4-15-2014

Position Tracking Using WiFi

Nathan Conrad
Western Michigan University, nathanrconrad@gmail.com

Position Tracking Using WiFi

By: Mitchell Woolley, Tyler Bayne, and Nathan Conrad
Advisor: Dr. Ralph Tanner
ELECTRICAL and COMPUTER ENGINEERING 4820
April 18th, 2014

Abstract

Position tracking using WiFi is needed as a more accurate method of intra-building tracking due to Global Positioning System's (GPS) lack of accuracy in buildings. The main application of this project is to provide a safety system, which may be either tracking employees within a building and allowing them to send an emergency message through a device, or to track objects throughout a building. The devices can send simple messages to a central location. These messages will include position information and which emergency has been selected. The devices consist of two buttons and two LEDs, and with a button press, the user transmits their location and depending on which button is pressed, a corresponding emergency message. The device utilizes the Arduino Uno and Arduino WiFi Shield to carry out the sending of data through a WiFi connection.

When a transmission from a device is sent, a message is sent to a Central Server that was developed. The Central Server handles the communication between the devices and a map display of the building, and shows the locations of the devices. The location coordinates are sent to another server called Redpin. This is an existing open source intra-building tracking system using WiFi, and using triangulation formulas based on signal strength of access points throughout a building, can track a device. The Redpin server then relays this information to the Central Server, and the Central Server displays the location of the device on the map. All of these components come together to form a single system to accurately track a device and display its location in real time on a map.

Acknowledgements

The project was sponsored by an anonymous company via Dr. John Kapenga, a Computer Science professor at Western Michigan University. This sponsorship included project advice, from Dr. Kapenga, and money for all the parts necessary for our design, from the anonymous company. We were also provided with two extra microcontrollers so all three team members have their own to experiment with.

Table of Contents

Summary

Global Positioning Systems (GPS) are not an accurate form of indoor position tracking due to interference with the building structure. A solution is needed that can accurately track a person or an item to a specific room. A portable device was built with an Arduino Uno and an Arduino WiFi Shield that contains a user interface and can be tracked inside a building using WiFi signal strengths. The location of the device is shown on a map of the building inside the Central Server software. This Central Server can track more than one device at a time if more devices were to be built.

The open-source Redpin software was used to calculate the positions of the devices based on the WiFi signal strengths and return the location to the Central Server for display on the map. The devices also have an emergency button that a user can press to send an alert message to the Central Server and the device's symbol on the map changes color to indicate an emergency.

The Arduino Uno was not the original microcontroller for this project. The MSP 430 and CC 3000 WiFi adapter were originally the main components for the device but were switched with the Arduino Uno and Arduino WiFi Shield during the semester due to issues with the documentation and example applications for the CC 3000. The examples provided by Texas Instruments (TI) were outdated and had incorrect steps to follow to compile the example program. In the interest of time, the Arduino Uno and Arduino WiFi Shield were substituted into the design of the device.

There were also issues with testing the device at the Parkview engineering building because the Arduino WiFi Shield can only scan and return up to 10 access points but there are more than 10 access points in the building. This conflicts with the Redpin fingerprinting system because it requires the same WiFi routers or access points to return each time it scans. Also, the

Arduino WiFi Shield cannot return the unique BSSID of the access points which prevents testing at Parkview with the access points all being named the same SSID. There isn't a way to differentiate between them.

In the end, the testing was successful at a residential home where there was only a single WiFi router. The tests listed in this report were measured at this house and show that this project was successful and able to track a device using WiFi.

Introduction

Global Positioning Systems (GPS) are not an accurate form of indoor position tracking because of obstructions to the signal [1]. Therefore, a solution is needed to track indoor positions. One solution to this problem involves a low power signaling device that will be integrated into an intra-building tracking system using WiFi.

The main application of this project will be as a safety system. It will have two possible implementations. First, the device can be worn as a badge by the users. Second, it can also be placed onto any object that needs to be tracked. The device acts as a very simple way for either a user or object position to be tracked. It is also capable of sending simple messages to a central location. The device itself consists of four main components: a microcontroller, a WiFi dongle, a battery, and a user interface. The microcontroller handles all of the logic required. The WiFi dongle is used to connect and identify information about the local WiFi network. The battery is used to power the device to allow for extended periods of use. Finally, the user interface allows users to send and receive simple messages. These four devices have been combined into a small enclosure for the initial prototype of the device.

Redpin, an existing open source intra-building tracking system using WiFi, is used to track the positions of the devices. This runs on a remote server that the devices relay information to about the access points that are in range. Redpin then uses this information to determine their locations. There is another server called the central server. This handles the sending and receiving of the simple messages from the devices. It also communicates with the Redpin server to get the locations of all of the devices. It uses this information to create a map showing the locations of each of the devices [2].

In conclusion, a low power signaling device integrated into an intra-building tracking system using WiFi solves the issue of not being able to use GPS to track positions indoors. The focus of this project was to create such a device. It consists of a microcontroller, a WiFi dongle, a battery, and a simple user interface. Redpin, an existing open source intra-building tracking system using WiFi, is used to locate each of the devices and will run on a remote server. A central server handles all of the communications between the devices and display the map of the locations of all of the devices.

Discussion

Physical Feasibility Study

The physical part of our project only consists of a simple circuit. This circuit consists of three buttons, three LEDs, a microcontroller, and a WiFi dongle. We have constructed similar circuits in previous classes, such as ECE 3550, and know that it can be done.

The most complicated part of our project is the software that is used to track the position of the devices. This will be done through the use of Redpin. Redpin has already been proven to be able to track using WiFi. Therefore, this will not be a concern.

Another piece of software that we must develop is the Central Server. This must simply be able to send and receive messages from the device and the Redpin server. The communication will be done via the WiFi network that all of the devices will be on. Communicating between devices that are on the same WiFi network is done all the time and is nothing new. The Central Server must also display a map with the locations of each of the devices on it.

Finally, the software for the microcontroller on the device must be developed. It must be able to communicate with the Central Server, be able to assess the signal strength of and connect to the access points around it, and be able to respond to button presses. We know how to make a microcontroller respond to button presses from previous courses, such as ECE 3550. We have observed the available functionality of the MSP340 and the CC3000 and have determined that we are able to get the necessary access point information that we need by looking at the APIs for each of them.

Mathematical Model

Our project does not require a mathematical model. The Redpin server uses its own mathematical model to triangulate the position of the different signal measurements compared to the stored fingerprint measurements. If our project involved writing a replacement to the Redpin server then we would have a mathematical model to do this triangulation.

Also, we are using the circuit already configured on the development board to light an LED and press a toggle button, so electrical models were not needed to be developed and the power consumption is simply measured using a multimeter.

Economic Feasibility Study

For this project, the sponsor has specified that it is to be developed as a proof of concept. This means that the design is meant to prove that the concept behind the project can be none, not to design it as a marketable product. Because of this, the economic feasibility study will not highlight future costs or a cost versus benefit approach. It is only pertinent to cover the initial costs of producing the device, noting that labor costs are not included either.

The device will consist of a MSP430 microcontroller, a CC3000 WiFi dongle, three LEDs, three push buttons, resistors, buffers, capacitors, wiring, and a housing case for these components. Also, an Android 2.2 tablet will be used for testing connection to the Redpin server. The costs of these items are given in Table 1 below, with some excess in quantities for insurance.

Item	Quantity	Cost	Total Cost
MPS430 Microcontroller	3	$12.99	$38.97
CC3000 (WiFi Dongle)	3	$35.29	$105.87
Android 2.2 Tablet	3	$50.00	$150.00
LED	10	$5.43	$54.30
Push Button	10	$2.16	$21.60
330Ω Resistor	10	$0.38	$3.80
10KΩ Resistor	10	$0.30	$3.00
541 Buffer	5	$0.88	$4.40
540 Buffer	5	$0.61	$3.05
100pF Capacitor	5	$0.61	$3.05
Wiring Kit	1	$4.94	$4.94
Housing Case	1	$5.00	$5.00
		Total Cost	**$397.98**

Table 1. Item list with cost

From Table 1, the total cost of the project will be 397.98. This is a reasonable production cost for parts and will give a nice margin for profit if the device goes to market, especially when considering the total cost for one device will be lower than the total cost given in Table 1 because lower quantities will be used. Therefore, this project is economically feasible.

