turns on, it goes into Standby Mode. It stays in this mode until a Campus ID is scanned. After it scans the Campus ID, it checks to see if it has any Triggers to lend to the user. If the Dispenser does not have any, then it will still allow users to return Triggers, but not borrow them. If the Dispenser has Triggers to lend it will allow users to borrow or return Triggers. When a user chooses to return, the Dispenser will scan the ID of the Trigger using Near Field Communication. After it confirms which Trigger is being returned, it updates the server that handles Trigger loan status of users. The Dispenser then displays the end screen and goes back to standby. If a user would like to borrow a Trigger, then the system will check to confirm that they have not checked out more than the allotted number of devices. If the user has checked out more than that allowance, the screen will display a message about calling Campus Police to get back to their residence, providing a number they can call. However, if the user is allowed to check out a Trigger, the Dispenser initializes one and updates the server. After the server has been updated, the Trigger is provided to the user. This is seen in Figure 20.

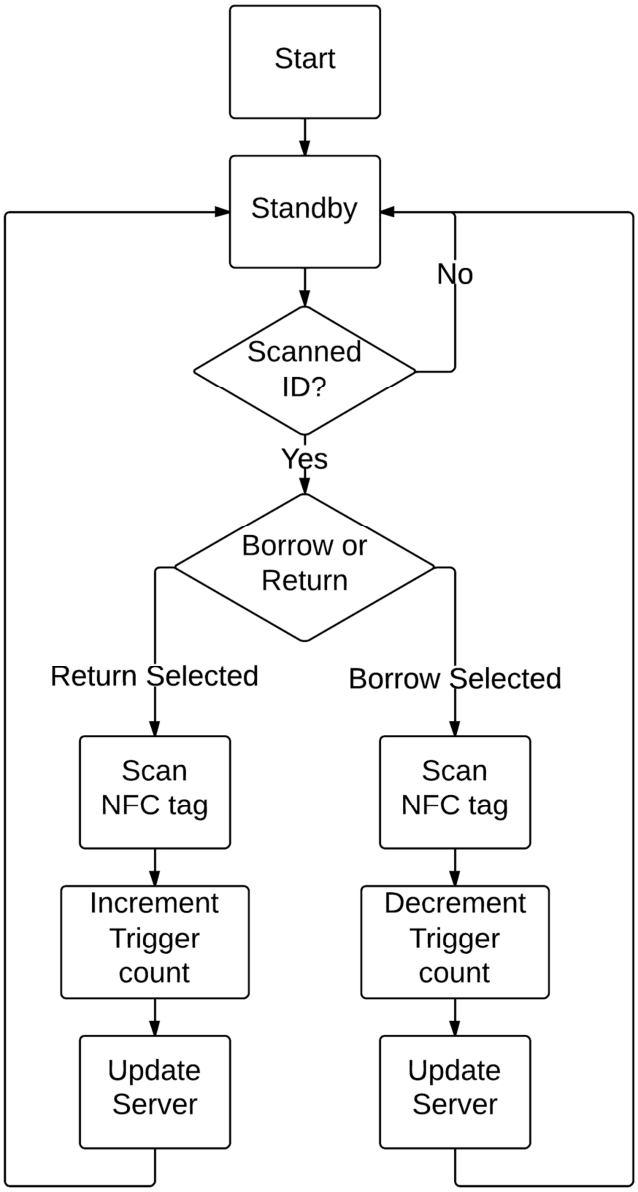


Figure 20: Software Flowchart for Dispenser

The Coordinator is a multithreaded program which both tracks inventory and maintains the Zigbee network simultaneously. It is important that things run in multiple threads so that the program can always be listening for incoming updates from Dispensers, Boxes, or Triggers. When it begins, it spawns a new thread, which will be the master server thread for Dispenser communication. It then continues on into what is now the master server thread for the Zigbee network. For initialization, it sets up communication with the XBee hardware which is connected via USB. It then waits and listens on that XBee device for incoming communication from others in the network. If a packet is received, a new thread is spawned to handle this incoming packet. It first parses the information in the payload of the packet to determine what its purpose is. Right now only the emergency functionality is implemented, but it would be here that various functions

could be called for a variety of system maintenance features. If it is an emergency packet, it calls a function which pulls the payload apart into the latitude and longitude coordinates. These are then formatted and written into the KML file linked to Google Earth. The thread will then exit, and the master Zigbee thread is still listening for other emergencies.

Meanwhile, the master server thread for inventory control will also be running. When this thread is created, it initializes itself by opening a socket in an open port to listen on for messages. It then waits and listens for incoming communication from Dispensers. If a packet is received, a new thread is spawned to check the contents of the payload for the desired update. Like in the Zigbee threads, this would be the place to call various functions for other functionality, but currently only checking in and out is implemented. Depending on what the payload is, the thread calls an inventory control function for changing the contents of the Dispenser and Trigger involved in the transaction. Finally, this thread is closed, though the master thread is left always listening for incoming information from Dispensers. All of this can be seen in Figure 21 below.

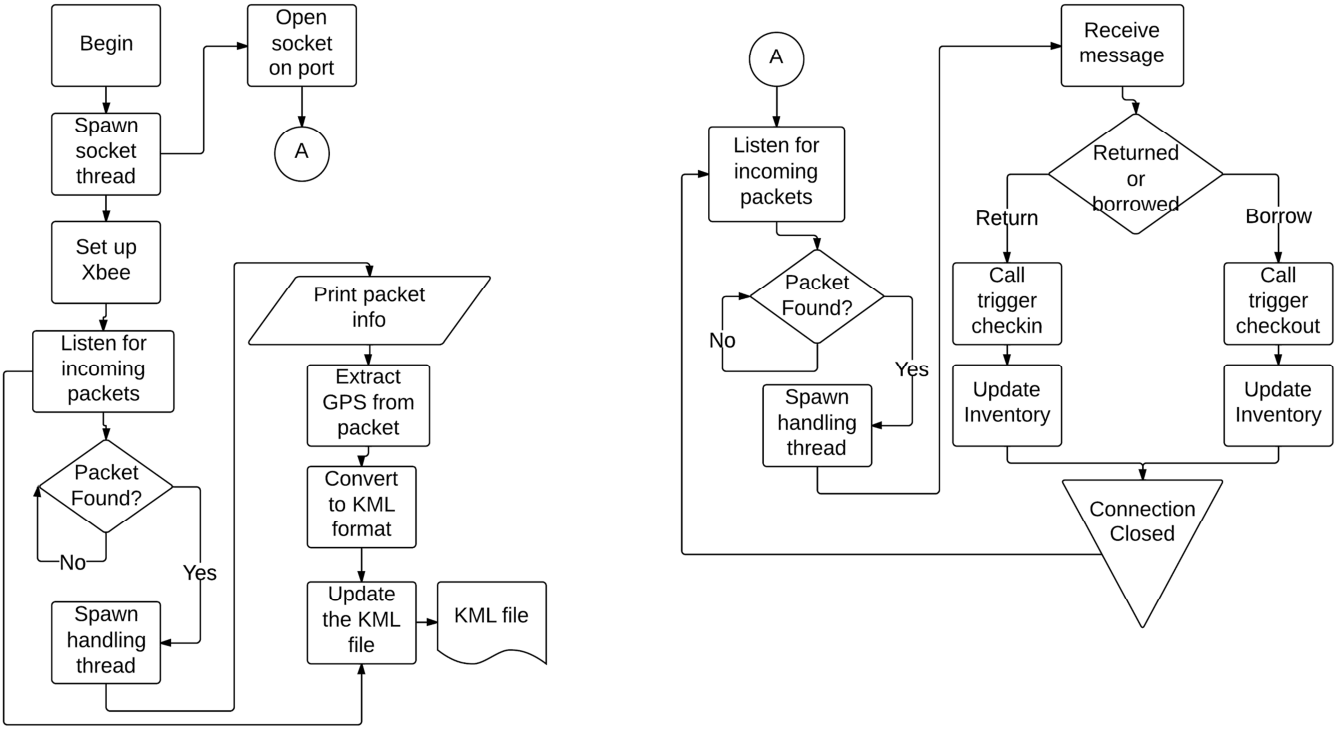


Figure 21: Software Flowchart for Coordinator

Module Selection

Wireless Communication

Based on the research of Zigbee networks, for this application, a mesh network was selected as the best option for creating the relay network. The Triggers in the network do not need to talk to each other. One company that makes Zigbee modules capable of creating a mesh network is Digi [35]. The series that allows us to create a mesh network is the XBee ZB series. The XBee ZB series has two main models, the XBee ZB and the XBee ZB Pro. The main difference between these two models is their range. The Pro series have a range of 300 ft in an urban setting and two miles for a line of sight range. The XBee ZB series have a range of 133 ft in an urban setting and 400 ft in a line of sight range.

For this project, the XBee ZB series had a long enough range for use in the devices. The next step would be to determine which antenna package would be used. For the XBee ZB modules, there are four different antenna packages. The four antenna packages that can be used with the XBee ZB modules are the wire antenna, the U.FL antenna, the PCB antenna, and the RPSMA antenna. For the Trigger and the Box, the wire antenna would be more than enough for their transmissions to each other. This means that the module being used is the XBee ZB Wire Antenna.

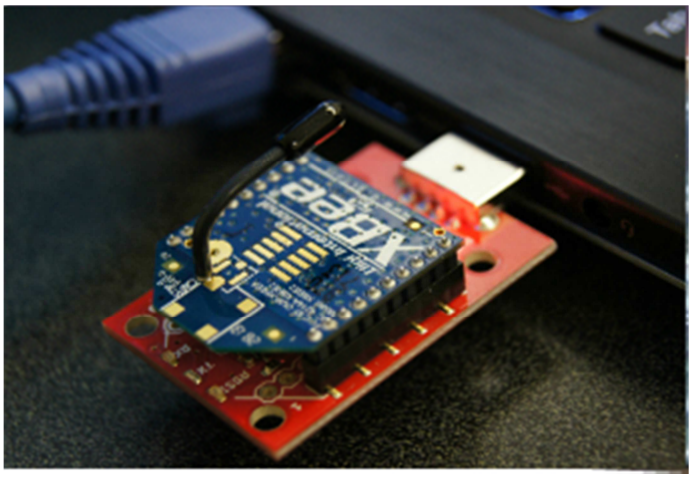


Figure 22: XBee ZB Module

Power Interface

Charging Circuit

Constant current trickle charging is used to charge the NiMH batteries in the trigger. According to research, NiMH batteries need to be charged at constant current with trickle or fast charging. For the system, the team decided to design a trickle charging circuit with a rate of one tenth of the total capacity (C/10) [36]. The reason for using trickle charging is that it is safer and does not affect the lifespan of the batteries as much as fast charging. Among the various methods of charging termination, the team decided to use analog circuitry consisting of a comparator to compare the battery voltage with a reference voltage and stop charging when the battery voltage exceeds the set reference voltage [37]. This method is easier to implement compared to the temperature sensing and the rate of change of voltage sensing methods

although not as accurate. However, since trickle charging would be charging the batteries with only a small amount of current, it is not required that the charging terminates accurately. In the design of the charging circuit, two current regulators, providing 90mA and 10mA respectively, are used. After the voltage of the batteries exceeds the reference voltage, the 90mA current regulator will be turned off but the 10mA current regulator will continue to charge the batteries to prevent the batteries from self-discharge. The IXCP10M45S current regulators are chosen to provide 10mA and 90mA of constant current to the battery. This module can provide an adjustable current of up to 100mA by choosing an appropriate resistor value.

Voltage Regulators

Multiple voltage regulators are used in the Box and Trigger to adjust the voltage levels between the battery and the individual components. For the Box, two switched-mode step down voltage regulators were used, also known as buck regulators. They have outputs of 3.3V and 5V each. The switched-mode regulators convert the voltage levels effectively by having an internal switch with a charge storage component such as an inductor or capacitor [38]. The internal switch is set at a certain frequency, allowing the switching between the power source and charge storage component to provide a constant voltage at the output. Therefore, the switched-mode regulators are preferable in battery-operated devices to reduce power consumption. The voltage regulator modules chosen for the Box are ADP1111-3.3 and ADP1111-5. These two modules are the same except for different resistor and inductor values that allow them to have outputs of 3.3V and 5V. These ADP1111 modules are chosen because they come with minimal external components and are a reasonable price. The 3.3V output regulator will be used to power the microcontroller and XBee modules while the 5V output regulator will provide power to the alarm system.

Figure 23: ADP1111 voltage regulator [39]

For the trigger, the team decided to use a linear voltage regulator with a low dropout voltage suitable for battery powered applications. A switched regulator was not used in the trigger since it uses more space, specifically its inductor. Instead, a L4391 linear regulator with a low drop voltage of 0.4V and 250mA current will be used.

Figure 24: L4931 voltage regulator [40]

NFC

In order to associate a borrowed Trigger with the student checking it out, there needs to be a built in inventory tracking system in the Dispenser. To track incoming and outgoing Triggers, RFID tags would be used inside the Triggers, which the Dispenser could read using an NFC reader module. The tags would simply need to contain the Trigger ID number, which would then be updated in the campus' global database maintained by the Coordinator. The Mifare RFID RC522 Reader IC and Antenna module was chosen to read the tags inside the Triggers. The tags used for the prototype were the ones included for testing with the module. This unit offers benefits like communication via an SPI interface and there are existing user created libraries available to use from the Internet.

Figure 25: Mifare RFID RC522 Reader IC and Antenna module [41]

GPS

The considerations made for choosing a GPS module were price, size, update rate, power requirement, number of channels, antenna types, accuracy and compatibility. There are many GPS modules that met the requirements of our system. The following is the table for the comparison of GPS modules.

Table 4: GPS Comparisons

GPS module	Antenna	Power	# Ch	Accuracy	Update Rate	Size	Sensitivity	Interface type	Cost
Maestro A2035-H	Yes	40 mA @ 3.3V	48	<2.5 m	1 Hz	16.5 x 30.5 x 30.5 mm	-148 dB	UART SPI	$20.66
Maestro A2200-A	No	41 mA @ 3.3V	48	<2.5 m	1 Hz	14 x 10.2 x 2.5 mm	-163 dB	UART SPI	$13.59
Linx RXM-GPS-SG-B	No	46 mA @ 3.3V	20	10 m	1 Hz	15 x 13 x 2.2 mm	-159 dB	Serial	$25.68
Linx R4 series	No	33-56 mA @ 3.3V	48	-	-	-	-160 dB	Serial	$21.48
Maestro A2100-A/B	No	47 mW @ 3.3 or 1.8V	48	<2.5 m	-	15.2 x 15.2 x 2.4 mm	-160 dB	UART SPI I2C	$19.44
Maestro A2235-H	Yes	31 mA @ 3.3V	48	2.5 m	.2 Hz	17.8 x 16.5 x 7.1 mm	-148 dB	I2C UART	$14.42
Skylab SKM53	Yes	45 mA @ 3.3V	66	3 m	1 Hz	30 x 20 x 8.5 mm	-165 dB	Serial	$19.99
Ultimate GPS	Yes	20 mA @ 50V	66	3 m	10 Hz	-	-165 dB	-	$39.95
Copernicus II DIP module	No	43 mA @ 3.3V	12	5 m	1 Hz	19 x 19 x 2.54 mm	-160 dB	Serial	$64.95

All the GPS modules listed above are similar in terms of size, power consumption, accuracy and update rate. Some GPS modules come with internal built-in antennas while others need external antennas to receive data from GPS satellites. The accuracy of the position depends on the number of channels: the more channels there are the better accuracy. As for the interface types, some modules provide two or more types of interface such as serial, UART and I2C and some provide only one kind of interface. One important thing that was considered in choosing the GPS module is the compatibility of the module with our prototype. The team looked for the GPS modules with UART and SPI interfaces in order to be compatible with the microcontroller used for this project. Since most of the GPS modules on the market are surface mounted, the team looked for a GPS module with a breakout board for easy implementation and some modules that we found are very expensive compared to others. However, the Skylab SKM53 is considered a good choice since it comes with a breakout board at a reasonable price. Therefore, the team decided to use Skylab SKM53 GPS module in the system.

Figure 26: Skylab SKM53 GPS module [42]

Microcontroller

Given the concept of the Trigger and the Box, as always-on devices with only a few repetitive tasks to run, both devices are well suited as embedded computing applications. A microcontroller can best serve the needs of these devices, given their simplicity of implementation and low level hardware access. In particular, when data is being sent at high speed through a relay network, the microcontroller will be necessary to help route the information.

When working on the predecessor to this project, the team had looked into the least expensive options possible, particularly ones that would be easy to program. What the team found was that TI's Launchpad offered the basic computing ability that was needed at the time while remaining incredibly inexpensive. This is another potential option as the team moved forward with the project, as the MSP430G2xx series could potentially meet our needs. One of the biggest hurdles to this series is if there needs to be more peripherals than the chip can handle.

Another option, however, is the G2xx Launchpad's new brother, the F5529 Launchpad. This new, more powerful option is still incredibly affordable but offers greater peripheral capability, higher clock speeds, and more memory, both volatile and for code. This option has been discussed somewhat, with the primary drawback being that the chip cannot easily removed from the Launchpad board to be placed in the design.

However, for this project the microcontroller chosen was the MSP430G2553 in both the Box and the Trigger. One of the main reasons that this microcontroller was chosen is that it is a very low power chip, and this feature is appealing for use in battery operated devices. Based on the requirements for the Trigger, this microcontroller has enough hardware to support the different devices it the Trigger. Another reason this microcontroller was chosen was because it is very inexpensive, and the chip is able to be removed, and can easily be placed onto a breadboard for testing.

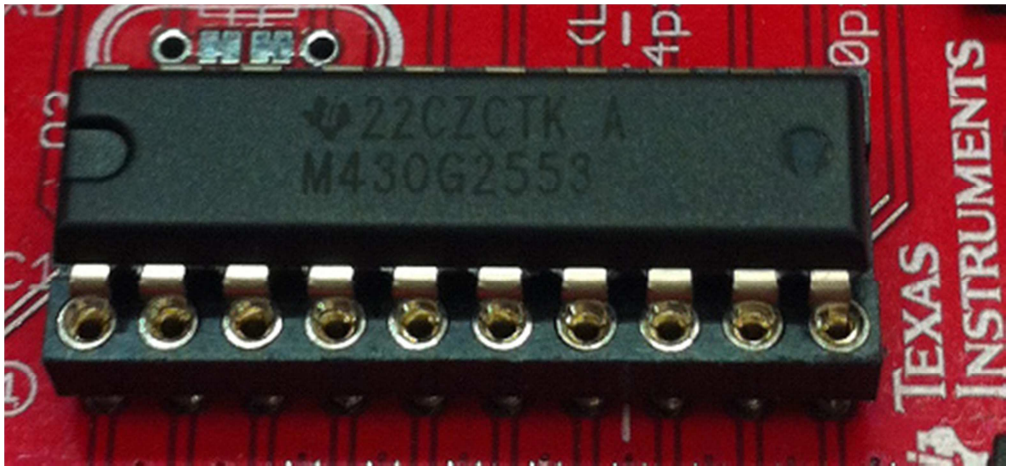


Figure 27: MSP430G2553 Microcontroller [43]

Raspberry Pi

While the MSP430 was well suited for the Trigger and Box, the Dispenser requires additional features not built into that microcontroller. Specifically, the Dispenser needs a more traditional network interface for a Cat5 connection so that it can send inventory information to the campus' global database. From this requirement, the Raspberry Pi emerged as a candidate and eventual champion, as it offers not only an Ethernet port but also two USB ports and 26 pins for general I/O, I2C, UART, and SPI communication. This allows for the use of USB-based modules, particularly a barcode scanner and a number pad for this project. The 26 pins would allow for a number of other modules to be used, including the NFC module via SPI and a small LCD display. The various types of connections allow for the simplicity of input setup. Additionally, the Pi being Linux-based allows for more libraries to be used and vaster Internet resources in development.

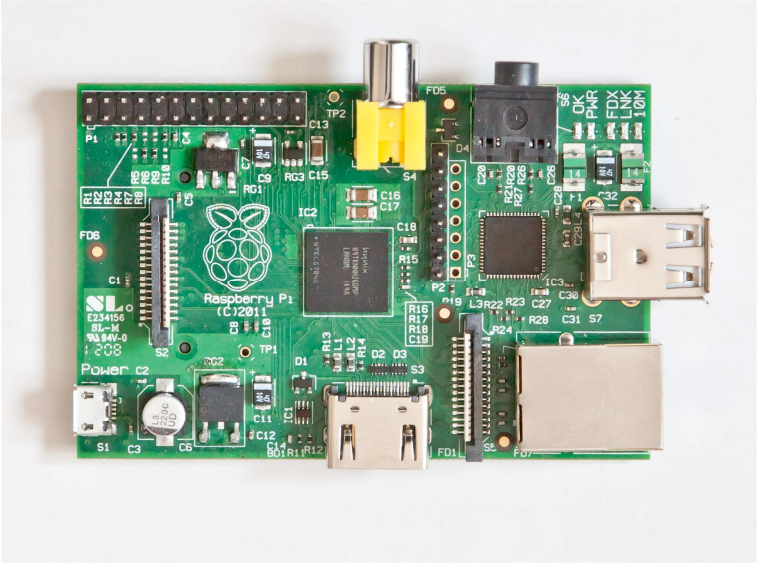


Figure 28: Raspberry Pi [44]

Alarm System

As a group, it was decided that the project would serve as a means of getting users out of danger rather than simply trying to catch the perpetrator. For this reason, a flashy and noticeable alarm system is preferable for calling attention to the emergency situation. Blue LEDs can be seen on current Blue Light Emergency Towers, and for this reason the project also incorporates them into the design. The LEDs of the prototype are from a dismantled set of blue LED Christmas lights. Each LED was rated 3.4V, 0.068W, and 0.02Amps. A siren is also part of this design, which plays into the requirement of a noticeable alarm. The group chose the 539-PS-953Q buzzer from Mallory Sonalert after noting that it can operate at 5V and listening to the sample sound clip.