Patent Search

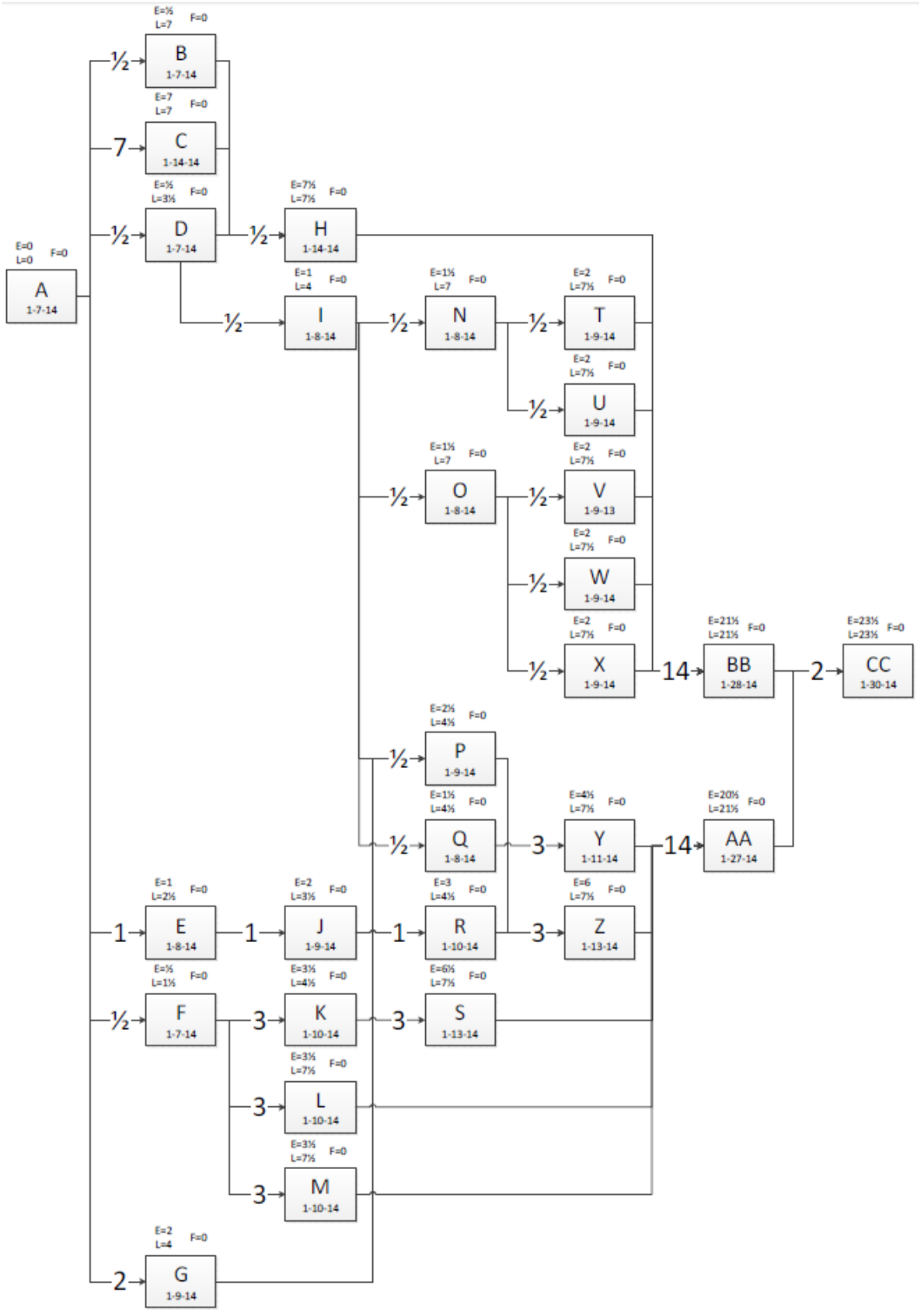
There have been a few concepts for indoor location tracking patented. Patent [3] describes a concept where mobile devices are tracked using a particle filter and orientation data from a gyroscope and/or an accelerometer. The map used for determining position could be designed in two different ways. The first way uses walls as an obstruction to the user's movement while the second option uses rails as the path the user must walk on. The map has particles associated with different locations and are given probabilities that the user is at that location when the orientation data is analyzed.

Another patent, [4], not only calculates the indoor position of an object using wireless signal strength but also uses non-RF (radio frequency) factors like GPS quality, operating system, and device type information. The patent application refers to each permanently positioned device as a tile with the non-RF factors. The design uses a fingerprint system that is trained for the RF factors and then includes the non-RF factors to get a more accurate localization. A test dataset is used to determine the percent error of the calculated distance from the device and the tile. A threshold is set for this percent error to determine whether the non-RF factors are necessary or if the current reading is accurate enough.

Patent [5] involves a mapping device, potentially a laser range finder, which scans the area to build a depth sub-map of all the nearby obstacles at a certain time. This information is projected onto an image plane which is then combined with all of the other sub-maps to create one general map of the entire area. From there, the general map is converted to a coordinate system. A future test device uses its own scanning data and compares it to the map collected previously to determine its position inside a building.

Our design involves using WiFi signal strength to determine the location of a device whereas the above designs use particle filters, orientation data, laser range finders, and non-RF factors. Therefore, we do not believe our design infringes on existing patents.

Critical Path Network



Events

Event A – Start of the process

Event B - Download official Redpin app to Android tablet for testing is complete

Event C - Acquire a digital map of Parkview and a few other buildings on campus is complete

Event D – Run the Redpin server is complete

Event E – Connect LEDs and buttons to create a circuit is complete

Event F - Attach CC3000 to Microcontroller is complete

Event G - Write a method that waits for an input is complete

Event H - Test the official Redpin app with the digital building maps is complete

Event I - Write a method to make the central server communicate with the Redpin server is complete

Event J - Attach the user interface to the microcontroller is complete

Event K - Write a method that searches for available WiFi access points with CC3000 is complete

Event L - Write a method that measures signal strengths of access points is complete

Event M - Write a method that makes the device act like an access point is complete

Event N - Write a method that uploads a map to the Redpin server is complete

Event O - Write a method that sends a fingerprint to the Redpin server is complete

Event P - Write a method that responds to a button press on the user interface is complete

Event Q - Write a method that responds to location information from microcontroller is complete

Event R - Write a method that lights up an LED on the microcontroller when a user interface button is pressed is complete

Event S – Write a method that connects to WiFi access point with CC3000 is complete

Event T - Write a method that gets the list of maps from the Redpin server is complete

Event U - Write a method that removes a map from the Redpin server is complete

Event V - Write a method that sends a measurement and requests location is complete

Event W - Write a method that gets all available locations from the Redpin server is complete

Event X - Write a method that removes a location from the Redpin server is complete

Event Y - Write a method that sends location information to Central Server is complete

Event Z - Write a method that tells Central Server a button was pressed is complete

Event AA - Combine the UI proof of concepts and the microcontroller proof of concepts to make the device is complete

Event BB - Combine proof of concept Central Server device communication methods with the Redpin server communication methods is complete

Event CC – Completion of the process

Activities

1. A-B [Downloading official Redpin app to Android tablet for testing]: One-half day
2. A-C [Acquiring a digital map of Parkview and a few other buildings on campus]: Seven days
3. A-D [Running the Redpin server]: One-half day
4. A-E [Connecting LEDs and buttons to create a circuit]: One day
5. A-F [Attaching CC3000 to Microcontroller]: One-half day
6. A-G [Writing a method that waits for an input]: Two days
7. B-H [Testing the official Redpin app with the digital building maps]: One-half day
8. C-H [Testing the official Redpin app with the digital building maps]: One-half day
9. D-H [Testing the official Redpin app with the digital building maps]: One-half day
10. D-I [Writing a method to make the Central Server communicate with the Redpin server]: One-half day
11. E-J [Attaching the user interface to the microcontroller]: One day
12. F-K [Writing a method that searches for available WiFi access points with CC3000]: Three days
13. F-L [Writing a method that measures signal strengths of access points]: Three days
14. F-M [Writing a method that makes the device act like an access point]: Three days
15. G-P [Writing a method that responds to a button press on the user interface]: One-half day
16. H-BB [Combining proof of concept Central Server device communication methods with the Redpin server communication methods]: Fourteen days
17. I-N [Writing a method that uploads a map to the Redpin server]: One-half day

18. I-O [Writing a method that sends a fingerprint to the Redpin server]: One-half day
19. I-P [Writing a method that responds to a button press on the user interface]: One-half day
20. I-Q [Writing a method that responds to location information from microcontroller]: One-half day
21. J-R [Writing a method that lights up an LED on the microcontroller when a user interface button is pressed]: One day
22. K-S [Writing a method that connects to WiFi access point with CC3000]: Three days
23. L-AA [Combining the UI proof of concepts and the microcontroller proof of concepts to make the device]: Fourteen days
24. M-AA [Combining the UI proof of concepts and the microcontroller proof of concepts to make the device]: Fourteen days
25. N-T [Writing a method that gets the list of maps from the Redpin server]: One-half day
26. N-U [Writing a method that removes a map from the Redpin server]: One-half day
27. O-V [Writing a method that sends a measurement to the Redpin server and requests location]: One-half day
28. O-W [Writing a method that gets all available locations from the Redpin server]: One-half day
29. O-X [Writing a method that removes a location from the Redpin server]: One-half day
30. P-Z [Writing a method that tells Central Server a button was pressed]: Three days
31. Q-Y [Writing a method that sends location information to Central Server]: Three days
32. R-Z [Writing a method that tells Central Server a button was pressed]: Three days
33. S-AA [Combining the UI proof of concepts and the microcontroller proof of concepts to make the device]: Fourteen days

34. T-BB [Combining proof of concept Central Server device communication methods with the Redpin server communication methods]: Fourteen days
35. U-BB [Combining proof of concept Central Server device communication methods with the Redpin server communication methods]: Fourteen days
36. V-BB [Combining proof of concept Central Server device communication methods with the Redpin server communication methods]: Fourteen days
37. W-BB [Combining proof of concept Central Server device communication methods with the Redpin server communication methods]: Fourteen days
38. X-BB [Combining proof of concept Central Server device communication methods with the Redpin server communication methods]: Fourteen days
39. Y-AA [Combining the UI proof of concepts and the microcontroller proof of concepts to make the device]: Fourteen days
40. Z-AA [Combining the UI proof of concepts and the microcontroller proof of concepts to make the device]: Fourteen days
41. BB-CC [Completing the process and report]: Two days
42. AA-CC [Completing the process and report]: Two days

Expected Completion Time

Our critical path is A-C-H-BB-CC.

The formula for expected completion time is:

$$T_e = \frac{Optimistic\ Time + 4 * (Most\ likely\ time) + Pessimistic\ Time}{4}$$

The optimistic time it will take for our project to be complete is 3 + ¼ + 7 + 1 = 11.25 days.

The most likely time it will take for our project to be complete is 7 + ½ + 14 + 2 = 23.5 days.

The pessimistic time it will take for our project to be complete is 14 + 2 + 21 + 10 = 47 days.

Plugging these numbers into our formula gets:

$$T_e = \frac{(11.25 + 4(23.5) + 47)}{4} = 38.06\ days.$$

To find the probability of completing our project 3 weeks behind schedule can be calculated using the formula:

$$Z = \frac{T_S - T_C}{\sigma_{\beta p}}$$

where

$T_s = 103\ days\ (the\ total\ number\ of\ days\ during\ the\ semester)$

$T_c = 23.5\ days\ (critical\ path\ time)$

$\sigma_{\beta p} = 3.337\ days\ (standard\ deviation\ along\ critical\ path)$

Plugging these numbers into the formula gets Z = 23.82 (completing the project three weeks behind schedule).

Doing the same thing with $T_s = 82$ (completing the project on time) gets Z = 17.531.

Lastly, $T_s = 75$ (completing the project one week ahead of time) gets Z = 15.433.

Since these Z values are very high, our project has a high probability of being completed in those time ranges.

Precedence Table

			Predecessor Level		
Specification	**Major Part #**	**Description**	**High**	**Medium**	**Low**
a		Size			
b		Low Power Consumption			
c		Portability			
d		Communication			
e		Cost			
	1	Central Server		d	
	2	User Interface		3. a	
	3	Device	a, 2, b, c, e	1, d	

Precedence Matrix

DESIGN ORDER		a	b	c	d	e	1	2	3
	a	*							
	b		*						
	c			*					
	d				*				
	e					*			
	1				M		*		
	2	M						*	M
	3	H	H	H	M	H	M	H	*

Specifications

1. Physical Characteristics

    1.1. Size

        1.1.1. The device should be relatively small. This is a **qualitative** specification and a **requirement**.

        1.1.2. An ambitious goal for the project is to have it be small enough so that it can be worn as a badge by a user. This is a **qualitative** specification and a **goal**.

2. Functionality

    2.1. Power Consumption

        2.1.1. The device should function with a low amount of power. It is going to be powered by a battery. Therefore, to help make the battery life as long as possible, the device should consume as little power as possible. This is a **qualitative** specification and a **requirement**.

        2.1.2. The device should have a low power mode. This will help to further improve the power consumption of the device. When it is not reporting its position, it should be in a low power state. This is a **qualitative** specification and a **goal**.

    2.2. Portability

        2.2.1. The device should be portable. This is a **qualitative** specification and a **requirement**.

        2.2.2. The device should be able to operate for long periods of time. This is a **qualitative** specification and a **requirement**.

    2.3. Communication

2.3.1. The device should be able to communicate on both the 802.11 b and g protocols. These are the most common and popular protocols in use today. Therefore, to make the device be able to be used in as many places as possible, it should be able to communicate using these protocols. This is a **quantitative** specification and a **requirement**.

2.3.2. A goal for this project is to also allow the device to communicate on the 802.11 n protocol as well. This would enable a wider range of communication possibilities. This is a **quantitative** specification and a **goal**.

2.4. User Interface

2.4.1. The device should allow users to send simple messages to a central location. It should also be able to receive simple messages from that same central location. Also, it should have a simple way of displaying the received message to the user. This is a **qualitative** specification and a **requirement**.

2.5. Position Tracking

2.5.1. The device should periodically report its position so that its approximate position is known at all times. This is a **qualitative** specification and a **requirement**.

2.5.2. The position of each of the devices should be shown on a map. This is a **qualitative** specification and a **requirement**.

3. Other

3.1. Cost

3.1.1. The devices will most likely be used by customers in large numbers. Therefore, the device should be inexpensive so that it is inexpensive to reproduce. This is a **qualitative** specification and a **requirement**.

Design Concept

Our current design concept for this project consists of three main components: the devices, the Redpin server, and the central server. How we plan for each of these components to meet the specifications listed on the previous page will be discussed in detail in the following paragraphs.

To begin, our design concept for the devices will be discussed. The device itself consists of four main components: a microcontroller, a WiFi dongle, a battery, and a user interface. The microcontroller handles all of the logic required. The WiFi dongle is used to connect and identify information about the local WiFi network. The battery is used to power the device to allow for extended periods of use. Finally, the user interface allows users to send and receive simple messages. These four devices are combined together for the initial prototype of the device. This is shown below in Figure 1. Having only these four components keeps the size of the device to a minimum while still being able to perform all of the necessary operations.

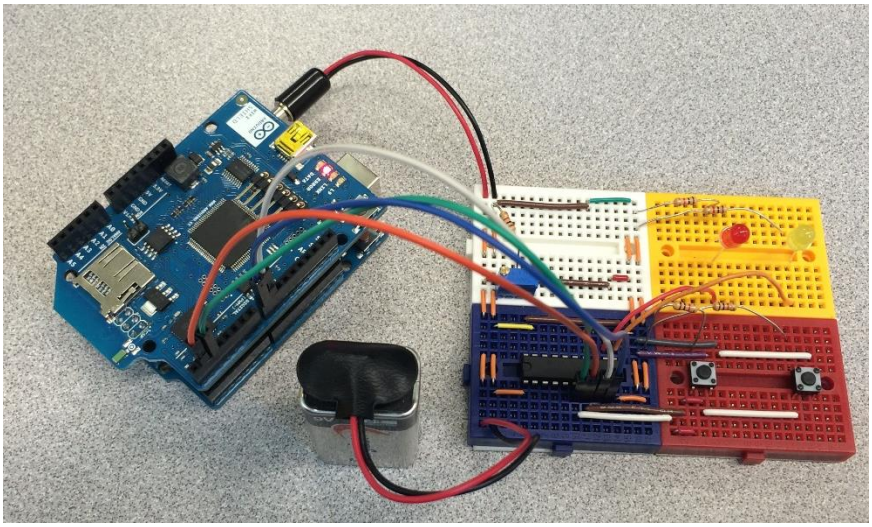


Figure 1. The completed device and user interface

The microcontroller that has been chosen for this project is the Arduino Uno. The WiFi dongle that is used is the Arduino WiFi Shield, which is easily interfaced with the Arduino Uno. The Arduino WiFi Shield also has an ultra-low leakage switch that can be controlled through the microcontroller. This has allowed a low power mode to be implemented easily and helped us to minimize power consumption. It is capable of communicating on both 802.11 b and g protocols. Both of these components are inexpensive and relatively small in size. They are also geared towards minimizing the amount of power consumed. This therefore helped to maximize the battery life of the device. A 9 volt battery is used. These batteries are common and have a low discharge rate.