Schematics

A detailed schematic was developed for each module in the Box and the Trigger as shown in the following figures. The Trigger is powered by four 1.2V rechargeable batteries for all the modules. The schematic for the Trigger in Figure 61 shows all the modules except the power interface and charging. The microcontroller is the central part of the Trigger and controls the functionalities of other modules. As can be seen in Figure 61 in Appendix B, the Trigger consists of an MSP430G2553 Launchpad, SKM53 GPS module, and XBee ZB. The microcontroller has 20 pins consisting of general I/Os and pins supporting UART.

The GPS module consists of 6 pins but only 4 pins are used for the operation in the system. As can be seen in the diagram, Pin 1 is connected to Vcc and pin 2 is connected to ground. Pin 3 is left open and pin 4 is the module reset which can be used to reset the GPS module to find the satellites. This pin will not be used for the prototype since it is unlikely that the GPS needs to be reset. The last two pins, pin 5 and pin 6 are the transmit and receive pins for the UART interface with the microcontroller. In an emergency situation, the GPS data will be sent to the microcontroller through UART communication.

The XBee module consists of 20 pins but only 6 pins are used for the system, leaving many of the I/O pins are unused. Pin 1 is connected to Vcc and pin 10 is connected to ground. The XBee module supports data transfer with the microcontroller through UART communication. Pin 9 which is the “sleep control line” is connected to one of the I/O pins of the microcontroller to wake up the XBee’s end point in the Trigger. The XBee in the Trigger is configured to be an end point and the end point goes to sleep mode when there is no data transfer. Therefore, the “sleep control line” pin is used to wake up the end point in the emergency situation. Lastly, pin 9 “clear-to-send flow control” is also connected to an I/O pin of the microcontroller to signal the microcontroller to stop sending data when the serial transmit buffer of the XBee is full. When the button is pressed on the Trigger, the “request to send flow control” pin, which is pin 16, is used to allow microcontroller to tell the module to not send data. This is used to allow the GPS to send data to the microcontroller.

The schematic for the Box is much simpler. The schematic for the Box can be seen in Figure 59. It consists of only the microcontroller and the XBee module configured as a router to route the data from the end point to the coordinator. The XBee communicates with the microcontroller through UART interface. The connections between the XBee and microcontroller are the same as in the Trigger except that the “sleep control line” is not used since the router

does not go into sleep mode.

The schematic for the charging circuit is shown in Figure 29. As shown in the figure, it is powered by 9V DC. The UA741 op-amp is used as a comparator to compare the voltage of the batteries to a reference voltage. The maximum voltage of the batteries when they are fully charged is 5.6V. Thus, the reference voltage is chosen to be 5.4V to make sure that the circuit does not overcharge the batteries. At the output of the comparator, N-channel and P-channel MOSFETs are connected to turn on or off the 90mA current regulator. When the voltage of the batteries is lower than the reference voltage, the output of the comparator will be high. This turns on the N-channel MOSFET, which also turns on the P-channel MOSFET. When the P-channel MOSFET is on, the 90mA current regulator will be on, charging the batteries in parallel with the 10mA current regulator. In this condition, the batteries will be charged with a total of 100mA. When the voltage of the batteries is greater than 5.4V, the MOSFETs will be off, thereby shutting off the 90mA current regulator.

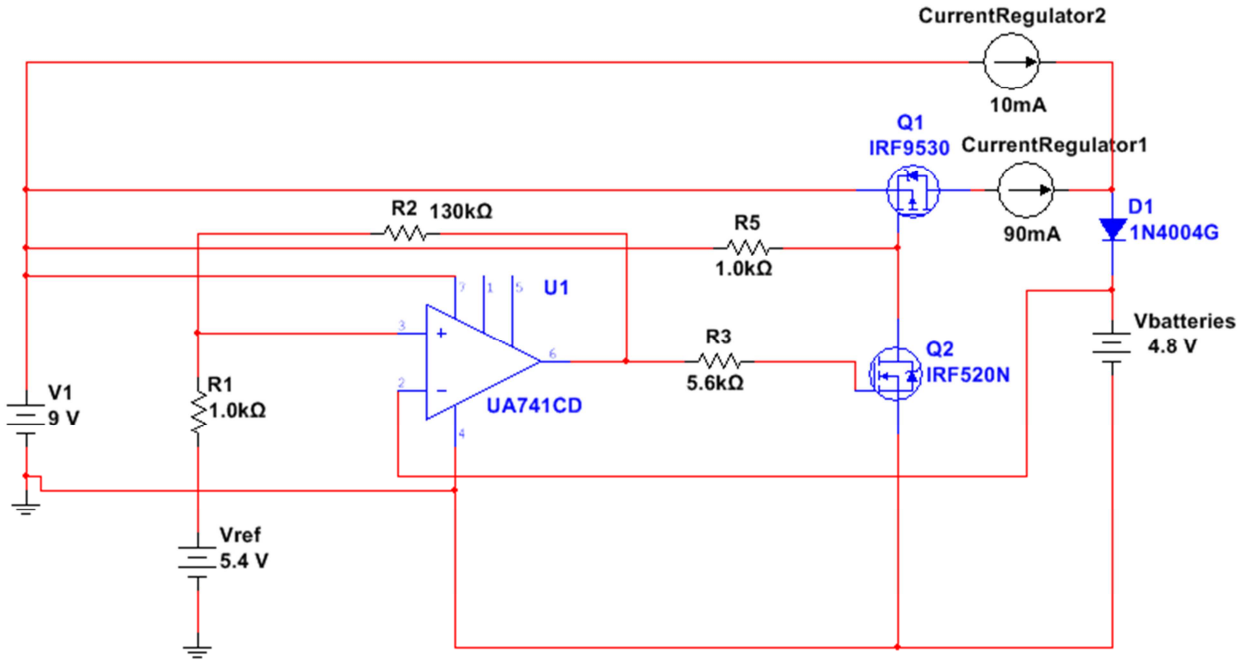


Figure 29: Schematic for Charging System

The schematic for the alarm system can also be seen in Figure 30. To control whether the alarm system becomes activated, a P-channel MOSFET IRF9530 is used as a switch controlled by the microcontroller. The input from the microcontroller is wired to the gate of an IRF9530, which in turn has its drain connected to a LM555 timer and its source connected to a voltage regulator that provides 5VDC from the box's battery. The siren's positive lead is also connected to the drain of the first IRF9530. Its negative lead is connected to ground. Most of the customized design for the alarm system can be seen in its LED layout. A 555 timer is used in the circuit to allow for the LEDs to flash in an alternating pattern. A total of 16 LEDs are used, divided into two groups of 8. A P-channel and an N-channel MOSFETs are used as switches for each group to enable flashing between the two. In default state or in nonemergency situations, one group of LEDs will be on without flashing. Each LED is first connected in series with a

resistor to protect against overcurrent. For each group of 8, the positive leads are connected together. Group one's positive leads are then connected directly to the voltage regulator providing 5V. Group two's positive leads are connected to the drain of an IRF9530, which has its source connected to the voltage regulator and its gate connected to the 555's output. Group one's negative leads are individually connected to 100Ω resistors and those resistors are then connected to the drain of an IRF520, which has its source connected to ground and its gate connected to the 555's output. Group two's negative leads are also each connected to 100Ω resistors and these resistors were then connected to ground. Essentially the LEDS of each group are in parallel. The groups are connected in this fashion so that each group will light up together, and the group that is lit will alternate depending on the output from the 555 timer. This is because the timer uses a square wave that alternates between high and low input.

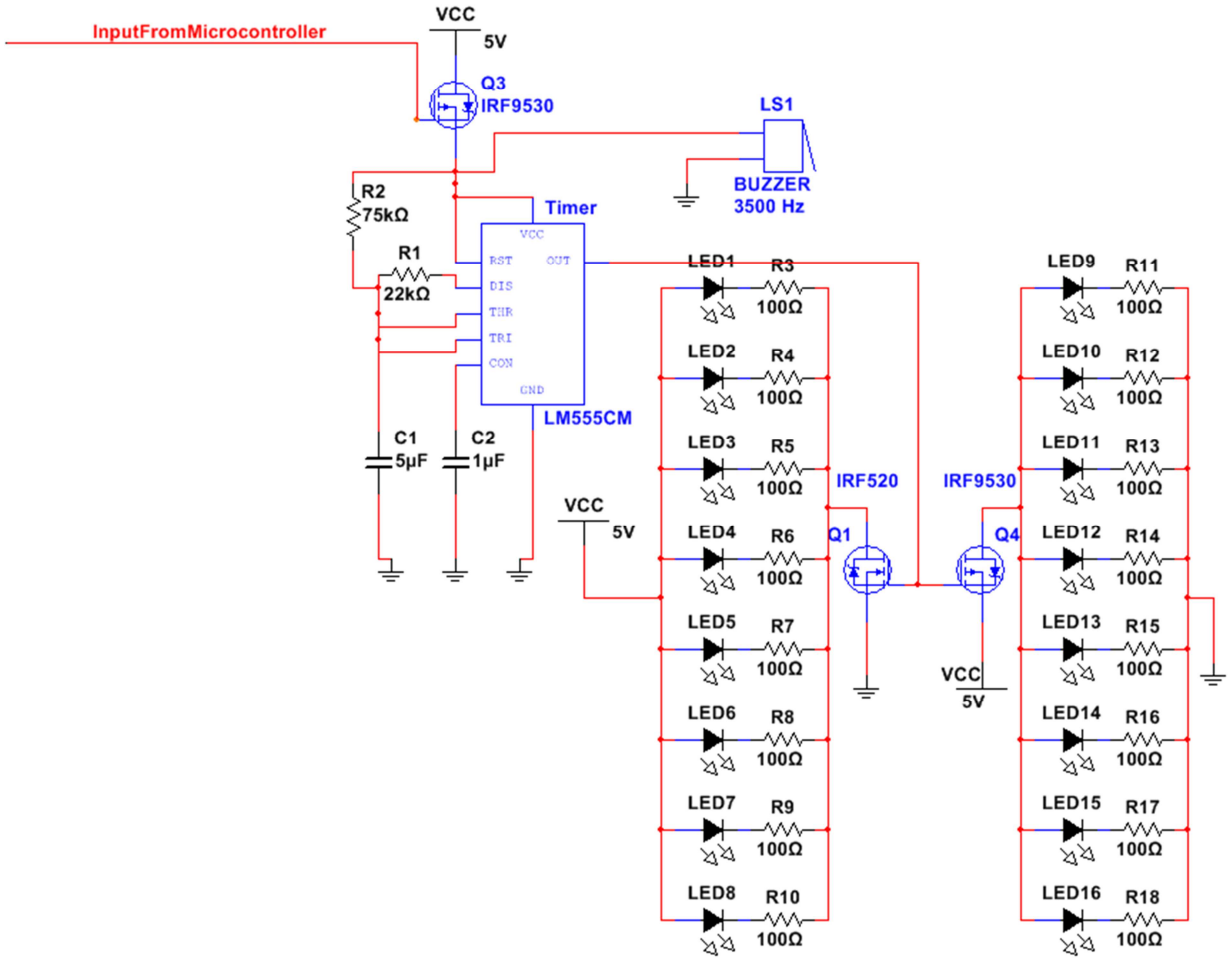


Figure 30: Schematic for Alarm System

Prototype Cost Analysis

The cost of our system is determined based on the prototype cost which includes considerations such as production cost and transportation cost. The complete system of the product consists of a network of Boxes, Dispensers and Triggers that are connected to a server that controls the operations of our system. The following tables show the breakdown cost of three units of the system according to their component costs.

Table 5: Cost Breakdown of the Prototype

Trigger		**Box**		**Dispenser**	
Component	Cost	Component	Cost	Component	Cost
Microcontroller	$9.99	Microcontroller	$9.99	Raspberry Pi	$35
XBee ZB	$17	XBee ZB	$17	NFC	$6.99
GPS	$25	Siren	$9.40	LCD Display	$5.17
Batteries (x10)	$24	LEDs	Free	Barcode Scanner	On Loan
		Battery	$13.54		
		Solar Panel	$32.75		
Casing	Free	Casing	$10.99	Casing	$10.99
Miscellaneous	~$15	Miscellaneous	~$30	Miscellaneous	~$20
Total	**$90.99**	**Total**	**$123.67**	**Max Total**	**$78.15**

The cost of the system depends on the size of the infrastructure. Since the infrastructure of the system is flexible, the cost will vary depending on the number of Triggers, Boxes and Dispensers used in the system. Using WPI campus as a sample site for the system, the group calculated the total cost of the system by having 15 dispensers, 80 boxes and 300 triggers. These numbers are based on the number of the buildings on campus, the emergency towers and the student body. The group plans to attach the boxes to the emergency towers on campus as an improvement to the towers. However, the system is also easily expandable to off campus areas since the Boxes are completely modular and wireless. The following table shows the potential cost of the system for WPI campus as a beta site.

Table 6: Cost of the System for WPI Campus

Unit	Cost
Trigger	$90.99 x 300
Box	$123.67 x 80
Dispenser	$78.15 x 15
Total Cost	$38,362.85

Power Estimate

As the group has chosen a number of the components to be used in the prototype, a preliminary power budget is incredibly useful when looking at energy storage and alternative energy sources such as solar. The breakdown of the power used by each module can be seen in Table 6 below. These are based on maximum values for both current and voltage, with all modules in active mode. Most of the time the modules will be in low power modes.

Table 7: Power Consumption Considerations

Component	Power Consumption
XBee ZB	40 mA @ 3.3V = 132 mW
GPS	150 mA @ 5V = 750 mW
Microcontroller	360 µA @ 3V = 1.08 mW
Siren	150 mA @ 5V = 750 mW
Light Array	312 mA @ 5V = 1.56 W
NFC	100 mA @ 3.3V = 330 mW
LCD Screen	120 mA @ 5V = 600 mW
Barcode Scanner	130 mA @ 5V = 650 mW

The breakdown of the three main system components, the Trigger, Box, and Dispenser, are shown in the smaller tables below. They do not all need every module listed above, and a sum of the maximum power in each is listed.

Table 8: Module Breakdown within Primary Devices

Box
XBeeZB
MCU
Siren
Light Array
Total = 2.44 W

Trigger
XBee ZB
GPS
MCU
Total = 883 mW

Dispenser
Raspberry Pi
NFC
Screen
Barcode Scanner
Total = 6.58 W

Fortunately, almost any solar option should provide plenty of power for the Box, which requires very little. The Dispenser is of little concern, as it will draw power from the grid. The main concern from a design view will be the Trigger, as it has a small form factor and would preferably be lightweight, but will clearly need a beefy battery to handle the power consumption necessary in Emergency Mode. Additionally, the charging from the Dispenser to the Trigger needs to be robust enough to have them charged when users need one.

Chapter 4: Methodology and Implementation

This chapter explains the process the group used to test individual modules or devices for this project. The first part of this chapter is explaining the different testing procedures for each module in this project. Next, some modules were tested together to emulate different aspects of the main devices in this project. Lastly, the testing procedures for the devices in the system were described.

Microcontroller UART

One of the main types of communication between devices for this project was UART. This type of communication was used in both the Trigger and the Box. The first step to begin this project was to determine that the UART was working correctly for the microcontroller. The reason that this one of the first tests done was that the microcontroller would have to communicate with the XBee and the GPS. If this functionality did not work, then there would be no way to determine if the microcontroller was communicating with each device correctly. The way that these two devices were connected together was by connecting the transmit line of one MSP430 to the receive line of the other line. The same would be done for the receive line of the first MSP430. The last part before testing was done was to load the simple test program onto both microcontrollers. This sample program can be found in Appendix C.

Using sample code and the user guide for the MSP430, a simple test program was written that included headers for MSP430 libraries, a main function, and definitions for interrupt service routines (ISR) that would control the communication functions. The main function of the program contained the necessary configuration to connect the MSP430 as one side of a UART link, including setting the proper pins to use their secondary UART functionality, choosing which clock is used and the baud rate for the connection, and initializing the state machine. Finally the interrupts were enabled. Additionally, the button on the Launchpad was configured as an input and the LED was configured as an output. Finally, the controller was put into a lower power state to run on interrupts.

The ISRs were written to describe the behavior desired for transmitting and receiving. The transmit ISR was designed to send the status of the button as an 8-bit character. The receive ISR was designed to test the character read into the receive buffer to determine if the button status transmitted was a ‘pressed’ state. If it was, it changed the output of the LED from on to off. If the received character was anything else, it turned the LED on. The program to do this was loaded onto both MCU’s so that if either one’s button was pressed, it would shut the light off on the other.

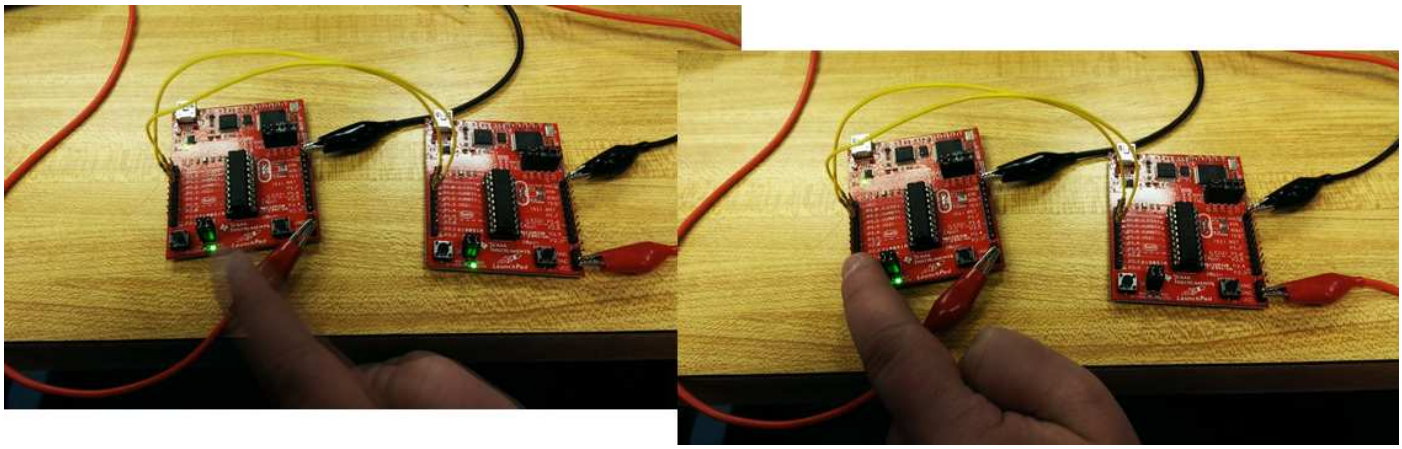


Figure 31: MCU UART Testing

XBee Communication Testing

To determine that the XBee's were working correctly, an XBee network had to be made for the devices to connect. This was done to determine that the different XBee modules were configured correctly by the X-CTU software, and that they were found in the network. If an XBee device was not found in the network, then there could have been problems testing the microcontroller with the XBee. One of the problems that could have occurred with the XBee's would be that the device is receiving data correctly from the microcontroller, but the XBee is not configured correctly in the network. The first step was to configure both XBee's in API mode. This was done by placing the XBee into the USB breakout board and loading the correct firmware onto the XBee. Depending on the device in the network, each XBee was configured with a different version of the API firmware. The next step was to make sure that all of the XBee's in the network had the same PAN ID and SC value. This was to ensure that all of the XBee's would be in the same network, and have the same network search configuration. The reason that this step had to be done was to ensure that the devices would be able to communicate with each other.