The device user interface has two buttons on it. Both of these button meanings can be defined by the users. For example, in a safety system, one button could indicate that there is danger in the location where the button was pressed and that people should stay clear. The other button could indicate that help is needed at the location of the button press. Whenever one of these buttons is pressed, all of the other devices on the same WiFi network will be notified through a simple LED system. There are three LEDs - one corresponding to each of the buttons. Whenever one of the buttons on a device is pressed, the corresponding LED will be lit on each of the other devices. A schematic of the completed device is shown in Figure 2 below.

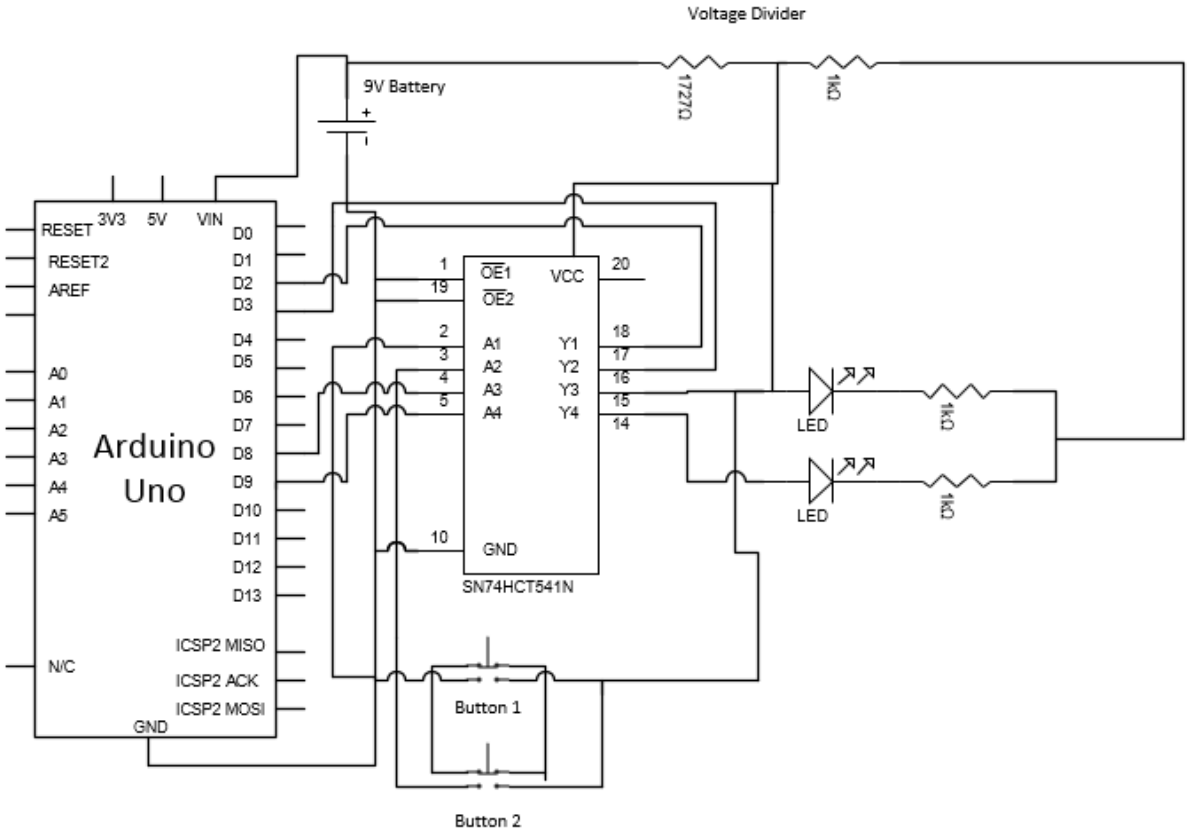


Figure 2. Device Schematic

Our design concept for the Redpin server will now be discussed. In order to determine the position of the device, we are using Redpin. Redpin is an existing open source intra-building tracking system using WiFi. It is a fingerprint-based system and therefore provides symbolic references as opposed to geographic coordinates. There are two components to the Redpin system - a locator (server) and “sniffers” (remote devices). The sniffers are responsible for collecting information about any WiFi access points that are in range and using this information to create a fingerprint. The locator stores all of the measured fingerprints and contains the logic to locate the devices. To use the sniffer, the user first chooses a map. That map is then loaded onto the sniffer and the locator. The user can then add new locations. When a new location is added, the sniffer enters a phase where it collects information about all of the access points that

are in range. Once this phase is completed, it creates a fingerprint from this data and then sends this data to the locator, where it is stored. This information is then used by the locator to determine the location of the device.

The locator in this project is a server that is running the Redpin server software. The sniffers are each of the devices. The devices are responsible for determining the characteristics of the access points that they are in range of. These characteristics include the service set identification (SSID), the media access control (MAC) address, and the strength of each of the signals from the different access points. The devices then send this information to the Redpin server.

Our design concept for the central server will now be discussed. In addition to the Redpin server, there is a separate server called the central server that is responsible for handling all of the communication related to the button presses and the LEDs. It is also responsible for generating the map that shows the location of all of the devices. The devices send a signal to the central server whenever a button is pressed. The central server then sends a signal to all of the other devices saying the button has been pressed so that the devices can light the appropriate LED. The central server then generates the map showing the position of all of the devices by getting the location data from the Redpin server. Also, whenever a button press signal is received, the central server indicates on the map which button was pressed on which device. A block diagram of the communication flow between each of the components is shown below in Figure 3.

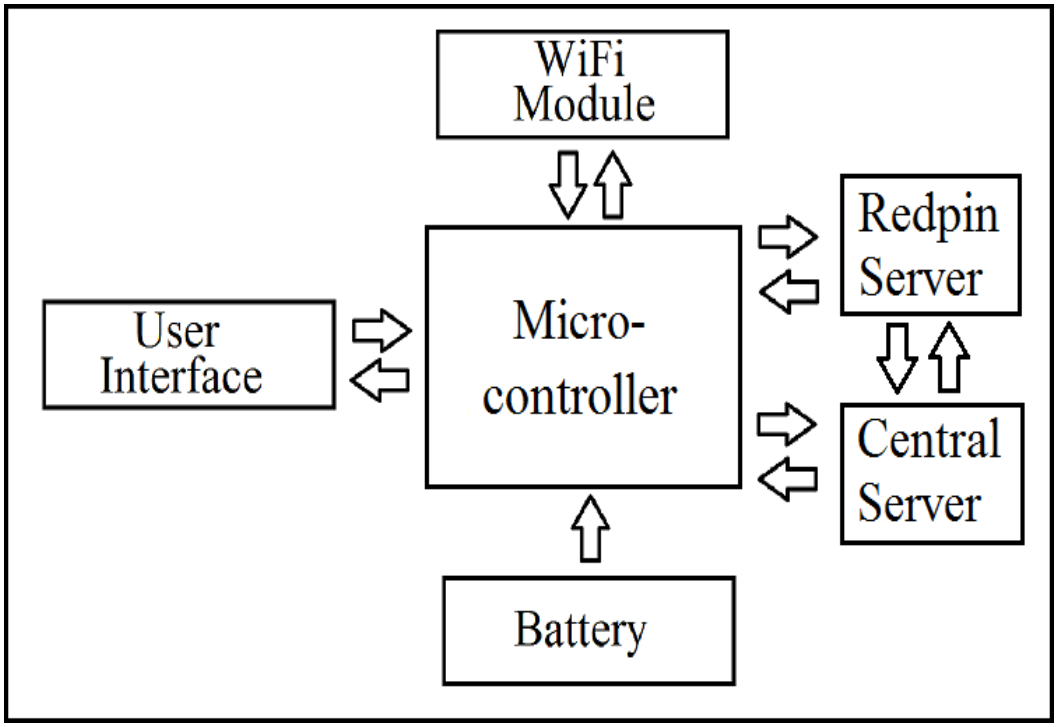


Figure 3. Block diagram

Alternatives

An alternative we found for the Redpin server is called TeroM. It has an Android app like the Redpin app and using the same fingerprinting location strategy. However, TeroM has less support and less of a development community than Redpin. Redpin has both an Android app and an iOS app which is another reason why we chose Redpin over TeroM.

The alternative we researched to the Arduino Uno would be to use an FPGA. However, we discovered that FPGAs are meant for high-end applications and thus use a lot of power whereas Arduinos are meant for low-powered devices but are not as powerful. For our project, we do not need a high-end microcontroller because it will be doing a low amount of processing. Also, low power consumption was a specification which is what the Arduino Uno is built to provide.

Lastly, an alternative for the Arduino WiFi Shield module is to use the Xbee Series WiFi modules. These WiFi modules are able to connect to the Arduino Uno in a similar way as the Arduino WiFi Shield ones but there isn't as good of documentation as the Arduino WiFi Shield.