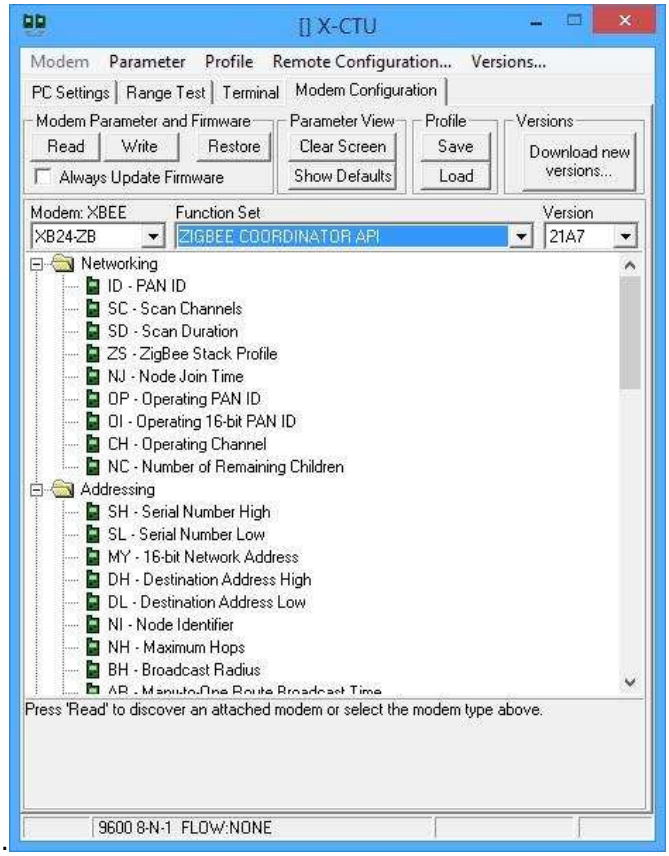


Figure 32: X-CTU Software

A sample API packet from the User Guide for the XBee ZB was used for this test. This packet sent a payload over the XBee network to the coordinator using the default address for the coordinator. One reason that a predetermined packet was used was to make sure that the packet would not be rejected from the XBee. When any part of the packet was not following the given standards, then the packet was rejected and no information was sent. The packet used is shown below:

0x7E 0x00 0x16 0x10 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFE 0x00 0x00 0x54 0x78 0x32 0x43 0x6F 0x6F 0xFC

The transmit pin, pin P1.2, on the empty Launchpad board, which is a Launchpad without a microcontroller on it, was connected to the receive pin on the XBee, which is pin 3. Then the XBee that was setup as the coordinator was placed into the USB breakout board. This USB board was plugged into the computer. Next, a terminal was opened for the empty Launchpad and for the XBee plugged into the computer. The XBee that was connected to the empty Launchpad was provided with power from a power supply. Using the terminal program, the predetermined packet was sent from the XBee on the breakout board to the XBee configured as a router using the empty Launchpad. Both sides of this communication were visible in terminals on the PC.

GPS

The GPS module was tested on breadboard first to explore its functionality. The purpose of this test was to make sure that the GPS module received data from the satellites and was transmitting the location data correctly. The GPS module was set up on breadboard with pull up resistors at the transmit and receive pins, and bypass capacitors at the power rails.

First, the GPS module was powered with power supply at 3.3V and 150 mA. The GPS was tested in a clear view of the sky and the output waveform of GPS at the TX pin was checked on the oscilloscope. The output (TX pin) of the GPS was connected to the computer using an empty Launchpad. The GPS data was displayed in a terminal on a PC and made sure that the correct NMEA strings were received. The GPS Locator Utility program, which is software that can translate GPS data into a readable format, was used to translate GPS data and made sure that the GPS module showed correct time, latitude and longitude.

Microcontroller and XBee

After both the XBee and the microcontroller were confirmed to have basic functionality working, the next step was to combine these two modules. The main reason that these modules were tested separately was so that we could test our functions to create XBee API packets. If these functions did not work, then there could have been an issue with sending the GPS data over the XBee network.

First, the receive pin on the Launchpad with the microcontroller was connected, P1.1, to the transmit pin on the empty Launchpad board, which is P1.2. Then the transmit pin from the microcontroller, P1.2, was connected to the receive pin on the XBee which is pin 3. A picture of this connection is shown below.

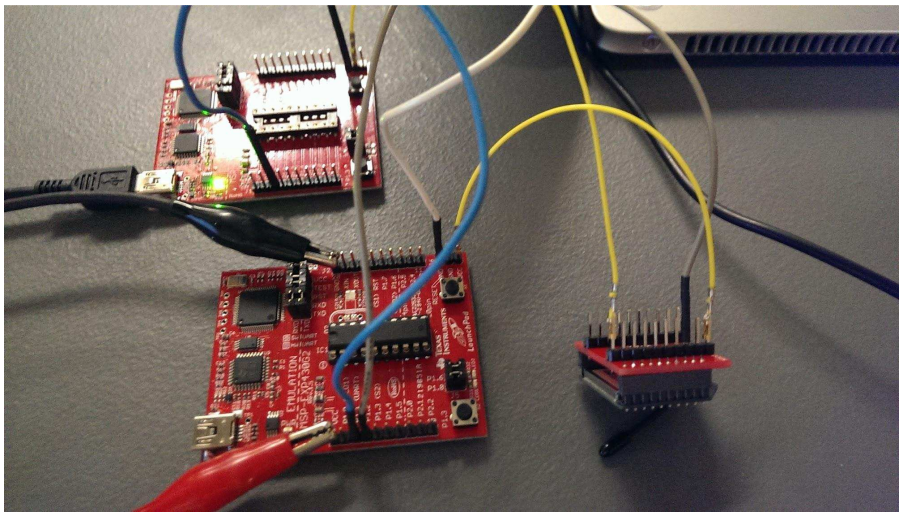


Figure 33: Connection between XBee and MCU

All of the grounds for the devices were connected together. The program in Appendix C.2 was loaded onto the microcontroller. This program waited for eight characters from the empty Launchpad before it sent the API packet over the XBee network. Both the empty Launchpad and the XBee coordinator plugged into the USB board were plugged into the computer. The XBee and the microcontroller that are not connected directly to the computer

were powered by a separate power supply. In the terminal window for the empty Launchpad, eight characters were typed into the window. Then the coordinator received the packet on the display.

Microcontroller and GPS

One last test that needed to be done before the XBee, GPS, and the microcontroller could be combined together was to test the GPS and microcontroller. If the GPS was not able to communicate with the microcontroller, then there would be an issue where the police would not receive any GPS data from the incident.

First the transmit pin of the GPS (pin 5) was connected to the receive pin on the Launchpad with the microcontroller on it, P1.1. The Launchpad was configured to echo back the data that it is receiving from GPS (i.e. at the transmit pin, P1.2 of the Launchpad, the data that is received is transmitted). The output of the microcontroller was checked by comparing it to the output of the GPS. This was done by connecting both the transmit pin, P1.2 of the Launchpad and the transmit pin, pin 5, of the GPS to the oscilloscope. Then the transmit pin, P1.2 of the Launchpad was connected to the transmit pin of the empty Launchpad, P1.2. The empty Launchpad was connected to the computer through the USB port. Then the data from the microcontroller was displayed on the screen.

Microcontroller and GPS and XBee

The last main test before the Trigger was made was to test the microcontroller, GPS, and the XBee together. This test was to confirm the ability of the microcontroller to receive GPS data and to create an XBee packet to send over the Zigbee network. The microcontroller needed to be able to take the raw GPS data and format the data into an XBee packet so that the coordinator XBee received the latitude and longitude.

The program in Appendix C.4 was loaded onto the microcontroller. The microcontroller, XBee, and GPS were connected on a breadboard following the picture shown below.

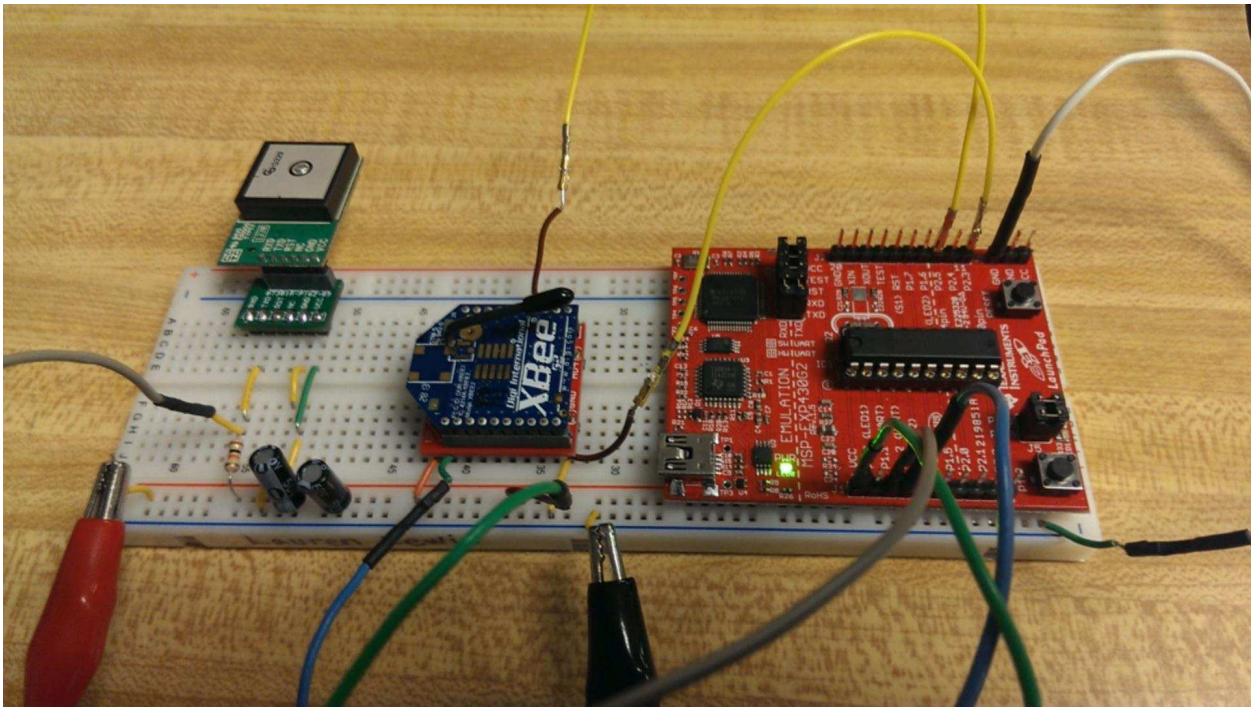


Figure 34: Breadboard for Microcontroller, XBee, and GPS testing

After all of the devices were connected correctly, the entire circuit was provided 3.3V and about 0.76A from the power supply. The common ground and the power were connected from the power supply to the circuit. The coordinator XBee was placed in the USB breakout board and plugged into the computer. A terminal on the computer was opened to interact with the coordinator XBee, and the power supply was turned on to provide power for the breadboard.

Alarm System

Dismantling the Christmas lights was the first step in constructing the alarm system. The wire connecting the Christmas LEDs was clipped midway between each pair of LEDs. This was meant to keep the lead lengths equivalent for each LED and also allow for the longest possible lead length. This process was repeated until there were 16 individual LEDs. From here, the leads of each LED were stripped approximately a half inch and lengthening wires were attached to both positive and negative leads. This was done to allow for easier testing, as the wired used are sized to fit in a breadboard.

To determine the ideal voltage for the LEDs, an LED was connected directly to a power supply and gradually increased voltage. Voltage began at 2.5V and was increased up to 3.4V to observe how bright the LED could shine.

A 510Ω resistor was connected to the negative LED lead. The other end of the resistor and the positive LED lead were then connected to the power supply. The voltage on the power supply was set to 2.6V and increased in 0.5V increments until the LED flickered out. This was meant to test how much voltage an LED could handle.

To gain a better understanding of how the LEDs would act in the circuit, the next step was to connect and power multiple LEDs. The negative leads of the LEDs were each connected in series to 510Ω resistors. The free leads of the resistors were connected, as were the positive

LED leads. This ultimately put the LEDs in parallel and they were then connected to the power supply. The power supply voltage was initially set to 2.5V and was increased in increments of 0.5V up to 6.0V.

Through Ohm's law, it is known that a decrease in resistance will cause an increase in current. It also known that the LEDs are rated at 20mA. For this reason, the next logical step was to try lowering resistance by putting a single LED in series with a 300Ω resistor and then a 100Ω resistor. First the 300Ω resistor was attached to the LEDs negative lead and 5V was provided by the power supply. The voltage across the resistor and LED were both recorded and the current could then be calculated. This same procedure was repeated for the 100Ω resistor.

To determine how loud the siren would be, an android app called Sound Meter was used to measure the number of decibels both up close and across the room from the siren. The siren was first tested at 5V both up close and across the room. This procedure was then repeated for 12V to observe if any change in decibel level occurred with a change in voltage.

Two LEDs were configured with an IRF520 and IRF9530. The first LED had its positive lead connected to the power supply and negative lead connected to a 100Ω resistor, which was then connected to the drain of the IRF520. The IRF520's source was connected to ground and its gate was connected to the gate of the IRF9530. The IRF9530's source was connected to the power supply and its drain to the positive lead of the second LED. That LED's negative lead was then connected to a second 100Ω resistor, which in turn was connected to ground. The gates of both MOSFETs were connected to a function generation that stood in for the LM555 timer. A square wave with a 50% duty cycle was sent out of the function generator to the circuit. This set up was implemented to ensure that the two LEDs would flash in an alternating pattern, where one was on when the output was high and the other was on when the output was low.

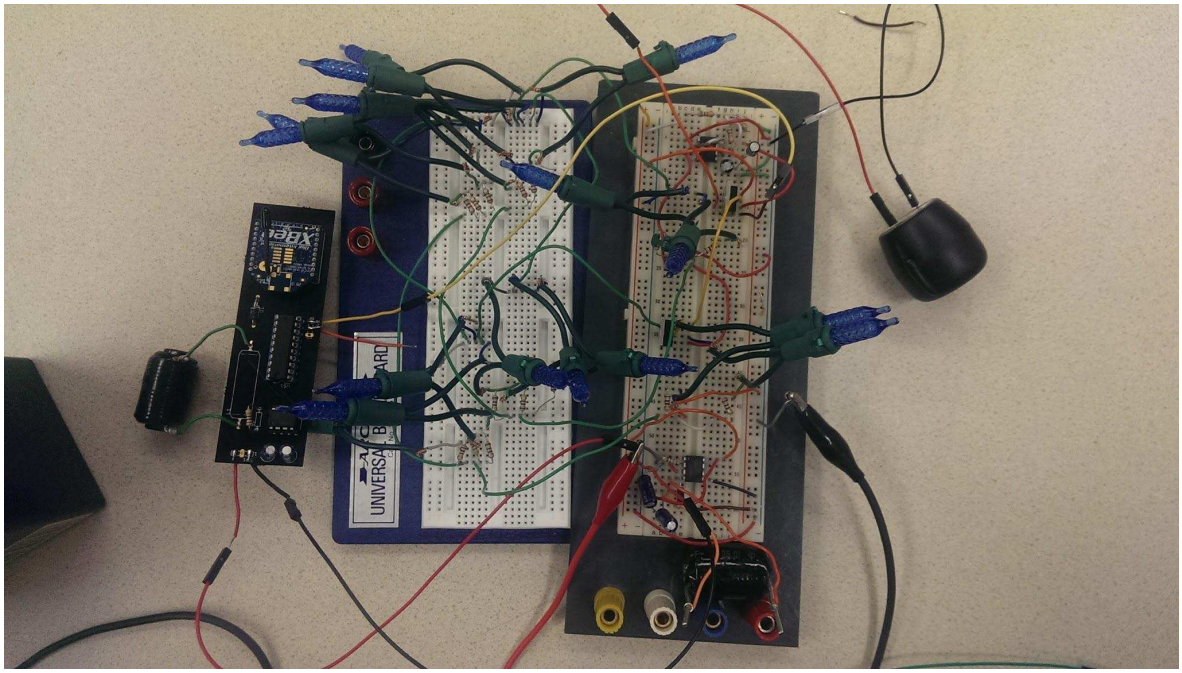


Figure 35: Alarm circuit on two breadboards

The next logical step was to test the circuit in its entirety. The LM555 timer along with the resistors and capacitors outlined in Figure 30 were all placed on a breadboard. The 16 LEDs and their 16 respective 100Ω resistors needed to be connected with the 555 timer circuit. The setup from the previous testing was kept and the remaining 14 LED and resistor pairs were added, 7 in parallel with the first LED and the remaining 7 in parallel with the second LED. The function generator input was removed and replaced with the output from the 555 timer. Several different capacitor values were tested for use with the LM555 timer. The final 555 timer circuit consisted of an LM555 timer with a 75kΩ resistor, a 22kΩ resistor, a 5µF capacitor, and a 1µF capacitor. The orientation of this circuit followed the Alarm System diagram in Figure 30. As can be seen from the diagram, it was inclusive of another IRF9530 that takes input from the microcontroller to determine when to turn the alarm on or off. This MOSFET was also connected to the power supply. Essentially, the entire circuit shown in the schematic entitled Working Schematic for the Alarm System was constructed across two breadboards with the exception of the siren. It was constructed so that each pair of 8 LEDs would flash in an alternating pattern when power was given to the circuit. This circuit was tested with the power supply set to 5V.

A piece of old tupperware was used to house the 16 LEDs. On each of the four sides of the container, four holes were created in a square pattern that allowed the LEDs to stick out. A slightly larger and transparent plastic bowl was selected to be the outer cover of the alarm hub. The outer cover was made slightly more opaque when it was covered with partially-transparent tape. This was done to allow the design to more closely resemble emergency alarm lights found inside campus buildings.

Once the tupperware container was prepped with 16 holes, sets of four LEDs were soldered together. Each negative lead was lengthened with a wire so that it could later be soldered to their respective resistors on the alarm PCB. Each positive lead of the 4 LEDs was soldered together with a single wire. Then the set of LEDs was placed inside the alarm hub. This was done so that lights that flash together were placed on opposite sides. Alternating pairs were placed in this pattern so that one row was filled in all the way around before the second row was inserted.

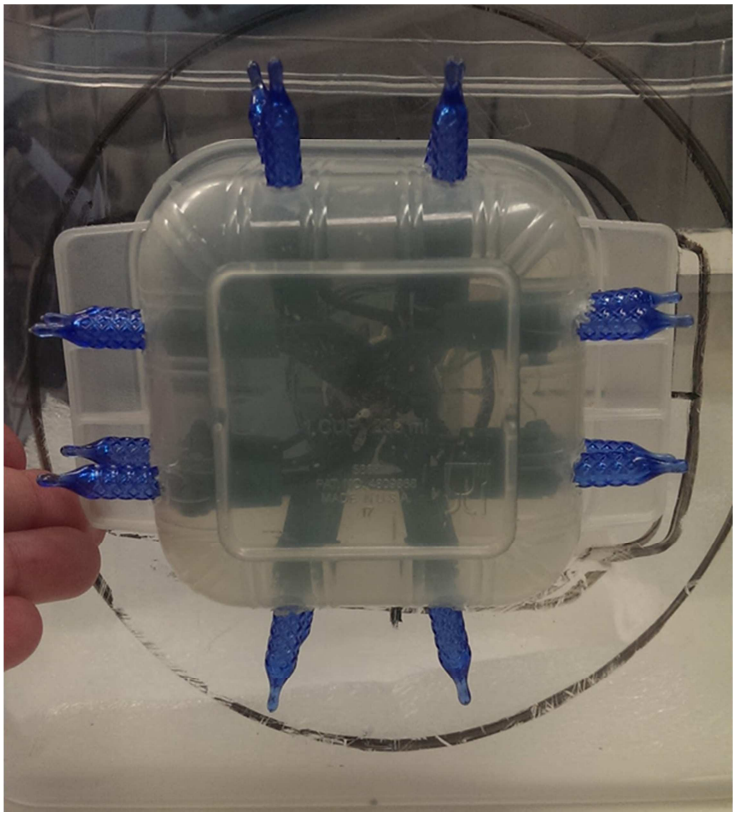


Figure 36: Alarm LEDs in housing

A generic PCB was used to hold the LM555 timer circuit and the 100Ω resistors that pair with the LEDs. These components were all soldered onto a generic PCB due to time constraints. With the alarm PCB soldered and all the LEDs carefully inserted, the lengthened wires of the LED positive leads were soldered through a hole in the box's container to the alarm PCB according to the schematic in Appendix C entitled Working Schematic for the Alarm System. Each negative LED lead was soldered to the appropriate resistor. For more detail on the hole in the box and the box's container, see the Box section of this chapter.

The final piece of the alarm circuit added was the siren. It was simply connected to ground and to the source of the IRF9530 that receives input from the microcontroller. Once the entire alarm circuit was assembled, it was tested by supplying power directly to the 555 timer circuit.

Solar Panel and Battery

Initial testing of the solar panel occurred by the window of a campus laboratory at 3:30pm EST. The solar panel was first exposed to indirect sunlight by opening the blinds and covering part of the solar panel with a notebook. That notebook was gradually moved off the solar panel until the entire panel was directly exposed to sunlight. The amount of voltage coming from the solar panel was observed. The solar panel was rated 22 V for open voltage and 0.31 A for short circuit current and this rating was compared to the measured voltage output and current flow in direct sunlight.

The solar panel came with a 6-port charge controller. The negative and positive ports on the far left were attached to the negative and positive solar panel leads. The two far right ports

on the charge controller were reserved for a load. The voltage output was measured across a 3 kΩ resistor attached as the load.

To test the Box battery, it was connected to the middle two ports on the charge controller. The solar panel leads were disconnected and the load was left attached. The voltage across the resistor was then observed to again determine the current flow.

The solar panel and Box battery were then both connected to the charge controller via their respective ports. The same resistor was used as a load. The voltage across the resistor was measured to observe any change in the current flow.

The last tests with the solar panel and battery were performed outside. First the solar panel was tested by itself. The way that this device was tested was to change the direction the panel faced in relation to the sun. The first direction of the panel was facing the sun and the second direction was facing away from the sun. Then the panel, charge controller, battery, and resistor were all tested together as before. The panel was then removed and the charge controller, battery, and resistor were tested together. Finally, the panel was replaced and the resistor was removed. These last tests with the solar panel were performed with the panel angled to face directly at the sun.

Power Interface

In order to provide power to the devices properly, various power interfaces were used to convert the voltage from the power source to each component. The voltage regulators were tested individually first to examine their functionality. There are 3 voltage regulators in our system. The two switched-mode regulators were used in the Box and a linear voltage regulator was used in the trigger.

Testing Power Interface for Trigger

The L4931 voltage regulator was used as a power interface in the Trigger between the batteries and other components in the device. The L4931 voltage regulator consists of 3 pins: the input, output and ground. After identifying the pins, a test circuit was set up as shown in Figure 14 from the datasheet. There are two bypass capacitors at the input and output pins for the purpose of filtering noises. The resistors used were 0.1uF at the input and 2.2uF at the output.

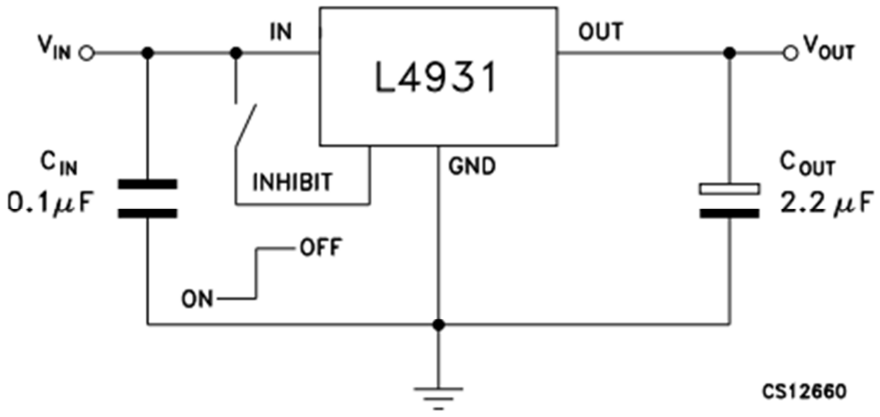


Figure 37: Test circuit for L4931 voltage regulator

After setting up the circuit, the voltage output from the power supply was connected to the input pin. Since this voltage regulator would be used with the NiMH batteries whose voltage varies from about 4.8V to 5.6V, the power supply voltage was adjusted gradually from 4.8V to 5.6V to ensure that the regulator worked at all the required voltage levels. First, the output was observed at the output pin without any load attached to it. The next step in testing this regulator was to include a load at the output pin and verify the output voltage. The output was tested with 100 ohm and 1 kilo ohm resistors one at a time and verified the output voltage. The purpose of this test was to observe the behavior of the regulator for various loads.

After verifying the functionality of the regulator with the test circuit setup, the regulator was set up in the trigger with the batteries as the power input, and the microcontroller, GPS and XBee as loads. Initially, only the battery was connected to the input of the regulator without any load and the output was observed. Next, the Vcc pins of the microcontroller, GPS and XBee were connected to the output pin of the regulator and examined if the loads were powered properly. Each of the Vcc pin of the microcontroller, GPS and XBee was checked with the multimeter. Moreover, to test if the regulator was providing enough current to the loads, the team tested the functionality of the loads. The coordinator XBee was connected to the computer through the USB port and searched for the end point XBee in the Trigger. Next, the functionality of the components in the Trigger in the emergency mode was tested by pressing the button on top of the Trigger. In this condition, the group observed whether the GPS data was sent from the Trigger to the coordinator. This test ensured the functionality of the whole power interface for the Trigger.

Testing Power Interface for the Box

The first step in testing the ADP1111-3.3 and ADP1111-5 was to set up the external components correctly. These modules required an external inductor, two capacitors, a zener diode and an optional resistor to set up the current limit. The instruction in the datasheet was followed to calculate the values of the external components. First, to determine the inductor value, the peak current was calculated using the following equation.

$I_{PEAK} = (2I_{OUT}/DC)*(V_{OUT}+V_D)/(V_{IN}-V_{SW}+V_D)$

where DC = duty cycle(0.5 for ADP1111)

V_{SW}= voltage drop across the switch

V_D= diode voltage (0.5V for 1N5818)

I_{OUT}= output current

V_{OUT} = output current

V_{IN}= minimum input voltage

The output current for ADP1111-3.3 was specified to be 50mA since the XBee consumes 40mA and the microcontroller consumes 100uA at maximum in the emergency mode. For ADP1111-5, the output current was specified to be 120mA since the LEDs when flashing used 120mA. The duty cycle for the ADP1111 was given to be 0.5(50%) in the datasheet. The output voltages for ADP1111-3.3 and ADP1111-5 regulators were 3.3V and 5V each and the diode voltage Vd was taken to be 0.5V for both regulators since the diode 1N5818 was used. The maximum input voltage was taken to be 11.5V considering the battery voltage drop when it is deeply discharged. The Vsw , voltage drop across the switch, was specified as 1.5V in the datasheet.

The inductor value was calculated using the following equation.

$L = (V_{IN(MIN)} – V_{SW} – V_{OUT})*t_{on}/I_{PEAK}$

where t_{on}= switch on time(7 µs for ADP1111)

After carrying out the calculations, the inductor value was chosen to be 100uH for the ADP1111-3.3. After the external components were specified, the voltage regulator was set up as shown in Figure 15. Pin 1, which is current limit pin, was connected to Pin 2 through a current limiting resistor whose value was chosen to be 100 ohm. Pin 2, the input pin, was connected to the 12V from the power supply for the purpose of testing. A bypass capacitor C2 was also connected from Pin 2 to ground. Pin 3, SW1 pin, was connected directly to the Vcc (Pin 2) to configure the voltage regulator in step-down(buck converter) configuration. Pin 4, SW2 pin, was connected to the 1N5818 diode and the inductor. The other end of the inductor was connected to Pin 8, the output pin. Another bypass capacitor was also connected from the Vout to ground to filter noises. For ADP1111-5 regulator, the inductor value was calculated to be 68 uH. The current limit was set up 1000 ohm resistor from pin 1 to pin 2.

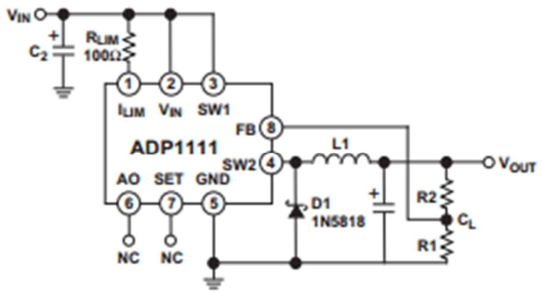


Figure 38: Test Circuit for ADP1111 Voltage Regulator

The input voltage was first given by the power supply which varied from about 11.5V to 13.5V to emulate the battery in the Box and the output voltage was observed. First, no load was given at the output. In the next step, various loads were connected at the output of the regulator to observe the behavior of the output at different load values. Next, the power supply was removed from the circuit and the 12V battery was connected to the input and the output was observed.

The next step in testing the voltage regulators was to test it with other components in the Box. The output of the ADP1111-3.3 regulator(pin 8) was connected to the Vcc of microcontroller and Vcc of XBee and observed if these components were powered properly. The output of ADP1111-5 was connected to the alarm system at the required pins. After connecting the power pins of the components to the output of the regulators, the Vcc pins of each component were checked using the multimeter to ensure they were receiving the voltage from the regulators. After that, to test the functionality of the components in the Box, a ZigBee network was set up using the coordinator and searched for the router in the Box using the X-CTU terminal program on the computer. This method utilized the coordinator in place of the endpoint XBee to send the emergency signal from the Trigger to the Box. The purpose of this test was to examine if the Box received the emergency signal and activated the alarm system.

Charging Circuit

The charging circuit consists of current regulators, MOSFET switches, and a voltage reference IC. In order to test the circuit, important individual modules were tested first before putting everything together. The NXCP10M45S current regulators, TL431 voltage reference and UA741 op-amp were tested individually.

Testing Current Regulator

Two of the NXCP10M45S current regulators were used to provide 90mA and 10mA current each. The functionality of the current regulators was an important part of the charging circuit since the ability to provide constant current was vital for charging the NiMH batteries. In order to test the functionality of this module, the three terminals were first identified as positive (A), Negative(K) and Control terminal(G) and the following circuit was set up to test the current with a resistor at the negative(K) terminal.

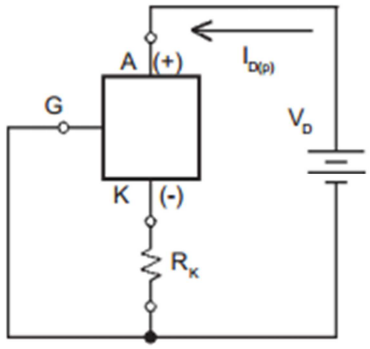


Figure 39: Testing Circuit for Current Regulator

As can been seen from Figure 39, an appropriate resistor value R(K) was chosen to obtain the desired current. For our circuit, a 300 ohm resistor was used to provide 10mA current and 20 ohm resistor was used to provide 90mA current. The resistor values were given in a graph (resistor vs current) in the datasheet. This current regulator was tested with different input voltages at the positive terminal(A) and a load resistor was connected. The voltage across the load resistor was measured at different input voltages and the output current was calculated using Ohm's Law. The purpose of this test was to compare the output current at different Vcc to determine the Vcc in the design of the charging circuit.

Testing Voltage Reference

TL431 voltage reference was used to set up the 5.4V reference voltage at the positive terminal of the comparator. It consists of 3 pins and these three terminals were identified as Cathode, Anode and Reference terminals. After identifying the pins, the following circuit was set up to test the output voltage from the IC.

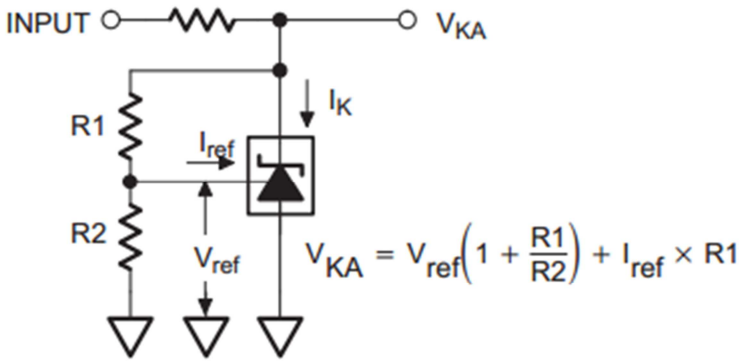


Figure 40: Test Circuit for TL431 Voltage Reference

The resistor values were set up using the Equation :

V(KA) = Vref (1+R1/R2) + (Iref*R1)

In order to get 5.4V, the resistors chosen were R1 =2 kΩ and R2 = 750Ω. The value of the resistor between the input and V(KA) was 1.9 kΩ.

Testing the Comparator

The comparator was set up as a positive feedback Schmitt trigger in an inverting configuration as shown in Figure 18 . It was designed to have a hysteresis of about 100mV to minimize noise and prevent the circuit from being unstable. The following equations were used to calculate the high and low threshold voltages.

$V_{TH} = (R1/(R1+R2))*V_{SAT}^{+} + (R2/(R1+R2))*V_{REF}$

$V_{TL} = (R1/(R1+R2))*V_{SAT}^{-} + (R2/(R1+R2))*V_{REF}$

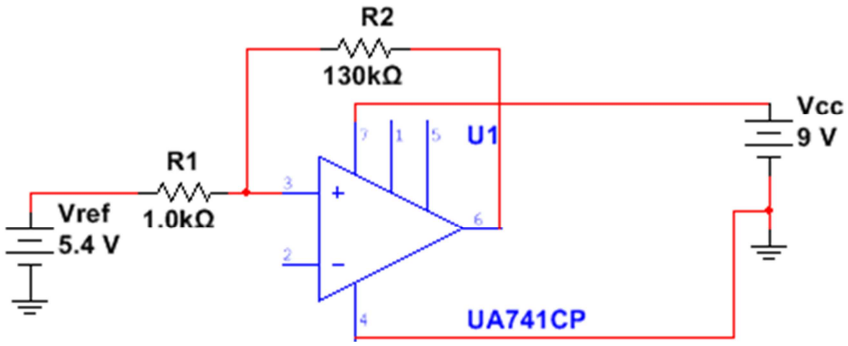


Figure 41: Test circuit for Comparator

By solving the two equations simultaneously, the two resistor values were calculated to be 1 kilo ohm and 130 kilo ohm. By using these resistor values, it was expected that the high threshold voltage would be 5.43V and the low threshold voltage would be 5.35V. The comparator was tested by applying varying voltage at the negative input terminal of the op-amp. The output voltage was measured at pin 6 to confirm that the op-amp worked as expected.

After testing the individual components, all the components in the circuit were assembled on the breadboard. Initially, a resistor of 10 ohm was used in place of the batteries since it was easier to measure the voltage across the resistor. Another reason for not having the batteries in the early stage of testing was to prevent damage to the batteries in case the circuit was not working properly. During this stage, a varying voltage from the power supply was given to the negative terminal of the comparator to compare with the reference voltage of 5.4V. When it is lower than 5.43V, the circuit should be giving 90mA to the resistor, and when it is greater than 5.35V, the circuit should provide only 10mA to the resistor. Since we already knew the value of the resistor, the voltage across it was measured to determine the current value using Ohm's Law.

After the initial testing with the resistor as the load was successful, the resistor was removed and four 1.2V NiMH batteries in series were connected at the output. At this point, the power supply at the negative terminal of the comparator was disconnected. Instead, a wire was used to connect the positive terminal of the battery to the negative terminal of the comparator. The voltage of the batteries was measured before connecting them to the circuit and made sure that they were not already fully charged. The batteries were placed in the circuit and observed the increase in voltage of the batteries. The current value was measured for the conditions when the voltage of the batteries was lower than 5.4V, around 5.4V and greater than 5.4V. This was done by observing the output of the comparator (whether it was high or low) and observing whether the MOSFET switches were on or off in each condition.

Display

The dispenser was broken down to its module components to simplify development, as with the other devices. The first of these modules to test which was crucial to the user functionality of the dispenser was the display that would be used. Using the 16x2 character display that we chose allowed use of several reference guides on the internet. Specifically, Matt Hawkins' code was used to communicate with the display, as it had all of the functions for initializing and configuring the display as well as functions for writing to the display [45]. Initializing was largely setting up the Raspberry Pi's General Purpose I/O ports for the four data pins and the pins necessary for pulsing. Writing to the display involved putting the first word of an ASCII character on the four data pins, pulsing, putting the second word on the pins, and pulsing again. This process would need to happen for the length of the string sent into the function. This code provided intuitive understanding of the operation of the display, as well as functions to call to operate it. Given the Attribution-Noncommercial 3.0 Unported license, the group decided to use what was there as opposed to rewriting from scratch, as it only needed modifications to allow the functions to be called from another program.

To actually operate the display with the Pi, however, it would need to be wired up to the GPIO pins on the open port. Given the way the display would be configured in code, it simply needed to be wired to any six GPIO pins. These pin numbers would be entered into the configuration portion of code. At this stage in the development of the project, there was no plan to put the dispenser components onto PCBs for the prototype, a breadboard would suffice. In order to put the display on the breadboard, it needed a pin header soldered onto it, so a 16-pin header was attached. With that securely in the breadboard, jumpers were used to wire the necessary display pins to the chosen Pi pins, as seen in Figure 19 below. The code providing initialization and writing functions needed to be updated with these specific GPIO pins. This would allow the necessary outputs to be configured upon calling initialization and for characters to be sent to the display.

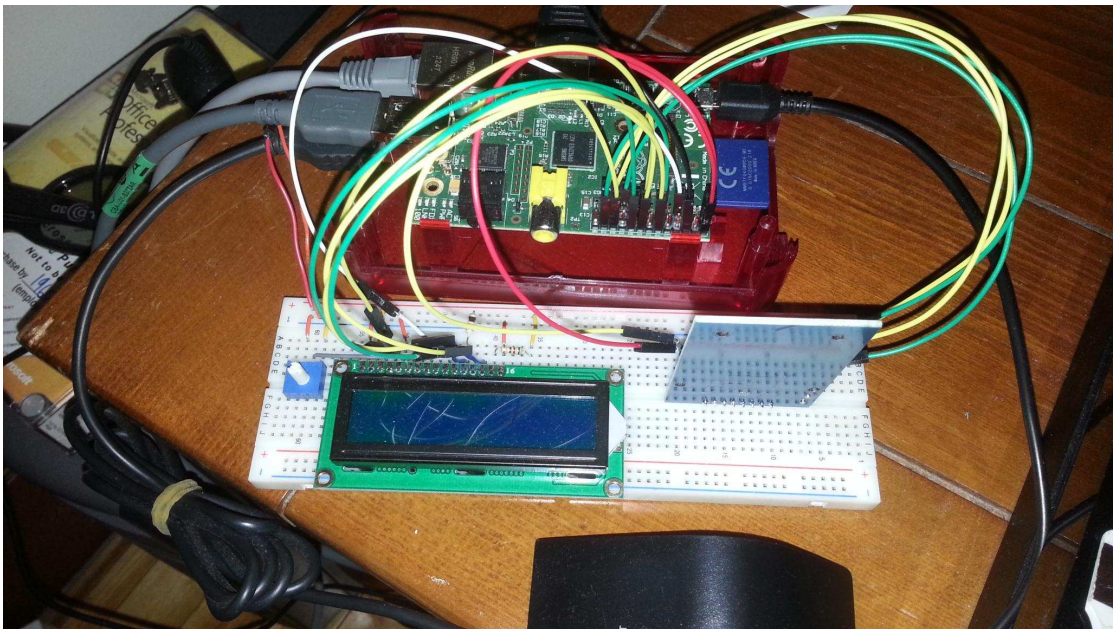


Figure 42: Raspberry Pi with LCD display and NFC radio

The main testing of this code came from investigating the capabilities of the library source code and calling its functions in various orders. From the source it became clear that once the display had been initialized with functions provided, two other functions would be the main necessities for writing to the display, and they were called lcd_byte and lcd_string. The byte function was used to actually send a character along to the display, while the string function would take in a whole string and call the byte function on the characters for the user. Calling the byte function outside of the string function was necessary because a byte would need to be sent to the display to inform it of which line to print on. So calling lcd_byte(First line) then lcd_string(whatever needs to print) would send that to the first line, then two more commands would be necessary for the second line.

The main hurdle in all of this testing was getting the GPIO pins working properly. For a while the initialization function which set up the pins was not working. After trying to change to different pins and set up a large number of them for GPIO, the display functions were still not working. At this point, a tutorial for using GPIO on the Pi was found on the internet. It was just to use one pin to power an LED on a breadboard, and it was all done using python in a terminal command line. The initialization of that individual pin was called by calling the GPIO.setup() function on the pin (from the RPi.GPIO library). Next the pin was set to different states using GPIO.output(). When this still did not work, further investigation found an answer. The Raspberry Pi GPIO pins have multiple names within the CPU, and these names are based on which model is used. For this particular model there are the “board” names, which are the same numbers as they are on the board, then there are the Broadcom SoC (BCM) number names for the pins, which are very different from the board names. The names in the initialization function needed to be changed to the BCM names to work properly.

NFC

The other challenging component to figure out for the Dispenser was the NFC module. The particular module used was chosen because of the resources found on the internet, specifically the SPI-Py library from Thiery/Wolf and program with functions written for this particular module by mxgxw [46] [47]. This module uses serial peripheral interface (SPI) for data communication, and fortunately the Raspberry Pi also has SPI capabilities using several of its GPIO pins. There are eight pins on the NFC module, with four being for SPI communication, two for power, one reset pin, and one unused pin in this implementation. The four necessary for communication are Master-In-Slave-Out, Master-Out-Slave-In, Serial-Clock, and Slave-Select, which are connected to the Raspberry Pi's corresponding pins. Fortunately for this project, the SPI-Py library takes care of the protocols there, so no development was necessary for those functions.

The code provided by mxgxw is a set of functions written in Python to interact with the NFC module, including a main loop that is constantly searching for an NFC connection and printing data about that NFC connection when found. It would be more preferable for the NFC to only read when it has something to read rather than constantly reading.

The code was tried to see if it would work as is, but a number of additional steps needed to be taken. Specifically, a line in a configuration file for the Raspberry Pi needed to be uncommented to allow SPI operation, and then it needed a reboot. It seemed like this should make it work, but another update needed to be done on the Raspberry Pi to allow access to the SPI port, using an update tool called rpi-update from Hexxeh [48]. This allowed access to spidev_0.0, which needed to be opened in the SPI-Py library functions. This would allow the NFC reader to actually communicate with the program.