Discussion of Constraining Factors

There are several constraining factors that could affect our project design. These include economic, health and safety, environmental, sustainability, ethics, social impact, manufacturability, and political constraints. Each of these will be discussed in detail below.

Economic Constraints

One of the specifications is that the cost of this project should be relatively low. This is setting a constraint on the price of the different components in our project and we are required to make an inexpensive device. We use an inexpensive microcontroller and other components for our physical device to keep the cost relatively low.

Health and Safety

The physical part of our project is only the small device. Therefore, the only issue for health and safety is to make sure that our device is well contained so that it does not give an electric shock to users after it is modified to fit on an employee identification badge. The device is enclosed in a case to prevent the electric shock described above and to make sure the device's wires do not accidentally get shifted when handled by a user.

Environmental

There are no environmental constraining factors for our project because our project is a proof of concept device and isn't an item that will be installed somewhere.

Sustainability

Our project will be delivered along with documentation for the Central Server to allow for our sponsor to easily modify our proof of concept device to be implemented as a way to track employees and items at their company. This documentation will specify the API of the Central Server as well as documenting the different parts of the software. Also, we have thoroughly

comment our Central Server code for easy understanding. As for the physical device portion, we have created our device so that there is a little bit of extra room in the case making it easy to understand and modify.

Ethics

Since our project involves a lot of software, we have to be aware of accidentally copying too much sample code from various internet sources like StackOverflow. We use the internet for reference of how to write code but we will not copy large chunks of code and use it in our software.

Social Impact

There is not a social impact constraining factor for our project because it is a proof of concept device. If our project was to deliver a final version that can be attached to an employee identification badge then there would be social impact constraining factors because many employees would be using project.

Manufacturability

Our project does not have a manufacturability constraint because we are delivering a proof of concept device to our sponsor. This isn't a device that will be ready for manufacturing. However, we will create our device to be as small as possible to make it easier to modify and manufactured to fit on an employee identification badge.

Political

There are no political constraining factors for our project because our project is a proof of concept that will be used privately by a company for internal use. If we were designing a device that will be used by the public then we would have political constraining factors to take into account for our project.

Plan to Measure Performance

In order to measure the correct performance of our final project, a plan was devised. This plan consisted of several tests that demonstrated the working project. These tests are necessary to show that the final project will function according to the initial specifications.

First, we proved that the device can report its location accurate to a single room. In order to do this we started up the device in a room, and then with a laptop running the central server, the laptop communicated with the device relaying its position.

Another plan that was necessary was to have the device relay a type of emergency to the central server. Similar to the previous test, this test involved pushing a button on the device corresponding to a type of emergency. This signal then relayed to the central server and a message was presented on a laptop connected to the central server with the corresponding emergency. The measurement of this test was seeing the correct emergency code on the central server that the device has transmitted.

Methods

We ended up using a few different programs to design our project. The Central Server was programmed in Java using the NetBeans Integerated Development Environment (IDE). This allowed us to drag and drop items onto a palette to create the user interface and then program the specific elements of the server. NetBeans also includes a debugger to aid in the discovery and correcting of bugs in the code.

The Arduino Uno's software was programmed in Arduino's version of C using the official Arduino IDE. Similar to NetBeans, the Arduino IDE included a debugger and allowed us to download the code to the Arduino microcontroller.

Lastly, a version control system was used to keep records of the various versions of the Central Server program throughout the semester as well as allowing the code to be easily shared between team members. A private BitBucket repository was created and linked to the Git repository on each team member's computer to facilitate the cloud storage of the repository.

Performance

The plan to measure performance, from above, was followed. Ten access points were set up in a residence on both the first floor and the basement. The reason for this was to get the signals coming from various directions instead of just one a single plane. These access points are shown in Figure 4 below.

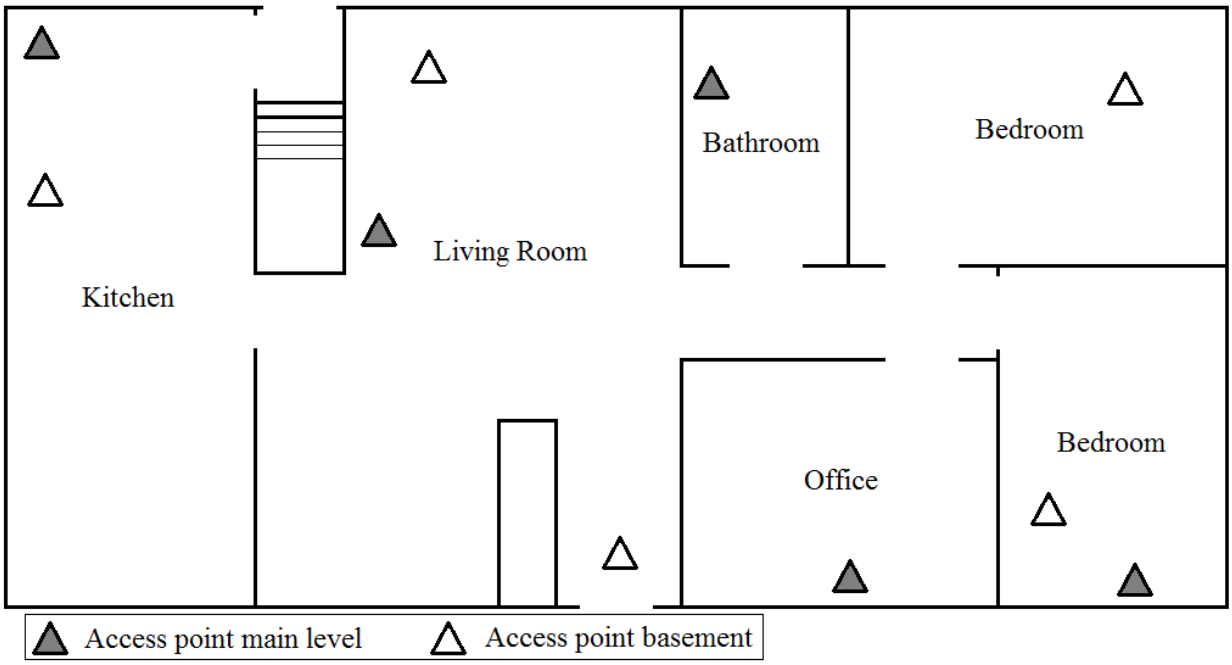


Figure 4. Access point locations

Next, fingerprints measurements were taken with the device to calibrate the system. In this step, one teammate carried the device around each room as the other teammate clicked on that room on the Central Server's map. This initiates a saved fingerprint with the current measurements at that location. These measurements were taken for the entire perimeter of the room as well as various locations in the center to make them as accurate as possible. This process was repeated for the other rooms until there were three fingerprints saved 20-25 feet apart. These locations are shown in Figure 5 below.

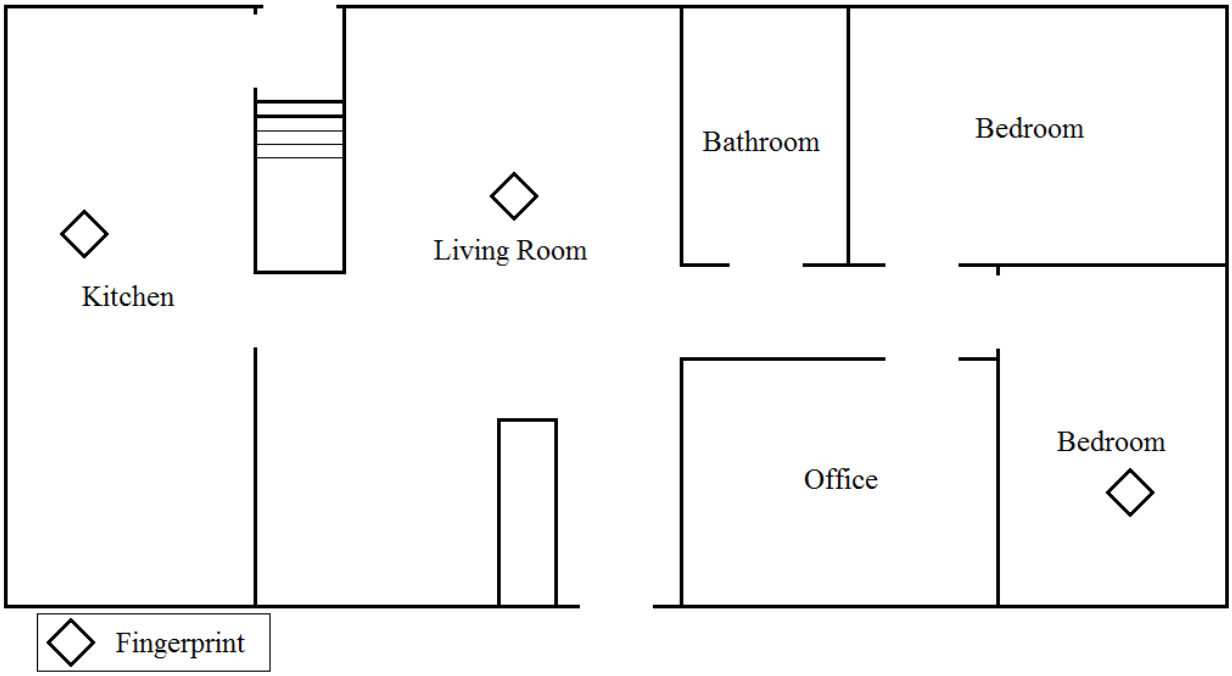


Figure 5. Fingerprint locations

A picture with the fingerprint locations and the access points is shown below in Figure 6.

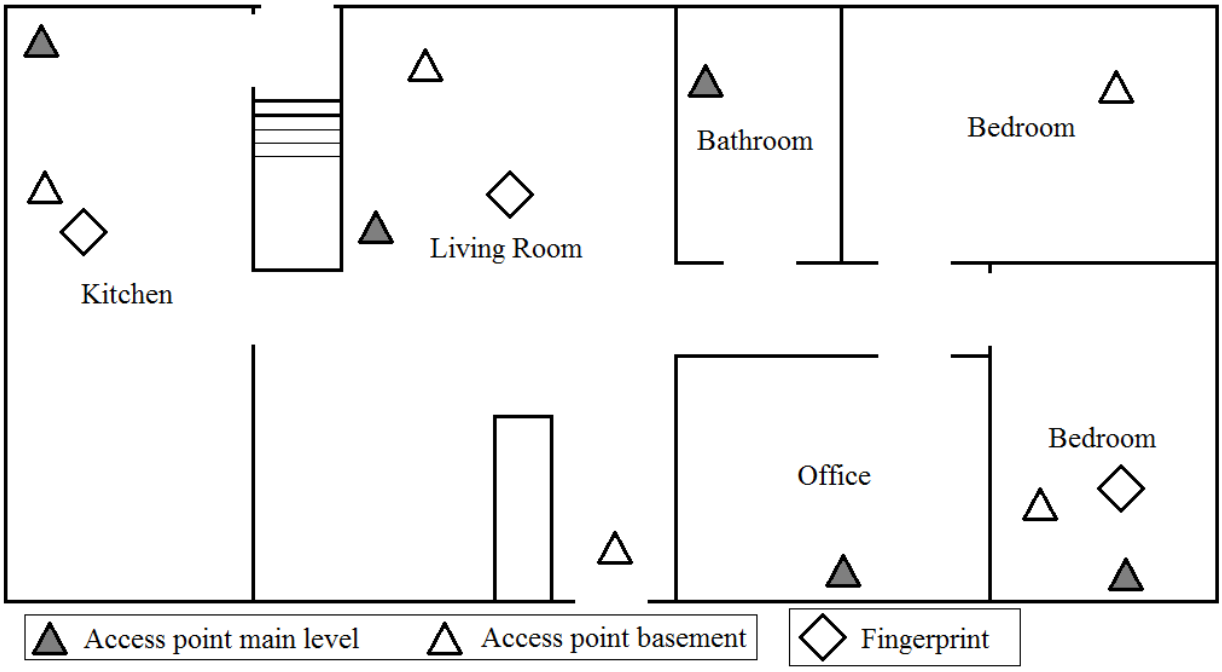


Figure 6. Access point and fingerprint locations

After the calibration phase was complete, the next step was to measure the performance of the tracking. One teammate walked around the house holding the device while the other teammate sat at the laptop running the Central Server. As the device entered the fingerprinted locations, the teammate would slow walk around the room and then exit after there had been ten measurements taken. The results of this test are shown in Table 2 below.

Location	Accuracy	Percentage
Kitchen	9/10	90%
Living Room	7/10	70%
Bedroom	9/10	90%
Average	8.3/10	83%

Table 2. Results of house test

As you can see, the device was very accurate in the far rooms but a less accurate in the middle room. The accuracy is predicted to increase as more access points are used and more calibration fingerprints are taken. There isn't any data to support this other than the observations our team was making as we were testing the device over the course of several days.

The performance of the emergency button press was not fully tested because it was not completely finished in time. The Arduino was able to detect a button press from the user interface and attempt to send a signal to the Central Server but that signal transfer was never fully working. On the other side, the Central Server was set up to receive the signal and change the color of the device's circle on the map. This was tested and the picture of the Central Server before the received emergency signal for device D is seen in Figure 7 and the picture after the received emergency signal is seen in Figure 8. You can see that device D changed from a green circle to a red circle to indicate an emergency in that room.

Figure 7. Before an emergency signal is received for device D

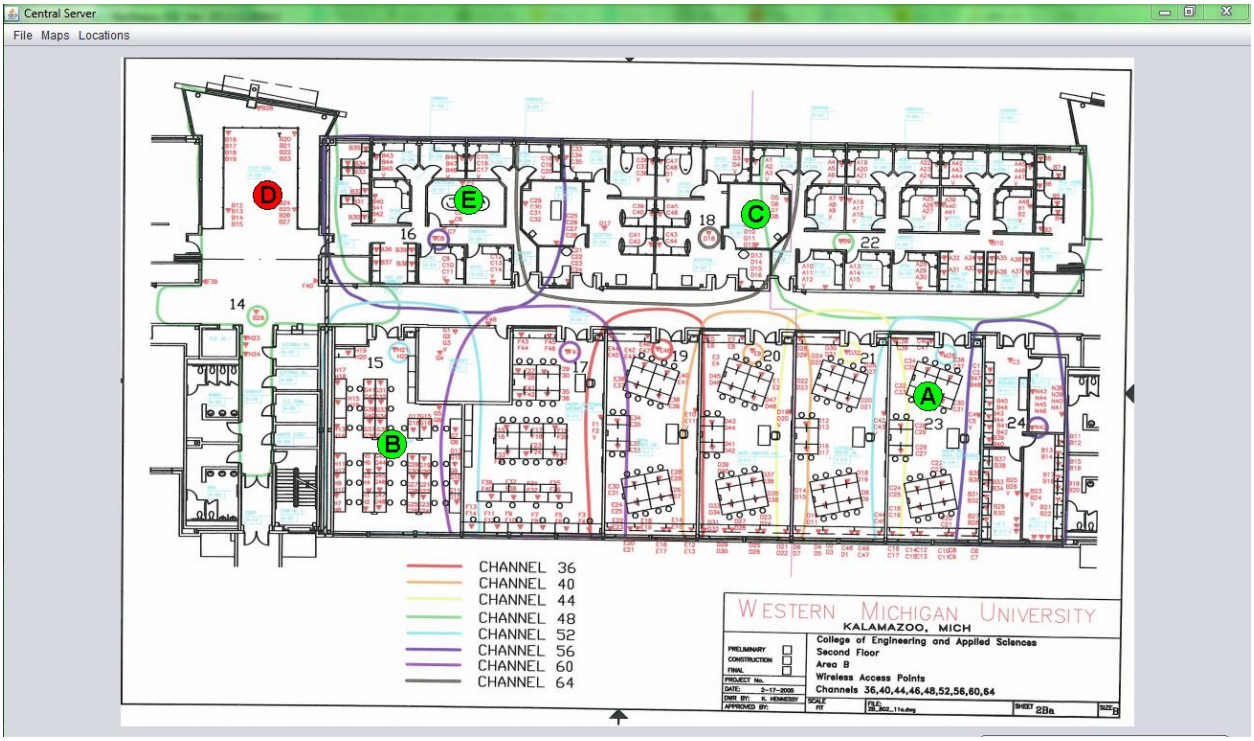


Figure 8. After an emergency signal is received for device D

Unexpected Problems

There were many problems encountered during this semester. The first major problem was with the original microcontroller, MSP 430, and the original WiFi adapter, CC 3000. The documentation and example programs for the CC 3000 WiFi were very difficult to learn from and get working. There were many outdated and incorrect steps in the documentation of how to configure the example program. The program was finally able to be compiled and successfully connect to a WiFi network but it was too late. The Arduino Uno had already been ordered and was very far along on the programming. It was best to continue with what was already working and use that for the project.