Next, tests were done with running the program to see what was actually returned from the NFC reads. When run as it was, mxgxw's code ran in a while loop on the condition that if a variable 'continue_running' is true, the while won't stop. Inside this loop, operations were done on an NFC object which had already been initialized, and in particular these operations would identify a tag which comes into proximity of the antenna. The function would return something called a UID, and this was printed to the console. To test the capabilities of this code, some of the other functions were called. Specifically, Read_MFRC522 and MFRC522_Read were tested. Despite the similar names, these functions operated very differently. The first was the function which actually transferred bytes through the SPI to the reader; while the latter took in a parameter it called 'blockaddr'. This was presumed to mean the address of the block on the tag to be read. Most of the testing revolved around trying to use the second function, MFRC522_Read to read blocks of data, after a connection was already established and the UID was already printed.

Trigger

The outer container for the Trigger was made from a flashlight with all of the internal components removed. This casing was a cylindrical casing, which was the ideal shape for the Trigger. After those components were removed, the next step was to cut off the top part of the device to get the length and the width needed to place the PCB inside. This PCB was based off the schematic shown in Figure 39 in Appendix B, and included the XBee, GPS, and the microcontroller. One part of the Trigger that needed to be added was the use of a voltage

regulator, which was included on a separate PCB. The positive leads of the main PCB, the battery pack, and the charge connection were all soldered together. The battery pack was placed into the container first and then the PCB was placed next. Lastly, the button with a washer was placed over the opening of the case. For the Trigger the XBee was configured to be an endpoint and the microcontroller was programmed with the code in Appendix C.5

Figure 43: Prototype Trigger in Casing

The first step to test that Trigger was working was to check the voltage of the batteries in the device. If the voltage of these batteries was not enough, then the device needed to be charged. To charge the device, the charging circuit described in this document was used. After the Trigger was confirmed to be charged, the next step was to see if the device was in the XBee network. To find the device in the network, X-CTU was used with the Coordinator plugged into the computer. When the XBee in the Trigger was found in the network, the next step of the testing procedure was done. The button was pressed on the device, which started the emergency mode for the Trigger. When the Trigger went into emergency mode, two main signals were sent from the device. The first signal activates the alarm signal in the Box, which turned on the alarm system in the closest Box. The Trigger also started to send GPS data to the Coordinator. After the Trigger was sending data consistently to the Coordinator, the next step was to move these two devices farther apart. This was done to determine the range of the Trigger. Another step was to determine the range of the Trigger and the Box. This was done by moving the Box and the Trigger farther apart to see if the Box received the alarm packet.

Box

A transparent, plastic food container was used to house the Box's components. A 1 inch diameter hole was drilled into the side of the box to allow for the alarm hub's LEDs to be mounted outside the box while still connected to the inner alarm PCB. The 12V box battery was simply placed on one side of the container. Velcro was used to secure the alarm circuit and Box PCBs, as well as the charge controller to the inside walls of the container. A hot glue gun was used to secure the alarm hub and its outer cover. Two steel holders angled approximately 35 degrees were centered and mounted approximately 4 inches apart on top of a foot long oak board. The oak board was then screwed to the top of the food container cover. Another hole was drilled through the oak board and food container cover to allow the solar panel wires to reach the battery within. With all the inside components secured, the top of the food container was replaced, the solar panel was put on the steel holders, and the Box was ready for testing.

In order to ensure that the box could both receive and send data, each one of the three XBee modules was utilized: one set as the coordinator, one set as a router, one set as an endpoint. This was programmed with the X-CTU software discussed in the "XBee to XBee" section. Once each module was programmed, the coordinator XBee was then connected to a laptop via a USB breakout board. The router XBee and endpoint XBee were provided power via separate power supplies of 3.3V.

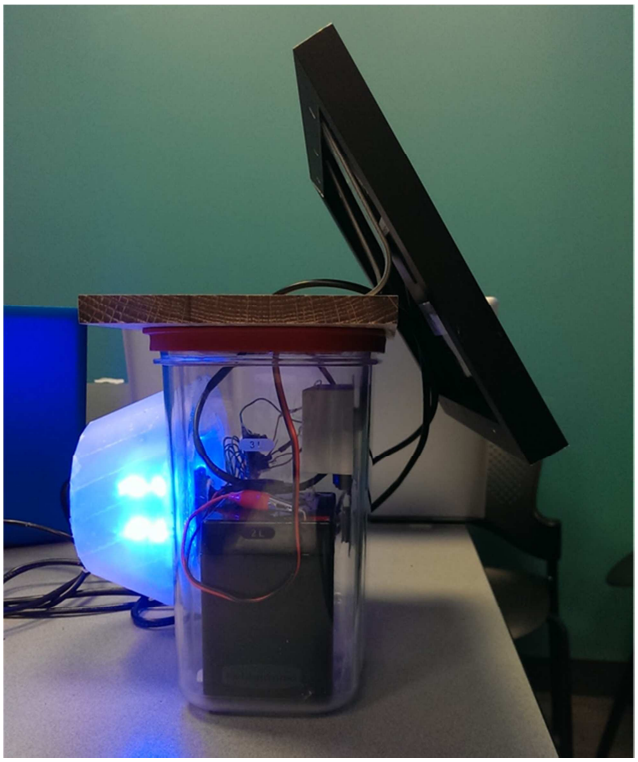


Figure 44: Prototype Box in casing

First, the router XBee was connected to the microcontroller as outlined in the "Microcontroller and XBee" section of this chapter and the microcontroller's button was pressed to send a signal from the trigger XBee to the router XBee. This signal was interpreted by the Box, and the alarm system would be activated.

The next step in testing was to have the endpoint XBee and its power supply spaced far away enough so that endpoint signals were no longer picked up by the coordinator XBee. This simulated the trigger being out of sight of the police headquarters. Then the router XBee was introduced with its separate power supply as a midway point between the two other XBee modules. The microcontroller with the same program as step 3 was connected to the endpoint XBee and the same button was pressed. This was meant to simulate the trigger (endpoint XBee) sending a signal to a box (router XBee) which would in turn pass along the signal to police headquarters (coordinator XBee).

Dispenser

The main components of the Dispenser broke down to several products that could be bought and combined for this purpose. The centerpiece was the Raspberry Pi, as the brains. This was augmented by the LCD display and NFC reader, both of which required programming to get running. Additionally to this, a barcode scanner, for the student IDs, and a numerical keypad were added for the user experience. Finally, there was the addition of the Trigger charging circuit to the Dispenser, though this did not attach to the Raspberry Pi, it was instead designed to run on its own. A container was obtained to contain all of this, while Velcro was added to the outside for holding on the number pad. The top would be left open, to allow the Trigger to rest in the top charging when not in use.

The Dispenser was tested piece by piece. First the Raspberry Pi was assessed, simply to set it up the first time and get a feel for running programs on it. Modules were added to it one by one to continue testing, and eventually the program combining all of the modules came together to control everything. This program initialized everything and then welcomed the user. This would run through a set of steps to take in the ID number through the scanner, then ask whether it's checking out or in, then read with USB, then send a signal to the Coordinator for inventory update. This was tested thoroughly by stepping through all of the possible paths through the program looking for bugs, as well as running and rerunning a lot to check for non-recurring problems such as issues with initialization.

The container of the Dispenser for the prototype was a transparent hard plastic container. The display and the keyboard for user interface were attached to the front side of the Dispenser outside the container. The charging circuit built to charge the batteries inside the Trigger was also placed inside the Dispenser near the top of the container, where the Trigger would be placed in the Dispenser. In order to provide the wire connection between the charging circuit and the batteries inside the Trigger, two 9V snap connectors were used. The positive and negative wires of the charging circuit were attached to one of the 9V connectors which would be left open when the Trigger is not in the dispenser. When the Trigger is returned to the Dispenser, the 9V connector from the charging circuit would be connected to the 9V connector of the batteries in the Trigger.

Coordinator

The Coordinator itself was born from testing the XBee radios and the Trigger, as there needed to be something to receive test messages for review. Using the USB breakout board with various programs in Windows, it was possible to send and receive test packets over the XBee modules. At a certain point, however, it became clear that a plan was needed for how the system would display information to the police dispatcher. Rather than build any casing, though, for the prototype it was decided to simply use this USB breakout board with a companion program to receive GPS coordinate packets, translate them into a KML file, then send that to the user interface.

To accomplish this, a user library for Linux called libxbee, written by Attie was installed for communicating with the USB connected Xbee directly [49]. This library provided functions for making connections with remote XBee's from the program. This would allow the Coordinator program to listen for incoming packets from Triggers so that if new coordinates came in, the Coordinator would translate them into a KML file. This KML was simply read into Google Earth for displaying for the purposes of the prototype.

Testing the Coordinator program happened in phases. The first of these was testing to see if the functions used to configure the connection between the program and the USB connected XBee were working by simply broadcasting a test packet. Next, a connection was set up with a specific XBee, which was controlled by another computer. This other XBee sent a packet to the Coordinator program to be printed to the console. Next, a function was written that would translate coordinates contained in packets of a specific format into KML format. This function was tested as a standalone program first, getting input from the console instead of a packet. Then the ability to actually open and write to a KML file was added to the function. After testing with this new capability, the function was ready to be added to the main program. Next, a function which would run in another thread was written, and this function would communicate with the Dispenser to update inventory. This was tested first by sending the correct packet from a test program into the Coordinator program to make sure it handles that case properly, without the Dispenser doing the sending. Then the Dispenser itself was used to do the sending.

System Level Testing

Range Tests

One of the first tests needed for the system was to test the range of the device. The reason that these tests needed to be done was to find out the range of the Trigger in the system. This range would be used to determine the optimal distance that the Boxes would have to be placed apart. The first range test done was the line of sight test with the Coordinator and the Trigger. This test was done by having the Trigger constantly sending data and moving the Coordinator and the Trigger farther apart. The way that the range was tested was to find the point that the Coordinator would stop receiving data, and move around that area back to where it would be receiving. The distance found by these tests would be the range of the Trigger.

After the Line of Sight testing was done, the next step was to test the range in an urban environment. This type of testing was done by having the Coordinator in one place and the Trigger moving around the WPI campus. This was done to emulate a student walking around the campus. The Trigger was carried around different objects, such as around buildings and through trees. The main test for this was to test the urban range of the Trigger in the system. The way to determine if the Coordinator was still able to receive data was to see the marker on Google Earth move in the same direction as the Trigger. The results from this testing will show the range of the Trigger in an urban environment. This will provide a range for the Boxes to be placed.

Obstacle testing was performed to determine the limits of the system's wireless communication. Various physical obstacles were purposefully placed between Trigger and Coordinator. The different types of obstacles chosen for testing were trees, metal, concrete, stone and glass. The goal of obstacle testing was to determine what types of obstacles would inhibit the wireless communication of the system. To start, the Trigger and Coordinator were positioned on either side of a cluster of trees and bushes. The trigger's button was pressed to determine if a packet would be successfully sent to the Coordinator. This same process was repeated with the Trigger beneath metal staircase and the Coordinator above it. Next, the wireless communication was tested with the Trigger beneath WPI's Earle Bridge and the Coordinator above it. This was to simulate a concrete barrier. To simulate stone, the Trigger and Coordinator were held at the corner of a stone building. The two were slowly moved away from each other along the sides of the building to see when the signal would no longer be received. Lastly, to see how the signal would react to a glass obstacle, the Trigger and Coordinator were held on opposite sides of a campus building that had glass doors and windows on either side.

Figure 45: Obstacle Testing: Through Metal

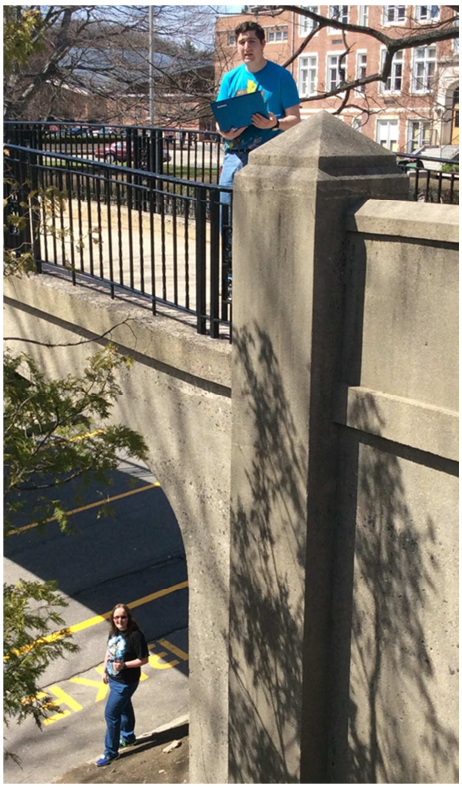


Figure 46: Obstacle Testing: Through Concrete

Figure 47: Obstacle Testing: Through Trees and Bushes

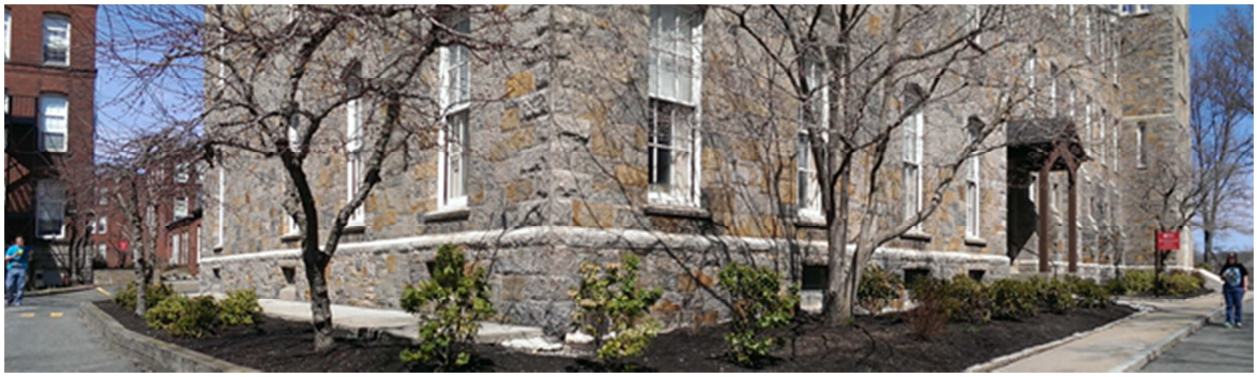


Figure 48: Obstacle Testing: Through Stone

Figure 49: Obstacle Testing: Through Glass

Reliability Testing

Being a safety system, it was important to test the reliability of the system. In order to test how reliable the system was, the packets sent from the Trigger to the Coordinator were observed over some period of time. The packets sent are vital part of the system and it was examined if there were any packets lost over time. For this reliability testing, only the Trigger and the Coordinator were involved, and the Trigger was powered by the power supply to avoid failure of the batteries for the purpose of this testing. The number of packets was counted at the Coordinator to confirm if all the packets from the Trigger arrived to the Coordinator. The packets were tracked for an hour and it was expected that the Coordinator would receive 3600 packets in total since the XBee communication was set up for one packet per second.

Chapter 5: Testing and Results

Microcontroller UART

The microcontrollers were the first modules tested, for a couple of reasons. Specifically, they were going to be running the program written to interface with the other modules, and they arrived first. The Trigger became the primary focus of initial testing, as the Box is just an MSP430 and an XBee radio, compared to the Trigger's two radios and the same MCU. These radios communicated with the MCU via a UART connection, so configuring the MCU for that type of connection was the first priority, as it would allow for modules to be combined.

In order to write the program, some of TI's sample code was referenced to see how it should be programmed, and the user guide for the MSP430 was used as a reference. Setting up the configuration and getting it to compile was the first challenge of this process, and a few tweaks were necessary to achieve that. After that, setting up the button and LED were straightforward. Next, when writing the ISRs, the sample was referenced again, but the sample's usefulness was limited by the differences between what we were trying to get the MCU's to do versus what the sample did. The sample checked the output status of two of the pins, and transmitted a character based on that. This changed the LED status accordingly. The button input was the aspect that changes the status of the LED, not outputs, so the ISRs needed to be written differently.

After this, the program was loaded onto both controllers and they were wired together with jumpers. The two UART lines needed to be connected to the reverse of itself on the other board, so that the transmit line of one was connected to the receiver of the other. As was discovered in running the programs, it is also important that the communicating devices share a ground connection, so an additional jumper was used to connect the grounds on the boards. After a few iterations to the program with tweaks in a few places, the result was the Launchpads communicating the button presses back and forth, so pressing the button on either would shut off the LED on the other. This simple test provided a set of functions and the necessary configuration to do UART communication with the MSP430, which would allow communication between the other modules later when they would be combined.

XBee to XBee

The results from the testing of XBee modules were that they were part of the same XBee network, and that the coordinator found all of the devices in the network. Figure 50 shows that all the modules were found during the test, and this screenshot is from the program the X-CTU. The purpose of this testing was to make sure that all of the XBee modules were able to be programmed to be on one XBee network, and the testing was successful. If this test was not done, then further during testing, the configuration of the XBee's could have been an issue.

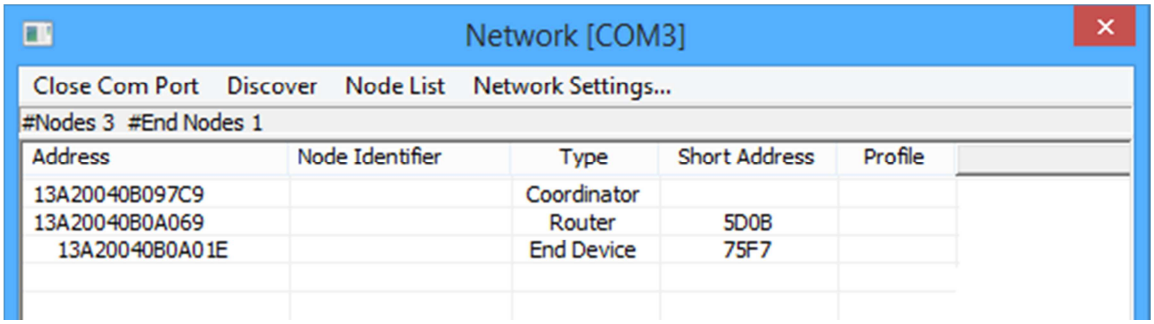


Figure 50: Screenshot of XBee Network

GPS

The testing results of the GPS on the oscilloscope show that the output is continuously sending out pulses with the amplitude of 2.85V. This agrees with the expected 2.85V LVTTL logic levels for the UART communication as described on the datasheet. The time taken for each bit was measured to be about 104 µs which agrees with the data rate of 9600 bits per second. One difficulty that the group encountered with this GPS module was to make sure that GPS has clear view of the sky in order for it to get data from the satellites.

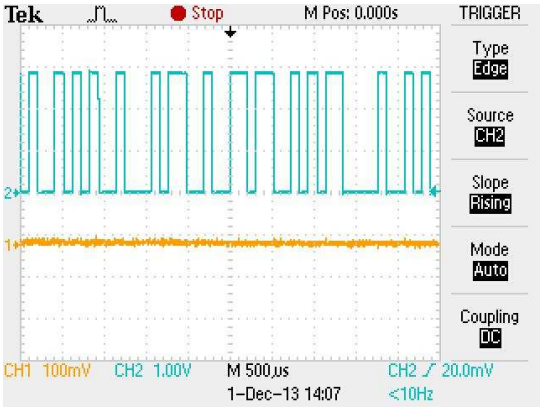


Figure 51: Scope photo of the GPS output

After making sure that the GPS is receiving data from the satellites, the GPS data was displayed using a terminal program. The results show that the terminal displays NMEA strings starting with $GPGGA, $GPGSA, $GPGSV, and $GPRMC. These NMEA strings contain different information of GPS data such as time, latitude, longitude and the number of satellites in view. After checking the NMEA strings, the GPS Locator Utility program was used to translate GPS data. It was found that the program could successfully translate the NMEA strings into readable formats with correct time, latitude and longitude. The picture below shows the screenshot of the GPS Locator Utility receiving data from the GPS module. The program also has a mini map showing the satellites in view.