Another major issue was that the Arduino WiFi shield cannot read the unique identifier BSSID of an access point. This is a problem when testing at the Parkview building because the WiFi network has many different access points with the same SSID. The Arduino program could not differentiate between the various signal strengths and therefore could not record an accurate fingerprint. This issue was worked around by using separate access points that were set up at optimal positions in Parkview and the building's WiFi network filtered out of the calibration and measurements. The SSID of the networks was then passed as the unique identifier. This worked as long as the SSIDs were unique between the access points.

This appeared to work at first but after significant testing it was found that the measurements were inconsistent and fairly random. The cause of this was found to be a setting in the Arduino WiFi Shield library that limits the access point scan to only 10 networks. Since this Redpin is based off of a fingerprint system, the same networks have to always be sent to the server for the reading to be accurate. The Arduino would pick up different access points at

different times. It was never consistent. To work around this problem, the device was then tested at a residential home where there was a single WiFi network. At this location, the device performed very well and was very consistent with its calculations.

Budget Table

The final cost for this project was $383.27. This was still an inexpensive project because most of the money came from finding a tablet that would work with the outdated Redpin app and a lot of money came from the MSP430 and CC 3000 parts that ended up not being used. You can see the complete budget table in Table 3 below.

Item	Part Number	Quantity	Cost	Total Cost
MPS430 Microcontroller	MSP430G2x53	3	$12.99	$38.97
CC3000 (WiFi Dongle)	CC3000BOOST	3	$35.29	$105.87
Ematic Twig Android 2.2 Tablet	N/A	1	$19.99	$19.99
Polaroid Android 2.2 Tablet	N/A	1	$32.00	$32.00
Cruz Android 2.2 Tablet	N/A	1	$50.00	$50.00
Arduino Uno	A000066	1	$35.00	$35.00
Arduino WiFi Shield	B009M8BU50	1	$89.95	$89.95
LED	NTE3164	2	$0.40	$0.80
Push Button	CR40-565	2	$1.59	$3.18
Potentiometer	3296W-1-103LF	1	$1.09	$1.09
5KΩ Resistor	CFR-25JB-52-1K	2	$0.30	$0.60
541 Buffer	SN74HCT541N	1	$0.88	$0.88
Wiring Kit	N/A	1	$4.94	$4.94
			Total Cost	**$383.27**

Table 3. Budget table

Low Power Calculations

Although low power modes were not successfully implemented into the coding of the device, all major components of the Arduino Uno and the Arduino WiFi Shield had low power capabilities. Therefore, as seen in Figure 1 of Appendix A, calculations were made for the average current draw of the system (top half), and also for the average current draw if low power modes had been implemented (bottom half).

The tables in Figure A1 shows the component to be evaluated for either the Arduino Uno or the Arduino WiFi Shield, its respective power consumption in mA, the duty percent from a total eight hour period, and the calculated total power consumption for that component. The devices are as follows: Atmega328 is the microcontroller on the Arduino Uno, AT32UC3A1256 is the microcontroller on the Arduino WiFi Shield, and the HDG104 WLAN is the WiFi connection chip on the Arduino WiFi Shield. The power consumption values were obtained from each component's respective datasheet. The duty percent was calculated based on how long each device would be used for the eight hour day. The smaller sub-table was used in calculating the HDG104 WLAN power consumption because its power consumption is based on the number of transmissions as well as the duration of these transmissions. The total is based on the duty percent and the power consumption, which is summed with all the totals to yield the average current draw, or the constant power consumption over the eight hour period.

The first table, representing the power consumption of the system built for the project, shows the Atmega328 and AT32UC3A1256 both at 100% duty because they are in constant active mode drawing 5.2 mA and 20 mA respectively. The HDG104 WLAN has a power consumption of 178 mA when transmitting and when not transmitting it is in sleep mode with a

power consumption of 0.06 mA. The duty percent for transmission was found by using the number of transmits per minute (2) and the duration of a transmit (1 second) resulting in 3.33% duty. The sleep mode was then simply whenever the HDG104 WLAN was not transmitting, so a 96.67% duty. Adding the totals resulted in an average current draw of 31.19 mA.

The second table shows what the power consumption values would have been if the components would have been put into their low power modes. The Atmega328 and the AT32UC3A1256 both have an idle state they can be put into when it is not necessary for them to be active. The assumption was made that when the system is not transmitting its location, the microcontrollers could be put into their idle states without affecting functionality. To give enough time for the complete operation of a transmission to be used, it was assumed that the Atmega328 would need three seconds and the AT32UC3A1256 would need 2 at the most. This means that the Atmega328 would be in active mode for 2,880 seconds which is 10% of the eight hour period. Also, the AT32UC3A1256 would be in active mode for 1,920 seconds which is 6.67% of the eight hour period. The power consumption analysis for the HDG104 WLAN is the same as the first table. Finally, the totals are summed resulting in an average current draw of 23.05 mA for the system's low power mode.

Comparing Performance to Original Specifications

One of the specifications was that the device should be relatively small. This specification was roughly 85% completed. The device that was constructed is small enough to fit in a person's hand, but is not small enough so that it could be comfortably worn as a badge. A second specification was that the device should consume low amounts of power. This specification was roughly 50% completed. A third specification was that the device should be portable. This specification was 100% completed. Our device is battery operated and does not require a physical connection to any other device.

A fourth specification was that the device should have long operating life. This specification was roughly 75% completed. Low power components were still chosen, but the MSP430/CC3000 were unable to be used. Therefore, the device was not as low power as it could have been. A fifth specification was that the device should communicate using the 802.11 b/g protocols. This specification was 100% completed. The device can connect to a WiFi network and communicate over it using the 802.11 b/g protocols. A sixth specification was that the device should be able to send an emergency message. This specification was roughly 50% completed. The Arduino is able to respond to button presses and the Central Server is capable of indicating which device has emergency. However, the Arduino being able to tell Central Server that a button press occurred was unable to be completed.

A seventh specification was that the device should periodically report its location. This specification was 100% completed. The device is capable of informing the Central Server of its location at specified time intervals. An eight specification was that the locations of the devices should be shown on the map. This specification was 100% completed. After the device reports its location to the Central Server, the Central Server updates the map to show the location of the

device. A ninth and final specification was that the device should be relatively inexpensive. This specification was roughly 50% completed. The device is relatively inexpensive, but it is more expensive than if the MSP430/CC3000 were able to be used. Overall, about 75% of the specifications were completed.

Recommendations

The Position Tracking Using WiFi project presented here was designed as a proof of concept. Although the concept and application of this system has been shown to be plausible and effective, there is still room for improvements that could be made in the future. There are several recommendations for progressing with this project including optimizing device size, improving communication, adding useful components, and modifying the code.

One recommendation is to make the device smaller. Along with the Arduino Uno and Arduino WiFi Shield, the device currently consists of two LEDs, two push buttons, a buffer chip, and resistors, all of which are through-hole design. This is not an ideal design for what is hoped to be a product resembling a badge for an employee. In order to improve this, one could design a printed circuit board that consists of only the necessary components, all with a slimmer design. The Arduino Uno and Arduino WiFi Shield both consist of many components that are unnecessary to the functionality of the project. These could be removed and the placement of all the components could be optimized for a sleeker design.

Another recommendation is to add two-way communication between the devices and the Central Server. Currently the devices can only transmit their position to the central server, but ideally the Central Server would also be able to send messages to the devices alerting them of button presses from other devices. This is a realistic goal because in a real life situation, it would be necessary for all of the devices to be alerted of a button press from a specific device. This feature would greatly contribute to the safety aspect of the project.

Continuing on two-way communication, a third recommendation is to add extra components to the device. If the system had two-way communication, and the devices can

receive messages from the Central Server, the devices would need a way to display this message. This could be done using a Liquid Crystal Display (LCD). Each device would have an LCD that when it received a message, such as a location and type of emergency from another device, that message could be written across the LCD for the user to see. Also, because the user may not be actively looking at the device display, another component that could be used is a vibration motor to output a short vibration when a message is received alerting the user to check the display.

Lastly, it is recommended that the Arduino WiFi Shield's firmware and libraries be modified to allow more than 10 access points per scan. Currently the Arduino WiFi Shield is hard-coded to only allow for 10 access points to be used. If more than 10 could be used, the calculations could be more accurate because it would have more data to use. Also, the Arduino WiFi Shield's firmware does not allow the Shield to see an access points BSSID, but it can see the network's SSID. It would be helpful to have the BSSID because in some situations, like that of at WMU's Parkview Engineering campus building, the access points all have the same SSID. Because of this, the Arduino WiFi Shield cannot distinguish between the access points and signal strengths taken from access points are not consistent, making the triangulation from the Redpin server incorrect and inconsistent.