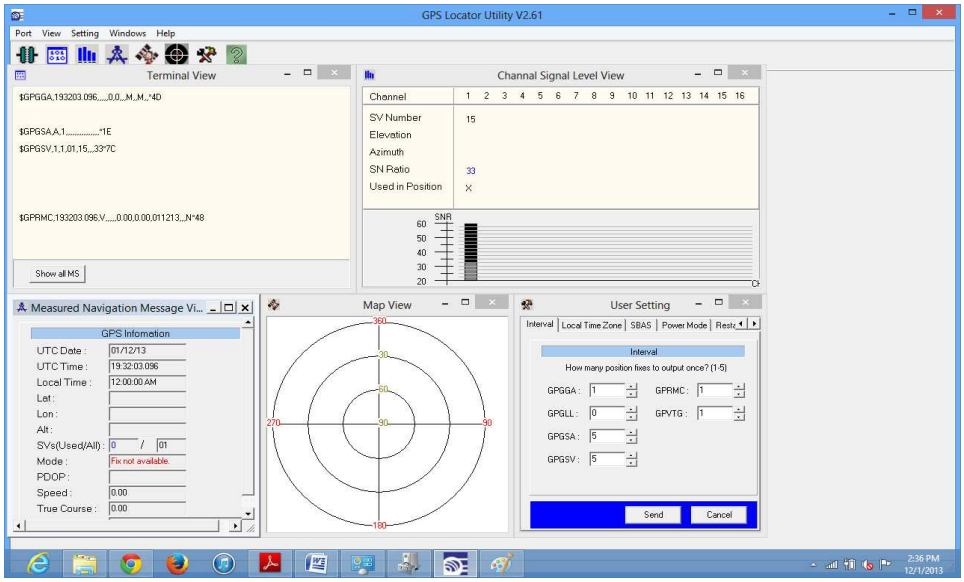


Figure 52: Screenshot of the GPS Locator Utility program showing GPS data

Microcontroller and XBee

The results from the testing of XBee modules and the microcontroller were that an XBee API packet was created as data was being added to the payload. This was important because the GPS data being sent from the Trigger was going to be changing every message, and it was not going to be one consistent message. During the testing of this combination of modules, code for the checksum and the header for the API packet were created. When the program was running on the microcontroller, it waited for an input of eight characters from the computer using a UART connection. Once it received the eighth character, the program started to create the XBee API packet. First, the program added the default header, which was being used for this test. Next, the program added the payload that was typed into the system. Lastly, the program created the checksum for the packet by following the rules defined in the documentation for the XBee API mode. After the packet was created, the microcontroller sent the full packet over UART to the XBee connected to the microcontroller. Once the XBee received the packet to send, it followed the header to send the data to the coordinator of the network. The coordinator received the packet on the computer and displayed the message on the screen with the rest of the API packet surrounding the payload.

Microcontroller and GPS

The combined module testing for the GPS and microcontroller was done by connecting the transmit pin of the GPS to the receive pin of the microcontroller through UART. To test that the microcontroller was receiving data from the GPS correctly, the microcontroller was programmed to echo back the data it was receiving from the GPS. The results displayed on the oscilloscope showed that the transmit pin on the microcontroller echoed back the correct GPS data being received at it's receive pin, but with a delay of 1.18ms. The following Figure 53 shows the comparison of the GPS output and the microcontroller output at the transmit pin using the oscilloscope. The next step was displaying the output of the microcontroller on the computer to make sure that the correct data was received. The results from the GPS Locator

Utility showed that GPS data being transmitted from the microcontroller was correct despite of the delay.

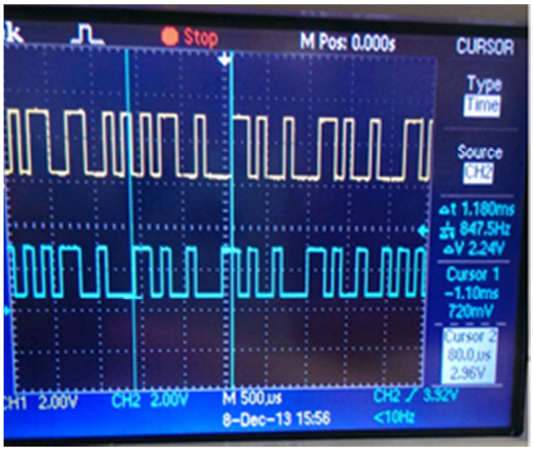


Figure 53: Scope photo showing output of GPS and output of microcontroller

Microcontroller and GPS and XBee

The result from testing the three modules together was that the microcontroller was able to parse the relevant GPS data, and create an XBee API packet. After the packet for the XBee was created, the microcontroller sent the data over the XBee network to the coordinator. When the GPS data was sent over the network, there was a new GPS packet every second. This was helpful to find the location of the user of the Trigger. One important result of this testing was a way to test the distance of the XBee devices and the consistency of the devices. Another test was to see how quickly the different devices in the Trigger can wake from sleep and start sending data back to the coordinator. There was more testing on the reliability and the wake up time for each device. One last thing that needed to be accomplished for this testing was to take the data received on the computer and store the data, so that the data was shown in a more efficient manner. The way that the data was shown on the computer was just from a terminal window showing all of the collected packets that the Coordinator received.

Alarm System

The process of dismantling the Christmas lights was not a simple one as compared to if the group had purchased blue LEDs. The wires of these Christmas lights were not easy to strip cleanly because the leads consisted of multiple thinner wires that were somewhat brittle. In an attempt to keep the thinner wires together as one, they were twisted together, but even then they had a tendency of snapping. There were a few cases where the leads had to be further stripped to access more of the thinner wires. The attachment of lengthening wires was also due to this frailness of the thin Christmas light leads that did not seem likely to fit nicely on a breadboard for testing.

When one LED was connected directly to a power supply and 2.5V was provided, it was observed that the LED was on but very dim. The voltage was increased to 3.0V and the LED became brighter. A further voltage increase to 3.4V, which is what the LEDs were rated, was noticeably brighter than the first two voltage increments.

The 510Ω resistor used for testing was selected at random from the lab kit provided to every Electrical and Computer Engineer student. When given 2.6V from the power supply, the single LED-resistor pair gave off dim light, almost to the point of being mistaken as off. At 3.0V the LED began to be bright and this brightness increased as the voltage from the power supply increased. That brightness appeared to plateau at 5.0V and upon reaching 6.3V the LED shorted out.

A similar process was repeated with the two LEDs in parallel. The following table shows the voltage from the power supply, the voltage measured across the LEDs and integrally-measured current across an LED with every 1.0V increase. By the time 6.0V was reached, the LEDs were very bright.

Table 9: Testing Two LEDs in Parallel

V_{supply}	V_{LED1}	V_{LED2}	I_{LED}
2.5 V	2.5 V	-	0.01 mA
3.0 V	2.83 V	2.80 V	0.49 mA
3.5 V	2.91 V	2.86 V	-
4.0 V	2.97 V	2.95 V	2.07 mA
4.5 V	3.01 V	2.98 V	-
5.0 V	3.04 V	3.01 V	3.85 mA
5.5 V	3.07 V	3.04 V	-
6.0 V	3.10 V	3.06 V	5.75 mA

As it can be seen from these test results, the current is just under 6mA, however the LEDs have a current rating of 20mA. Because resistance is indirectly proportional to current, a lower resistance value of 300Ω was first chosen. After seeing that the voltage across the resistor was approximately 2V for the 300Ω resistor, the desired 20mA current was used to calculate the ideal resistance. This was how an even lower resistor value of 100Ω was chosen and ultimately made its way into the final design. The testing measurement from using different resistor values is seen in the table below. As it can be seen, the current became very close to the desired value with the change in resistance.

Table 10: Testing With Lower Resistor Values

V_{supply}	$V_{resistor}$	V_{LED}	R	I
5.0 V	2.09 V	2.89 V	300 Ω	7.0 mA
5.0 V	1.95 V	3.13 V	100 Ω	19.5 mA

Testing the siren of the alarm system took place in an otherwise silent room. The following table demonstrates the decibel result of testing with different voltages at different distances from the siren. Apart from an increase in pitch, such a change in change in voltage did not greatly affect the decibel level of the siren.

Table 11: Testing the Siren Up Close and Across the Room

Distance	Voltage	Decibels
up close	5 V	91 dB
across room	5 V	74 dB
up close	12 V	91 dB
across room	12 V	81 dB

The two LEDs from previous testing when connected with the two MOSFETs, power supply and function generator successfully flashed in an alternating pattern. The power supply was set to 5V and 0.02A.

Once all 16 LEDs, resistors, and the LM555 timer were placed on breadboards, several different capacitor values were tested with the 555 timer. The following table lists the different pairs of capacitor values tested with the circuit, and C1 and C2 shown correlate to the C1 and C1 of the schematic in Figure 30 called Working Schematic for Alarm System. The initial values of R1 and R2 were 22kΩ and 51kΩ, respectively.

Table 12: Testing With Different Capacitor Values

V_{supply}	C1	C2	Frequency
5.0 V	100 µF	1 µF	8Hz
5.0 V	100 µF	10 µF	0.750Hz
5.0 V	1 µF	10 µF	-
5.0 V	10 µF	1 µF	8.7Hz

It was found that a change in C1 determined the frequency of LEDs' flashing. A higher C1 value resulted in a slower rate of flashing. The final two C1 and C2 values tested were fast but the flashing was unevenly timed between the two LED groups. The value of R2 was changed to the 75kΩ, as this allowed the circuit to further approach the desired 50% duty cycle. The group determined that the rate of flashing should be slightly slower than that of the fourth C1 and C2 test, and so C1 was changed to the final value of 5µF.

After the LEDs were placed in the alarm hub and the alarm circuit was soldered to the alarm PCB, it was found that the alarm system still functioned as desired. It continued to function properly with the siren added and the volume of the siren was found to be lower when the cover of the box was closed. During this testing, one of the 16 LEDs failed and no longer turned on. At this point in prototype construction, it was deemed counterproductive to replace it.

Solar Panel and Battery

Initial solar panel testing was done by the window of a campus laboratory revealed the effect the amount of exposure has on the solar panel's voltage output. In indirect sunlight, the voltage measured was 11.9 V with 0.0001 A of current flow. With direct sunlight, 21.12 V was measured with 0.12 A. As compared to the solar panel's rated current and voltage, the voltage and current values for direct sunlight were very close.

When the solar panel, 3 kΩ resistor, and charge controller were connected, 2.9 V was measured across the resistor. This calculated out to 0.97 mA of current. The reason the voltage was so low was due to the size of the resistor.

Testing of the Box battery, resistor and charge controller showed 12.8 V flowing across the resistor. From this measurement, it could be determined that there was 4.26 mA of current. The voltage measured would be how all the circuitry inside the Box was powered.

Connecting both the solar panel and Box battery to the charge controller with a load was meant to simulate the full Box circuitry. When this was done, 12.84 V was measured across the resistor, with a current flow of 4.28 mA.

During outside testing of the panel, facing directly towards the sun produced 21.9 V. Facing the panel away from the sun measured a voltage of 19.48 V. It was observed that there was not a large difference in voltage. The panel, battery and resistor all connected to the charge controller and facing towards the sun showed 13.11 V across the resistor. Removing the panel, 12.92 V were measured across the resistor. When the resistor was removed and the panel was reconnected, 13.05 V were measured. Since all these voltage measurement are above 12 V and the battery is rated 12 V, the solar panel has the ability to charge the battery and power the box's circuitry. Even in indirect sunlight, the panel still collected enough light energy for this purpose.

Power Interface

Power Interface for Trigger

The initial testing of the L4931 voltage regulator with power supply voltage revealed that the output was a constant 3.3V without any load attached to the output. When the 100 ohm resistor load was connected to the output, the voltage was a constant 3.3V as expected, but when the bigger resistor of 1 kilo ohm was connected, the output dropped by about 10mV.

However, since the actual loads in the Trigger were much smaller than 1 kilo ohm, this was not a problem for the system. When this voltage regulator was tested with the batteries at the input without any load, the output voltage was verified to be a constant 3.3V. When tested with microcontroller, GPS and XBee as the loads at the output pin of the regulator, the Vcc of each load was measured and verified to be 3.3V. Next, when the coordinator was set up on the computer to search the endpoint in the Trigger, the end point XBee was found in the network using the X-CTU terminal program. Lastly, the power interface for the whole device was tested in the emergency situation with the button press. It was found that the GPS data was received correctly at the Coordinator on the computer.

Power Interface for Box

The results of testing the ADP1111 regulators with power supply voltage as input showed that the output was constant 3.36V and 5.08V respectively at the output of each voltage regulator. Testing with the different loads also revealed that the output voltage was constant, but bigger loads tended to reduce the output voltage by a few millivolts. When tested with the 12V lead acid battery, the output result was found to be the same. The results of the next step when the loads were connected to the output of the regulators also confirmed that the power interfaces were working properly. The ADP1111-3.3 regulator, when connected with the microcontroller and XBee at the output pin, showed that it was providing power to both the microcontroller and XBee with a constant 3.32V. For ADP1111-5V, the alarm system including the LEDs and the siren was connected to its output through a MOSFET switch controlled by the microcontroller. It was observed that the MOSFET switch was receiving 5.02V at its source as expected. The other power pins of the alarm system were measured and confirmed that they were receiving a constant 5.02V. After measuring the voltages at the power pins of the components, a Zigbee network was set up and it was confirmed that the router XBee in the Box which was attached to the 3.3V regulator was found in the network. When the coordinator XBee sent the alarm signal to the router XBee, it was observed that it activated the alarm system with LED lights flashing and siren sound**.**

Charging Circuit

The testing of the individual modules in the charging circuit provided useful information to define some parameters in the final design of the charging circuit such as the charging current, the optimal reference voltage, and the values of the resistors used in the comparator.

The result of testing the NXCP10M45S current regulators showed that it yielded constant current but the value of the output current varied by a few milliAmperes if the input voltage was changed. The following tables summarized the results of this testing for the 10mA and 90mA current regulators. For the 10mA current output, a load resistance of 50.5 ohm from the ECE lab kit was chosen and for the 90 mA current output, a load resistance of 10 ohm was chosen. The reason for choosing a smaller resistor for 90 mA current output was to prevent shorting of the resistor.

Table 13: Testing results of 10mA current regulator at different Vcc

Vcc	R_L	Current
5 V	50.5 Ω	10 mA
8 V	50.5 Ω	10.23 mA
12 V	50.5 Ω	10.26 mA
15 V	50.5 Ω	10.28 mA

Table 14: Testing results of 90mA current regulator with different Rk

Vcc	R_K	R_L	Current
5 V	14.3 Ω	10 Ω	65 mA
5 V	8.3 Ω	10 Ω	73 mA
8 V	5 Ω	10 Ω	88 mA

As can be seen from the tables, it can be concluded that the 10mA current regulator produced the current accurately only with a small variation for higher Vcc values. However, it was observed that when the current regulator was configured to produce 90mA, the resistor specified for the 90 mA current output in the datasheet, which was 15 ohm, could only yield 65 mA with 5V Vcc at the input. Therefore, a smaller resistor R(K) of 5 ohm was replaced to achieve the desired 90 mA current output.

The testing of the voltage reference TL431 with the specified resistors revealed that the output voltage was a constant 5.4V. It was a few millivolts different from the calculated values due to the 5% tolerance of the resistors.

Testing of the comparator using UA741 op-amp also revealed that the low threshold voltage was 5.32V and the high threshold voltage was 5.45V. This means that if the negative input terminal was increased from a small voltage, the output voltage would start at 0V and change to high(Vcc) at 5.45V and if the voltage of the negative input decreased from a high voltage, the output would start at Vcc and change to 0V when it hit 5.32V. The testing results confirmed this behavior of the comparator.

After the testing of the individual components was carried out successfully, the whole charging circuit was assembled and tested. The initial testing of the charging circuit with a resistor as the load showed that the voltage across the 10 ohm resistor was 0.98V which confirmed that the total current flowing in the load resistor was (0.98V/10ohm=98 mA). The next test was replacing the load resistor with the NiMH batteries and connecting the negative terminal of the op-amp with the positive end of the batteries. During this testing, it was observed that the voltage of the batteries gradually increased until it reached to 5.45V. It was confirmed that the current charging the batteries before the voltage hit 5.45V was 98 mA. During this stage, it was confirmed that the output of the comparator was high(Vcc) and the two MOSFET

switches were turned on, which in turn, provided power to the 90mA current regulator. However, it was observed that the current regulator providing 90mA was producing a lot of heat when it was turned on. This led to the decision of reducing the current from 90mA to 75mA to avoid heating up the current regulator too much. The resistor R(K) in the current regulator circuit was changed to 20 ohm to obtain 75mA. However, after the voltage of the batteries was greater than 5.45V, it was observed that the charging circuit was still charging the batteries with 85mA. After debugging the circuit, it was observed that this behavior was due to the current from the 10mA current regulator feeding back to the circuit. This problem was resolved by adding an additional 1N4004 diode between the output of the 10mA current regulator and the 75mA current regulator. After adding the diode, it was measured that the current charging the batteries was 10mA. The voltage output of the comparator was also measured to be 1.8V which was lower than the threshold voltage of the MOSFET, and thus the switches were turned off, thereby shutting off the 75mA current regulator.

Display

The tests run on the display eventually yielded useful functions for printing to the display to be used for the Dispenser's interface. Additionally, an understanding of how to call the functions, and proper initialization of the display was a result of the testing. A number of strings were sent to the display, of various sizes, to test its capabilities. For example, strings longer than 16 characters were sent to a single line to determine how the display would handle the overflow. In this case, it simply did not print more than 16 characters per line, so if it would be necessary to print a sentence longer than that, both lines would be needed. Also a result of testing with the display showed how fast the display printed things and the effect that using delays of various times would have between printings. For example, one of the print tests printed two lines of text, called a sleep function, and then printed new strings to look at the noticeable delay in printing. It was determined that a sleep of 10 was much too long for natural reading, while a delay closer to 1 was a bit too quick. However, there was no noticeable delay as the actual printing occurred, from the moment the printing started to when it finished looked instantaneous.

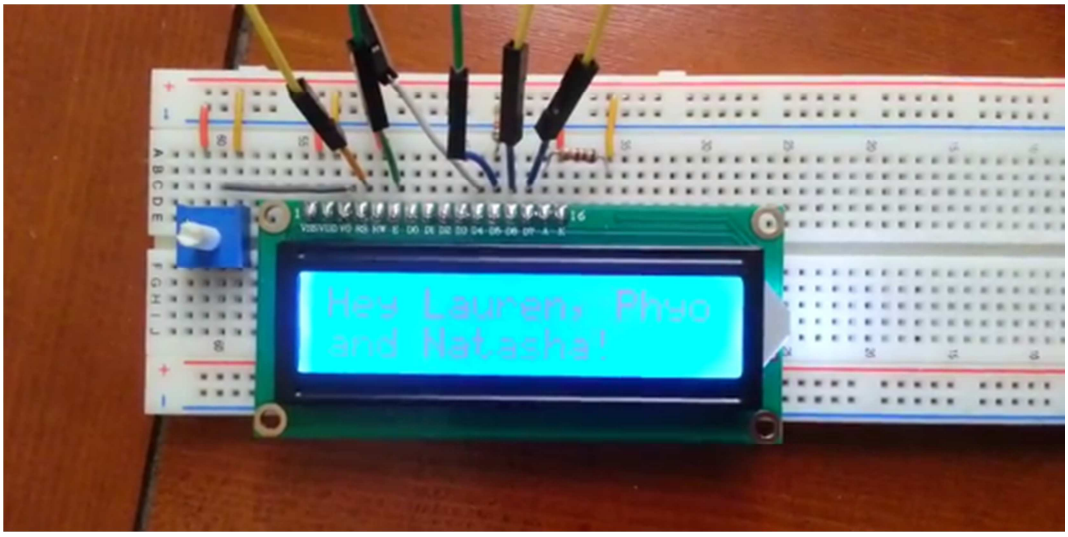


Figure 54: Working LCD display reads "Hey Lauren, Phyo and Natasha!"

NFC

The testing of the Read_MFRC522 function did not yield useful results, as reading from individual register addresses does not provide payload data. Unfortunately, the testing of the other read function, MFRC522_Read, also did not return payload data. Despite trying a wide variety of addresses to attempt to read the payload, which was written to the test tags using a smartphone, this function never returned any of the data. The test reads would either return that the size was 4 but there was no data, or that there was an “Error while reading” and the size was 0. However, through all of this failed payload reading, the UID consistently worked, and was a unique identifier of the tag. Given this, it could be used instead of the system-specific trigger ID number for identifying the trigger. Also given that this testing was very late in the development of the system, it was decided that the UID would be used instead.

Trigger

The Trigger was soldered onto the PCB ordered from OSH Park. The microcontroller, GPS, and XBee were all placed in sockets. The reason that these devices were placed in sockets was to allow the group to reprogram the devices. The button was added onto the board, and wires were placed instead of the transistor logic to switch the receive line for the microcontroller. The reason that the button was used to provide power to the GPS and to control the switching of the receive line was that the transistor logic did not work. This did not work due to the transistors being unable to send the data to the microcontroller. Instead of trying to fix the logic, the button was used instead.

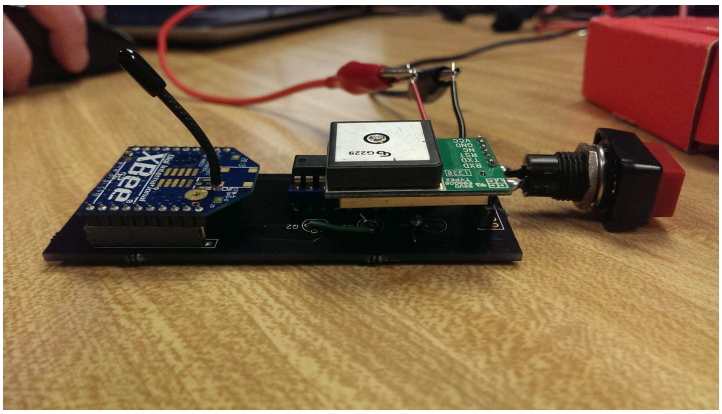


Figure 55: Soldered and assembled Trigger on PCB

The first part of testing for the Trigger was done by making sure that the voltage on the batteries was high enough for the Trigger to operate. The minimum voltage needed to operate all of the different devices within the Trigger was 3.7V. When the batteries were charged above the minimum level, the next step of testing occurred. This next step was to see if the endpoint XBee in the Trigger was found in the network. Once this was found in the network, the next step was that the button was pressed. Once the button was pressed, power was provided to the GPS, and the RTS line of the XBee was set high. The reason that this line was set high was so that the receive line of the microcontroller would only receive data from the GPS and not the XBee. The way that this was tested before placing this feature in the Trigger was seeing what the output line of the XBee would be when the RTS line was set high. When this line was set

high, the output from the XBee went into high impedance. This meant that the XBee would not send any data to the microcontroller as long as line was high. One way to fix the issue of having to set the RTS line would have been to use a microcontroller with two UART connections.

After the Trigger went into emergency mode, it sent an alarm signal to the closest Box in the system. The way that this was tested was to have the Box turned on, and not in emergency mode. The button was then pressed on the Trigger, and when the Box received the alarm packet, it activated its alarm system. The results from this testing allowed the Box to be activated from the Trigger instead of the Coordinator in past testing. Once the emergency signal was sent to the Box, the Trigger sends GPS data to the Coordinator. The Trigger sends XBee packets with GPS data in the payload about once a second to the Coordinator in the network. The results from this testing were that the Trigger was able to send GPS data to the Coordinator consistently.

One last test that occurred was to determine if the Trigger would able to be charged from the charging circuit. This was done by connecting the fully assembled Trigger to the charging circuit and seeing the voltage change for the batteries in the device. When the voltage of the batteries increased, then it was shown that the charging circuit was able to charge the batteries with the Trigger being fully assembled.

Box

The first step in checking the Box functionality was to send a sample alarm signal from the Coordinator. This signal was used to emulate the signal sent from the Trigger to the Box, which would activate the alarm system. When this signal was sent, the two lines on the PCB with the XBee and the microcontroller would be set low. The reason these lines were set low was the alarm board was expecting a control signal that was active low. Since these two lines were set low by the microcontroller, this showed that the Box was able to interpret an alarm signal and set the appropriate lines to low.

The next step was to confirm that the alarm PCB worked. This was done by providing power and a control signal to the PCB. When the alarm board was activated, the correct alternating signal was generated by the 555 timer, and the LEDs would flash on opposite sides. The next hurdle was to get the LEDs connected through the hole on the side of Box. The result from this was that the LEDs were able to flash in an alternating pattern when the control signal was provided by power.

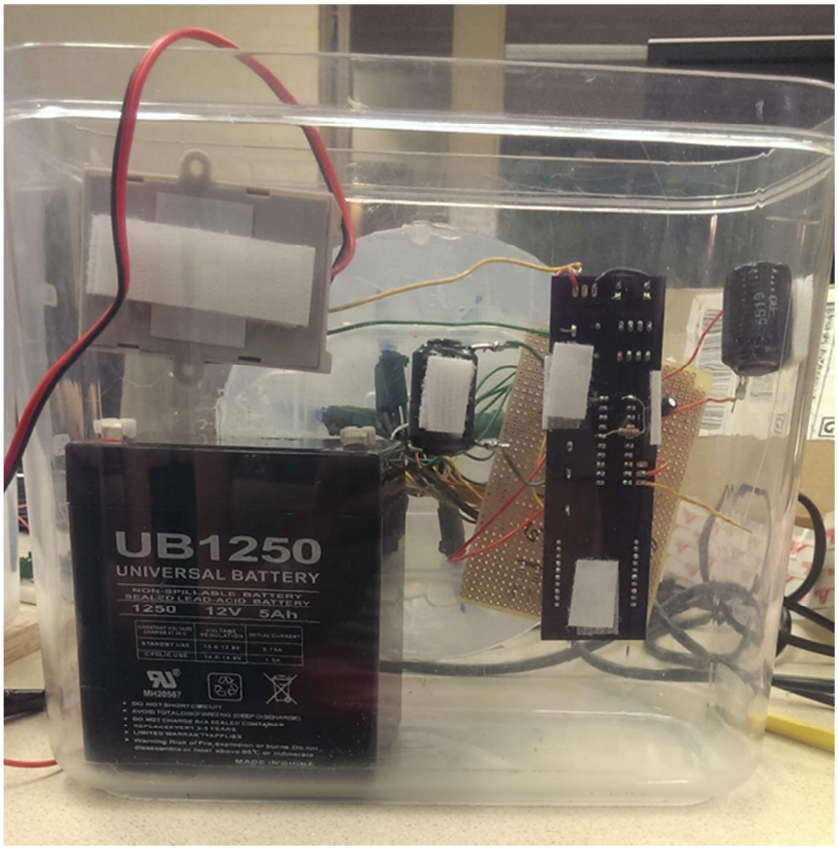


Figure 56: Prototype Box in case

After both the Box PCB and the alarm PCB were confirmed to be working correctly, the next logical step was to combine these two boards together in the casing. Both of these boards, the battery, siren, and the charge controller were placed inside the casing. Once all of the devices were connected correctly, the Coordinator sent a sample alarm packet. This alarm packet caused the alarm system to be activated on the Box, and this showed that both PCBs worked together to activate the alarm system. One last part to confirm with the Box was to see that the Trigger sent the alarm signal, and the Box was activated. This showed that the Trigger was able to send the alarm signal to the Box.

The last step to confirm that the Box was working correctly was to take the entire unit outside with the solar panel. This result was that the Box was able to be powered from the 12V battery and the solar panel connected to the charge controller. When the solar panel was in direct sunlight, it provided enough power to the battery. One reason for this was that the battery chosen for the Box was a 12V 5Ah battery. This battery was able to provide enough power to all of the circuitry in the Box.

Dispenser

The testing for the Dispenser as a whole largely resulted in success, though not all features worked. The program that ran all of the various devices used together can successfully read in ID numbers from the barcode scanner, then use the basic user interface of the LCD display to give the user options, read in input from the keypad to allow the user to choose an option, and it then scans the NFC ID of the tag that will be in the Trigger to track which one is

being checked out or returned. The code to send inventory information to the Coordinator was written in C as a Python extension. It was capable of taking in a string from the Python program containing either “checkout XXXXXXXXX” where the X’s are the ID number of the student, or “checkin”. These strings would then cause a packet to be sent to the Coordinator through a socket, and the Coordinator program would be updated accordingly. The program flow for the Dispenser was interacting with the user and scanning the NFC ID in the outgoing or incoming Trigger, then sending the updated user information to the Coordinator.

Coordinator

It took some reworking the code and a bit of digging into the root directory, but eventually the packet from the program was sent out for other XBee’s to receive. Additionally, once the ability to create a connection between nodes was established, the other tests with the libxbee library fell into place. It was easy to send individual packets to other XBee’s, and simple to receive them, and they printed to the console for easy verification. The next step was taking the incoming packets containing GPS information and pulling the relevant pieces out to convert to KML format. A function was written outside the XBee communication program to parse the packets that would be received containing GPS information. This was written separately at first so it could be run by itself and tested with input in the correct format from the console. This took debugging, and a lot of print statements, to get working properly in its own program, but once it was capable of formatting information outside the program, it was tested running inside the program. Here, there arose a problem with the dynamic memory allocating happening inside the function, as it needed buffers to use to pull the packets apart. Fortunately, this could be solved for simplicity but changing to statically allocated buffers within the function, although a future program would need to dynamically allocate if it were running for long periods of time

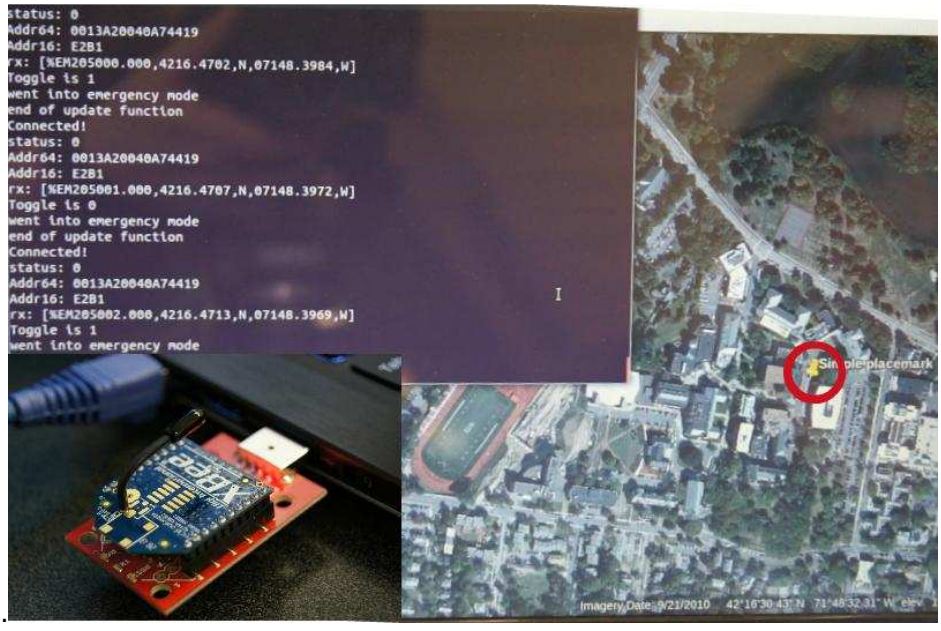


Figure 57: Coordinator hardware and Google Earth interface

At this point the program needed to be tested with the Trigger to see if it could receive data properly in emergency mode. When an initial test was run to see if the program was parsing packets correctly, the program crashed abruptly whenever it received packets. This bug did not seem to go away at all for a while during testing, and it was only solved when it was modified to print the data it was parsing. The problem during this testing was that the Trigger was indoors, so it was not receiving data. It still should have parsed the packet, just unsuccessfully, but that was filling buffers with null and then attempting to access elements of those buffers by indexing them, which was an issue. This was solved by adding a test case within the parsing function to check for null values. If the values were null, it printed to the screen that GPS was not found yet and it was unsafe to write to KML. Upon testing outside, the parsing worked just fine, including writing to the appropriate spot in the KML file.

This KML file would allow Google Earth to network link to the program, so that it automatically updated with the new KML. This would allow tracking of the user if they were running from an assailant. However, during initial testing with Google Earth, the location of the GPS according to the software was not correct. Upon inspecting the parsing program further and manually entering the coordinates, the correct location was found. It was discovered after more digging into KML formatting, that the parsing function needed to not only reformat the data, but also convert from the form that the GPS module uses to that which KML expects. This involved writing a new function which took in a string, converted it to an integer, converted the integer to the notation used by GPS, then converting that back into string format. When this was added to the functionality, the KML was in the correct format and Google Earth appeared properly with the Trigger's location.

The next step was to test part of the program which creates a new thread to communicate with the Dispenser. This would wait to receive packets of inventory updates when they were relevant. This was first tested with a standalone program simply sending the information that this thread expected, though not the actual Dispenser program. The next test after this was using the Dispenser program itself to send the packets.

This was the last step to open up for whole emergency mode system testing. With the Coordinator program working to receive information through the network, all of the devices could be tested in emergency mode to see if they work as well as to find additional information about the system, such as range limits.

System Level Testing

The results from testing the system are qualitative results. The main reason that this occurred was due to time constraints that the group encountered.

Range Tests

The results from the line of sight testing showed that the system was able to send data from a long distance away. The only restriction for this distance was that the Trigger and the Coordinator needed to be in line of sight. The testing was done on a football field, and the Coordinator was still able to receive data from about 200 yards away from the Trigger. The data was not being received at a constant rate, but it was being received sporadically. This shows that the distance was able to be received, but the distance should be lower for more consistent data.

The results from the urban testing show that the Coordinator was able to receive data

from around the area, but once the signal was obstructed it stopped receiving. This shows that in an urban environment there would need to be more Boxes to account for the buildings obstructing the data sent.

Obstacle testing revealed that the wireless communication had varying levels of disruption dependent on the medium. The signals sent through the trees and bushes were received, but slightly sporadically. Sending signals through a metal staircase did not seem to disrupt reception. When this test was performed on the bridge, the wireless communication was more sporadic than when it occurred through the trees and bushes. After traveling less than a third of the way along their respective sides of the stone building, the Trigger and Coordinator lost communication. They failed to communicate at all through the glass building. At one point during testing, the GPS module in the Trigger became too loose to receive GPS location data and had to be readjusted. It is acknowledged that the possibility of this module being loose during testing before it was noticed exists.

Reliability Testing

The results of the reliability testing showed that the Trigger was able to send 3602 packets to the Coordinator in an hour, when it was set up in the emergency mode. One explanation that there were 2 more packets that the expected 3600 packets were that the packet sending started a few seconds before the timer began. Since the packet sending was taking place inside the room where the GPS did not have a clear view of the sky, it was observed that all the packets sent were empty. This testing method confirmed that the communication between the Trigger and the Coordinator was reliable by proving that there was no packet loss over a long period of time. However, there needs to be more reliability testing done for the system.

Chapter 6: Conclusions

This project achieved the goal of proving its design concept: a system aimed to address the safety problems on campus here at WPI. The issue is multifaceted, and for that reason it needs to be addressed on both the victim side and at police headquarters. This safety system has the potential to be capable of just that. The user has a simple interface to signal campus police and send his whereabouts via real-time GPS coordinates. The Trigger prototype is able to receive and transmit coordinates via the Zigbee network to hail police assistance remotely. Correspondingly, the Coordinator gives police a simple graphical interface to view coordinates of a student who has signaled an emergency with assistance from Google Earth. Additionally, the Box and Dispenser designs create the system infrastructure for use by a large, populated campus. The Boxes create the wide area mesh network necessary for reliable system performance, while the Dispensers track and maintain Triggers and provide convenience to users.

The Trigger is battery operated, GPS capable, and it can generate outgoing packets for the Zigbee network. The Box is able to relay emergency packets, drive an alarm system, and it charges its internal battery via a mounted solar panel. The Dispenser is capable of interfacing with users and charging a Trigger, and the Coordinator processes emergency packets and presents them to police dispatchers. Already these baseline features comprise a rudimentary system capable of addressing the issue, which was the goal of this project. Though more development could be done, the project as it is has the potential to benefit the WPI community.

Over the course of this project, the group learned about the problem of frequent incidents of crime on and around campus. In addressing this, the group researched numerous technologies, some of which carried into the final prototype. For example, it was discovered that XBee modules can create a self-healing Zigbee network, and the group learned how to build one. The group also learned how to use serial interfaces to communicate between a range of devices, and how to safely tie these data lines together without affecting signal clarity. Applying what the group learned in classes at WPI, they were able to create interrupt-driven functions to efficiently process data in microcontrollers, and they were able to implement power interfaces using DC/DC converters. Additionally, the group learned about NMEA protocols, KML format, Python programming, how to design PCBs, and how to create an array of alternating flashing LEDs.

Ultimately, this MQP was able to explore various technologies to solve a relevant problem. These modules on their own serve specific purposes, but combined into a system they provide a meaningful solution.

Chapter 7: Future Works

A number of improvements can be made for a campus wide implementation of this system. All of the devices would benefit from more robust components, the infrastructure would benefit from cleaner software and some additional features would make the system more appealing and advanced. While the current feature set works for a prototype and proof of concept, this project was limited due to time and budget constraints. A consumer version of this product would be further developed and more heavily tested.

On the hardware side, the components chosen for this project prioritized ease of development and testing over advanced functionality. If this project went to a consumer version, a number of modules would be upgraded or replaced. A microcontroller with more than one UART port would be useful for simultaneously communicating with multiple devices. A GPS module with less overhead time for finding satellites would benefit a time-sensitive safety system. Additionally, a microphone and wireless charging system for the Trigger, and a larger display for the Dispenser would add to the user experience. Even features like Box-activating panic buttons to allow them to be operated like traditional phone towers could be useful added functionality. Finally, indicator LEDs where appropriate would enhance the ease of use of any device. For instance, allowing the Trigger to indicate its battery life, network connectivity, and whether its emergency mode was activated.

With much of this project based in software, the lack of a dedicated software engineer in the group meant that some parts of the system were underdeveloped. Specifically things like the databases, network security, and police user interface should all be addressed and expanded on to enhance the system overall. Also, time constraints prevented development of significant system maintenance functions to be built in. In a full, campus-wide deployment, this system would also need to interface with campus resources such as ID servers and possibly financial software. These issues would need to be addressed and thoroughly tested, as there would need to be a certain level of system assurance before a campus would invest in this product.

Along with more rich features and a robust software backbone, a full rollout of this product would need to address a number of additional logistical considerations. For example, the casings used in the prototypes for this system would not suffice for outdoor conditions or being in the hands of users. The casings used in the future would need to be ready to withstand the elements, and there would need to be backups for situations such as snow on solar panels. There also needs to be regular stress testing of the system in a deployed state. Having thousands of devices on the network being maintained by one central location needs to meet a certain standard of stability so users can rely on the product to work when called upon. This project would benefit from an additional prototype build, followed by a beta implementation on an actual campus in order to work out user bugs, increase the system's capabilities, and test system stability under real world conditions.

References

[1] Location, Inc. "Crime rates for Worcester, MA." *Neighborhood Scout*. Web. 8 Sep. 2013. < http://www.neighborhoodscout.com/ma/worcester/crime/>.

[2] Advameg, Inc. "Crime rate in Worcester, Massachusetts ." *City-Data*. Web. 8 Sep. 2013. <http://www.city-data.com/crime/crime-Worcester-Massachusetts.html>.

[3] Reis, Jacqueline. "The 12-member Colleges of Worcester Consortium considers changes." *News*. telegram.com, 06 Mar 2013. Web. 8 Sep. 2013. <http://www.telegram.com/article/20130306/NEWS/103069906/1116>.

[4] Annual Security & Fire Safety Report. (2013). Retrieved April 28, 2014, from http://www.wpi.edu/Images/CMS/Police/Combined_File_Annual_Security_and_Fire_Report_2013.pdf

[5] Martha Waggoner.“Researchers Tackle Campus Safety Devices”. Web. 19 Sep 2013. .< http://www.ecu.edu/cs-admin/news/041305campussafety.cfm>

[6] Campus Guardian :: Campus Safety Service. (n.d.). *Campus Guardian :: Campus Safety Service*. Retrieved April 17, 2014, from http://www.campusguardian.com/index.html

[7] Mobile Safety - TapShield. (2014, January 1). *TapShield*. Retrieved April 17, 2014, from http://tapshield.com/

[8] RAVENAlert Keychain – Campus Emergency Alert. Web. 19 Sep 2013. < http://www.intelliguardsystems.com/students-parents.php>

[9] Campus-Wide Personal Duress Security Systems. (2014). *National Protective Systems*. Retrieved April 17, 2014, from http://www.nationalprotectivesystems.com/

[10] Praphul, Chandra. *Wireless Security: Know It All*. Burlington, MA: Newnes, 2008. E-book.

[11] Gislason, Drew. *Zigbee Wireless Networking*. Newnes, © 2008. Books24x7. Web. Oct. 19, 2013.<*http://common.books24x7.com.ezproxy.wpi.edu/toc.aspx?bookid=32185*>

[12] Hebel, Martin, George Bricker, and Daniel Harris.*Getting Started with XBee RF Modules*. 1.0. Web. <http://www.makershed.com/v/vspfiles/assets/images/122-32450-xbeetutorial-v1.0.1.pdf>.

[13] GPS Visualizer: Do-It-Yourself Mapping. Web. 10 Oct 2013. http://www.gpsvisualizer.com/

[14] Griffin, D. (2011, June 26). How does the Global Positioning System work ?. . Retrieved April 28, 2014, from http://www.pocketgpsworld.com/howgpsworks.php

[15] Strickland, Jonathan. "How Wireless Mobile Chargers Work" 17 July 2012.

HowStuffWorks.com. 4 October 2013. <http://electronics.howstuffworks.com/gadgets/other-gadgets/wireless-mobile-charger.htm>.

[16] Berg, Andrew. "Wireless Charging and a Tale of Two Standards." *Wireless Week*. Advantage Business Media, 03 Jul 2013. Web. 6 Oct. 2013. <http://www.wirelessweek.com/articles/2013/07/wireless-charging-and-tale-two-standards>.

[17] Higginbotham, Stacey. "10 Things to Know About Wireless Power." *Gigaom*. Wordpress.com, 04 Oct 2009. Web. 19 Oct. 2013. <http://gigaom.com/2009/10/04/10-things-to-know-about-wireless-power/>.

[18] Pollicino, Joe. "WPC updates Qi standard, increases inductive charging distance to 40mm." *Engadget*. AOL Inc., 20 Apr 2012. Web. 16 Oct. 2013. <http://www.engadget.com/2012/04/20/wpc-updates-qi-standard-increasing-inductive-charging-distance/>.

[19] Bodo's Power Systems. *Wireless Power By IDT*. 2013. 3-4. eBook.

[20] About Us. *Power Matters Alliance*. Power Matters Alliance, Inc. Web. 16 Oct. 2013. <http://www.powermatters.org/about/about-2>.

[21] Who are we. *Alliance for Wireless Power*. Alliance for Wireless Power, 2012. Web. 16 Oct. 2013. <http://www.a4wp.org/about-us.html>.

[22] Rouse, Margaret. "Resonance Charging." *Search Mobile Computing*. 2008. <http://searchmobilecomputing.techtarget.com/definition/resonance-charging>.

[23] Water, Allens. *Wireless Charging System Using Inductive Coupling*. 2010. E-book

[24] Waffenschmidt, Eberhard. “Resonant Coupling”. Web. 14 Dec 2013.<http://www.wirelesspowerconsortium.com/technology/resonant-coupling.html>

[25] Svensson, Andreas. *Design of Inductive Coupling for Powering and Communication of Implantable Medical Devices*. 2012. E-book

[26] Dr. Kesler, Morris. *Highly Resonant Wireless Power Transfer: Safe, Efficient and Over Distance*. 2013. E-book

[27] Wireless Battery Charger. (2011). Retrieved April 28, 2014, from http://www.leevenspark.com/2012/02/wireless-battery-charger.html#.U2AtEvldX01

[28] Hollywood Solar Panels (2012). Retrieved April 28, 2014, from http://losangeles-solarpanel.info/wp-content/uploads/2011/12/hollywood-solar-panel.jpg

[29] Bluejay, M. (2013). Battery Guide. Retrieved April 28, 2014, from http://michaelbluejay.com/batteries/rechargeable.html

[30] Mouser Electronics - Electronic Components Distributor. (n.d.). . Retrieved April 29, 2014, from http://www.mouser.com/

[31] Laythe, A. B. P. (2013, November 15). Interview by J. Beaulieu, N. Bonina, L. Lewis, P. Thinzar [Personal Interview]. Campus ID System.

[32] EDGE EVO. (2014). Retrieved April 28, 2014, from http://www.hidglobal.com/products/controllers/edge-evo

[33] WPI Police. (2014). Retrieved April 28, 2014, from http://www.wpi.edu/offices/police.html

[34] Jacobs, M. (2013, November 07). Interview by J. Beaulieu, N. Bonina, L. Lewis, P. Thinzar [Personal Interview]. Campus System.

[35] Your M2M Expert - Digi International. *Your M2M Expert - Digi International*. Retrieved April 30, 2014, from http://www.digi.com/

[36] Simpson, Chester. *Battery Charging*. Web. Jan 2014 <http://www.ti.com/lit/an/snva557/snva557.pdf>

[37] Shackell, Steven. *Constant Current Regulator Charging Circuit.* Web. Jan 2014 <http://www.onsemi.com/pub_link/Collateral/AND9031-D.PDF>

[38] Mohan, Ned. *Power Electronics.* Oct 2011. E-book

[39] ADP1111ARZ-5 Analog Devices. (n.d.). . Retrieved April 29, 2014, from http://www.mouser.com/ProductDetail/Analog-Devices/ADP1111ARZ-5/?qs=sGAEpiMZZMv9Q1JI0Mo%2ftarsCQOsF%2faj

[40] L4931 5V LDO TO-92 Voltage Regulator. (n.d.). . Retrieved April 29, 2014, from https://solarbotics.com/product/17130/

[41] Mifare RFID RC522 Reader IC Card RF Module Read Antenna Proximity SPI Interface. (2014). Retrieved April 28, 2014, from http://www.ebay.com/itm/Mifare-RFID-RC522-Reader-IC-Card-RF-Module-Read-Antenna-Proximity-SPI-Interface-/161135300790?pt=BI_Control_Systems_PLCs&hash=item25846990b6

[42] GPS with antenna. (n.d.). . Retrieved April 29, 2014, from http://www.skylab.com.cn/En/Product_Detail.aspx?UserInfo_ID=707901&CorpProductClass1_ID=44452&id=559783

[43] Serial. Retrieved April 28, 2014, from http://energia.nu/Serial.html

[44] Orsini, L. (2014, January 20). Raspberry Pi: Everything You Need To Know. Retrieved April 28, 2014, from http://readwrite.com/2014/01/20/raspberry-pi-everything-you-need-to-know#awesm=~oCS4sknbprwmVV

[45] Hawkins, M. (2012, July 27). 16×2 LCD Module Control Using Python. *Raspberry Pi Spy*. Retrieved April 19, 2014, from http://www.raspberrypi-spy.co.uk/2012/07/16x2-lcd-module-control-using-python/

[46] Thiery, L. (2014). In Wolf C. (Ed.), *SPI-py* Available from

https://github.com/lthiery/SPI-Py

[47] mxgxw. (2014). MFRC522-python Available from https://github.com/mxgxw/MFRC522-python

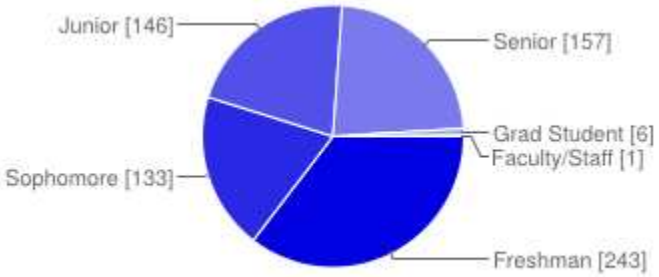
[48] McLoughlin, L. (2014). *RPI-update* Available from https://github.com/Hexxeh/rpi-update

[49] Grande, A. (2012, February). libxbee. *attie.co.uk*. Retrieved April 29, 2014, from http://attie.co.uk/libxbee

Appendices

Appendix A - Survey Results

What year are you?

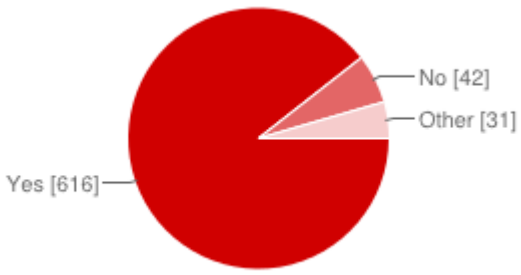


Freshman	**243**	35%
Sophomore	**133**	19%
Junior	**146**	21%
Senior	**157**	23%
Grad Student	**6**	1%
Faculty/Staff	**1**	0%

Where do you live in reference to the campus?

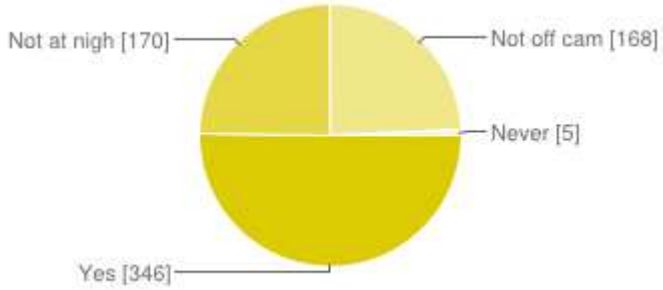
Off campus apartment	**257**	38%
Riley	**39**	6%
Daniels	**53**	8%
Morgan	**52**	8%
Stoddard	**42**	6%
Institute	**24**	4%
Founders	**47**	7%
East	**49**	7%
Faraday	**45**	7%
Other	**77**	11%

Do you like that WPI is an open campus?



Yes	**616**	**89%**
No	**42**	**6%**
Other	**31**	**4%**

Do you feel safe on and around campus?



Yes	**346**	50%
Not at night	**170**	25%
Not off campus	**168**	24%
Never	**5**	1%

What safety measures on and around campus are you familiar with?

Campus Police	**664**	28%
Blue light emergency towers	**636**	27%
SNAP	**646**	27%
RAD	**128**	5%
Safe Havens	**312**	13%
None	**4**	0%
Other	**6**	0%

Improvements for security/safety on and around campus?

Additional blue light towers	**210**	15%
Additional lighting	**456**	32%
Remote access to blue light towers	**216**	15%
Easier access to Campus Police	**233**	17%
Opt-in safety initiatives	**134**	10%
None	**78**	6%
Other	**78**	6%

Appendix B: Schematics

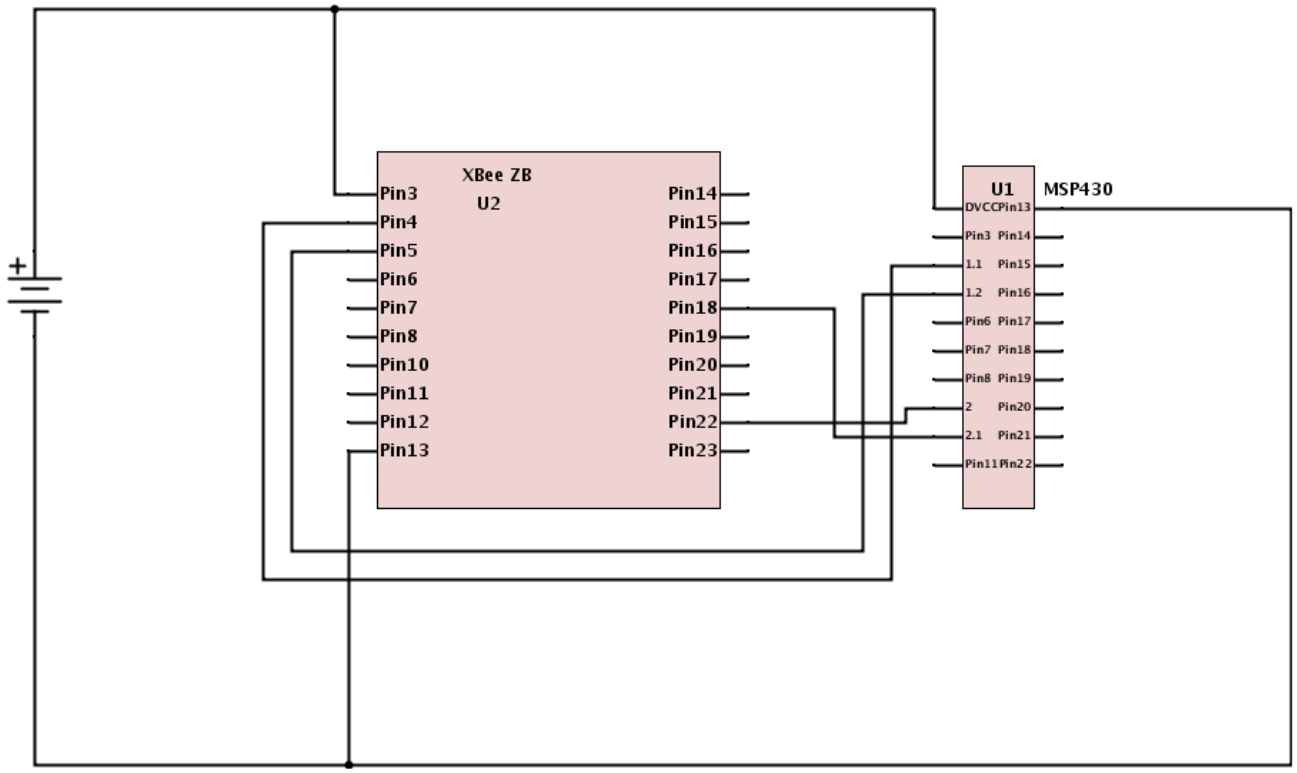


Figure 58: Early Schematic for Box

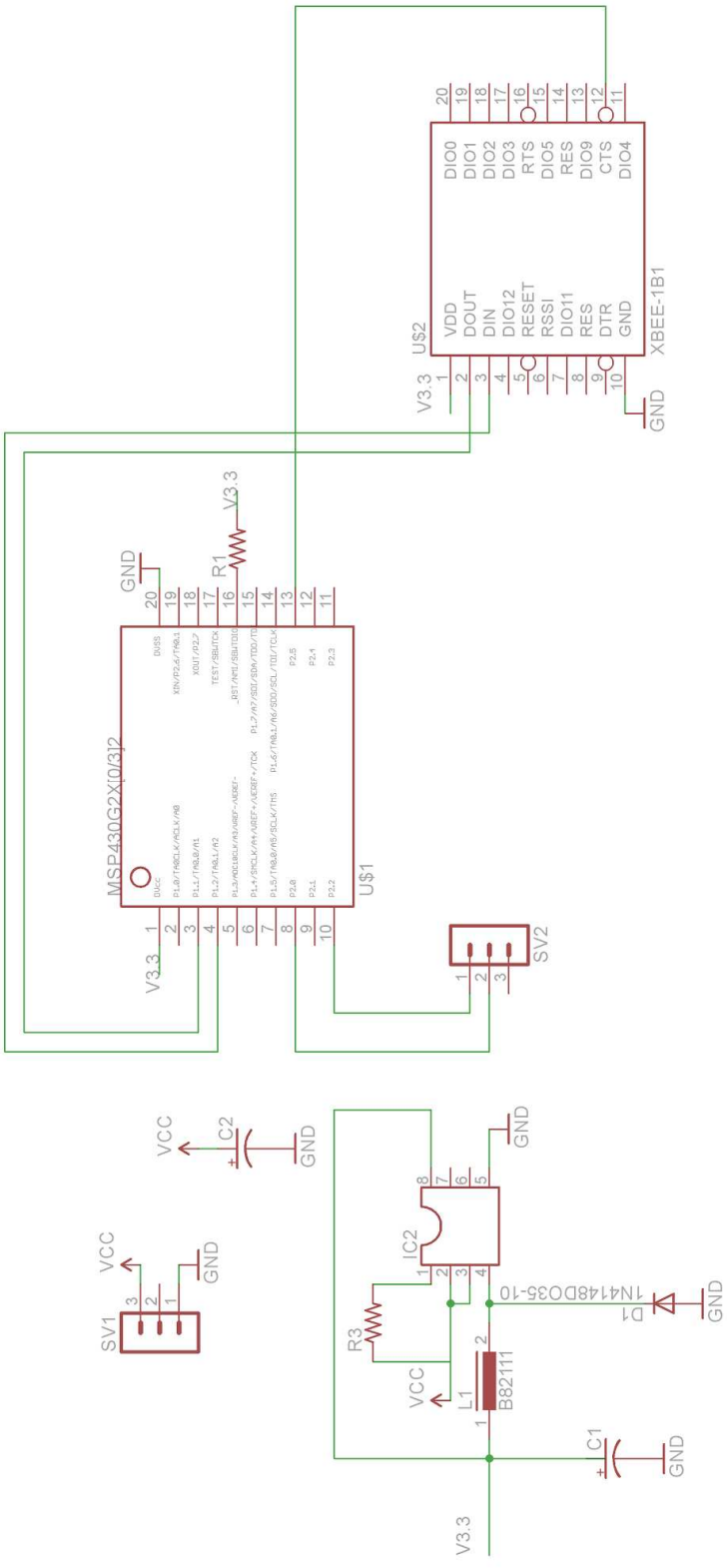


Figure 59: Schematic for Prototype Box

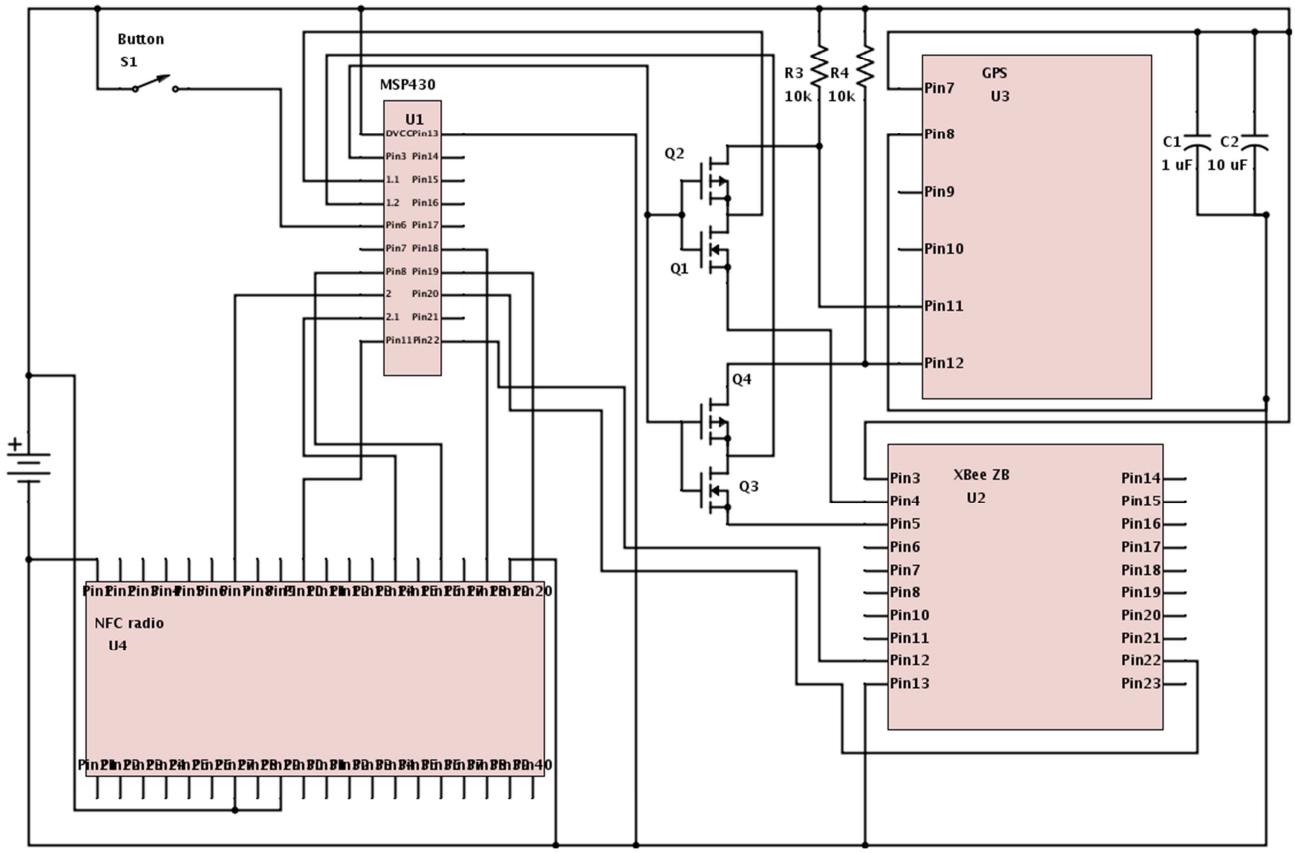


Figure 60: Early Schematic for Trigger

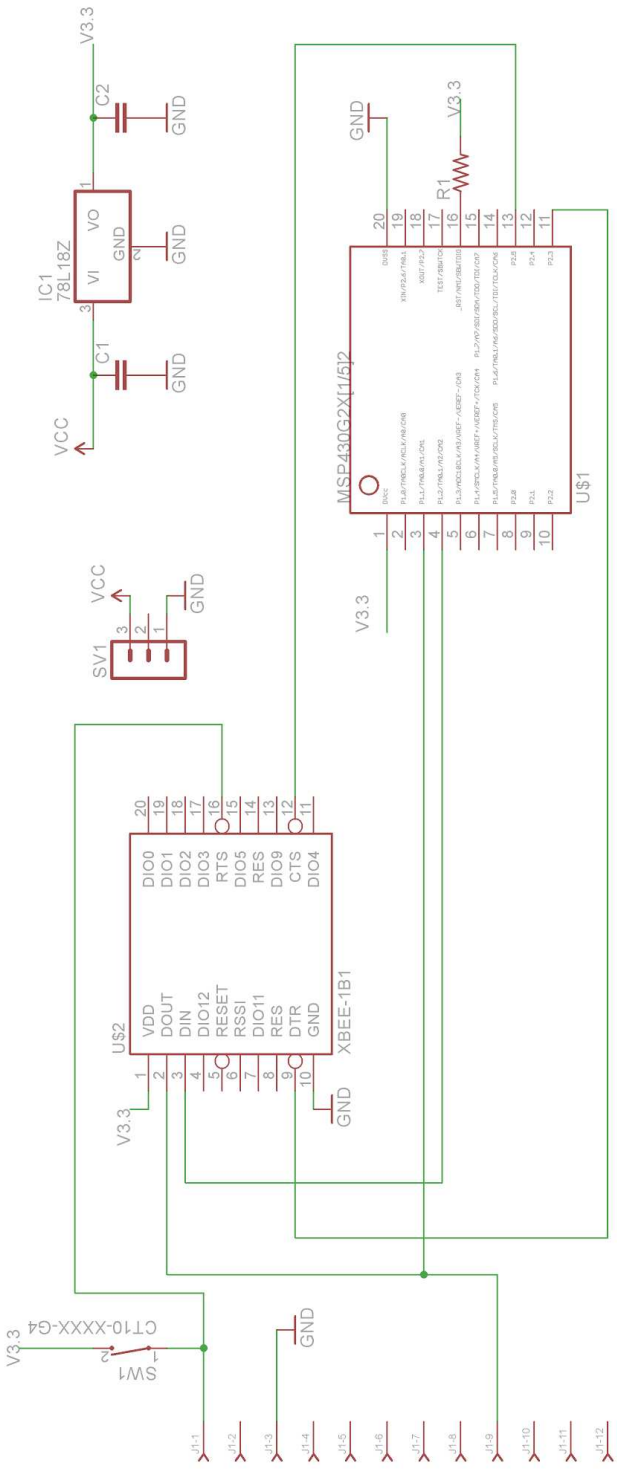


Figure 61: Schematic for Prototype Trigger

Appendix C: Microcontroller Code

Appendix D: Coordinator Code

Appendix E: Dispenser Code