Deliverables

1. The device.

The device consists of the microcontroller, WiFi dongle, and a battery. The goal is for it to be able to fit on an employee's identification badge so we designed it as small as possible. The microcontroller uses the WiFi dongle to read the signal strengths of the nearby WiFi networks and pass that information, along with the SSIDs, to the Redpin server. Also, there is an user interface on the device that consists of LEDs and buttons. These buttons are used to send a signal to the central server and the LEDs are used to show responses from the server. Portability of the device is important and is accomplished by using a battery for power.

2. The central server.

The central server is a program that we developed that communicates with the different devices and the Redpin server. The device communication is initiated by button presses on the physical device or by the central server requesting an update on the position of the device. From there, the central server is able to display a map of all the different devices with the location generated by the Redpin server. This map also has an indicator to show when a button was pressed on a device.

3. Software Documentation

We provided a reference manual documenting the Central Server's code that contains the different APIs used by the device. This reference manual allows our sponsor to create items that work with the Central Server without having to review the device's code. Also, we thoroughly documented our code so that it can easily be understood by our sponsor.

4. Project Report

This final project report was written and delivered at the end of ECE 4820 – Senior Design II.

5. Presentation

A presentation concluding our senior design project was given at the end of the Spring 2014 semester. This presentation provided a concise analysis of our design process, findings, and demonstrated our working project.

Conclusions

The main focus of this project was position tracking using WiFi. A device was built that could be tracked within a building using only the WiFi signal strengths produced by wireless routers and access points. The device will eventually be used as part of a safety system at a company where employees wear it as part of their identification badge. If there is ever an emergency then they will press the emergency button on the device to alert the Central Server and the employee monitoring the Central Server.

There were multiple issues encountered with the original microcontroller, the MSP 430, and the original WiFi adapter, the CC 3000, which forced the design to change more than halfway through the semester. The newly picked microcontroller was the Arduino Uno with the Arduino WiFi Shield for the WiFi adapter. This new microcontroller performed much better and was easier to understand due to the better documentation and example programs.

There were also other problems with the tests performed at the Parkview engineering building because of interference from the building's WiFi network. The Arduino WiFi Shield can only scan and return 10 wireless networks at once and can't return the unique BSSID of each access point. This forced the testing to be done with controlled access points that could be configured with a unique SSID. This still did not work correctly so the testing had to be moved to a residential home which didn't have as many wireless access points as Parkview.

In conclusion, a device was built that could be tracked inside a building and be accurate to a specific room. A Central Server was written that could receive communication from the devices and send it to the Redpin Server as well as display the locations of each device. Finally,

the device was designed to be portable and used a 9 volt battery so that the user could freely walk around the building.

References

[1] Wei, T., & Bell, S. (2012). Impact of indoor location information reliability on users. Lecture Notes in Computer Science, 7478(2012), 258-269. doi: 10.1007/978-3-642-33024-7_19

[2] Redpin - indoor positioning for the rest of us. (2008). Retrieved from http://redpin.org/

[3] Thrun, S., Lookingbill, A., & Kadous, M. (2011, May 23). United states patent: 8,583,400. Retrieved from http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=/netahtml/PTO/search-adv.htm&r=22&f=G&l=50&d=PTXT&p=1&p=1&S1=(indoor AND tracking)&OS=indoor AND tracking&RS=(indoor AND tracking)

[4] Sundararajan, A., & Lin, J. (2011, July 15). United states patent: 8,589,318. Retrieved from http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=/netahtml/PTO/search-adv.htm&r=2&f=G&l=50&d=PTXT&p=1&p=1&S1=(indoor AND localization)&OS=indoor AND localization&RS=(indoor AND localization)

[5] Bickerstaff, R. (2011, February 25). United states patent: 8,589,230. Retrieved from http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=/netahtml/PTO/search-adv.htm&r=2&f=G&l=50&d=PTXT&p=1&p=1&S1=(indoor AND localization)&OS=indoor AND localization&RS=(indoor AND localization)

Appendix A

Device	Power Consumption (mA)	Duty (8 hrs)	Total (mA)
Atmega328	5.2	100%	5.2
AT32UC3A1256	20	100%	20
HDG104 WLAN (Tx)	178	3.33%	5.93
HDG104 WLAN (Sleep)	0.06	96.67%	0.06
		average current draw	31.19133333

# of transmits per minute	2
Duration of transmit (sec)	1

Low Power

Device	Power Consumption (mA)	Duty (8 hrs)	Total (mA)
Atmega328 (Idle)	1.2	90.00%	1.2
Atmega328 (Active)	5.2	10.00%	0.52
AT32UC3A1256(Idle)	14	93.33%	14
AT32UC3A1256(Active)	20	6.67%	1.334
HDG104 WLAN (Sleep)	0.06	96.67%	0.06
HDG104 WLAN (Tx)	178	3.33%	5.93
		average current draw	23.04533333

# of transmits per minute	2
Duration of transmit (sec)	1

Figure A1. Low Power Calculations

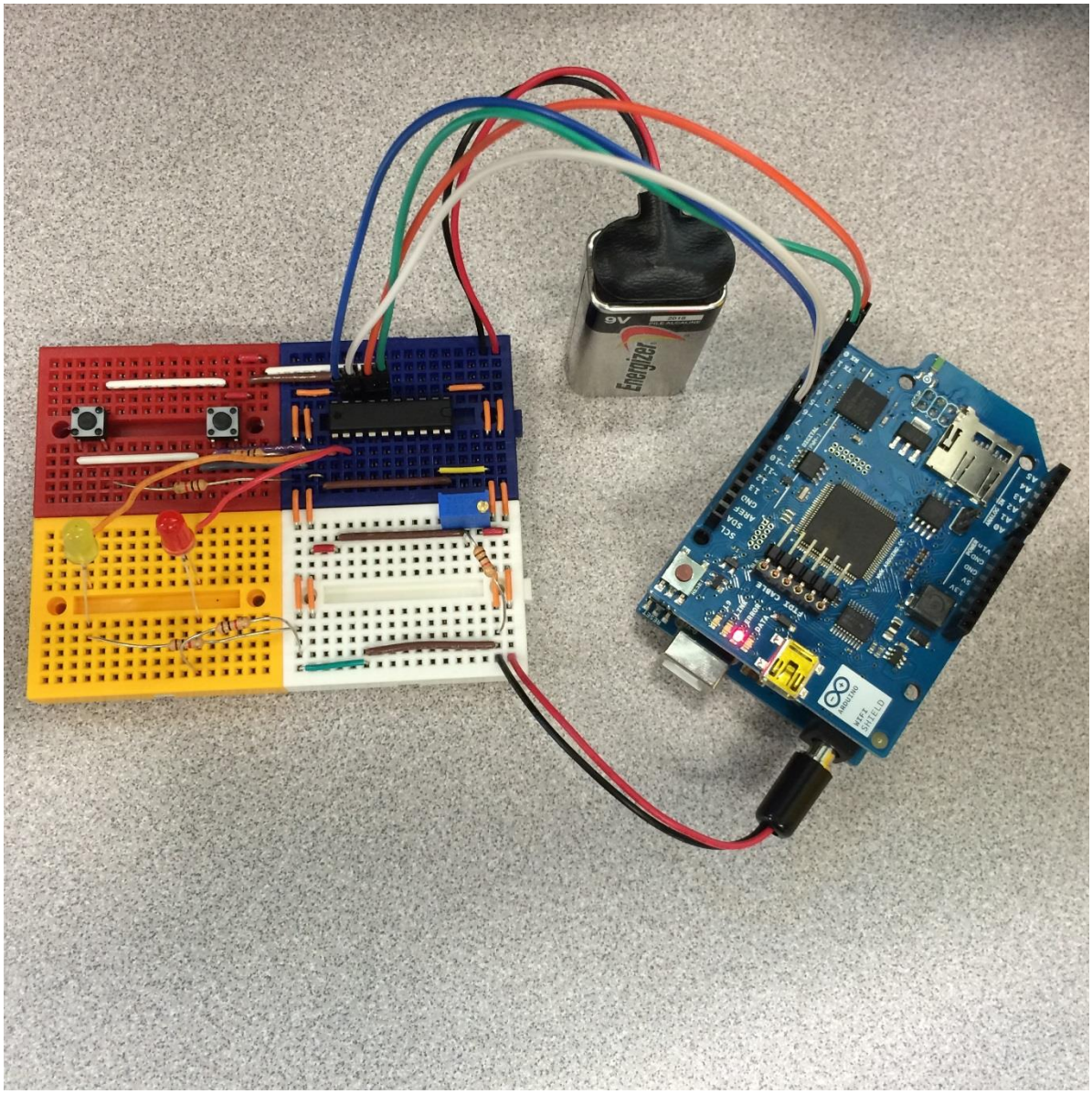


Figure A2. The device

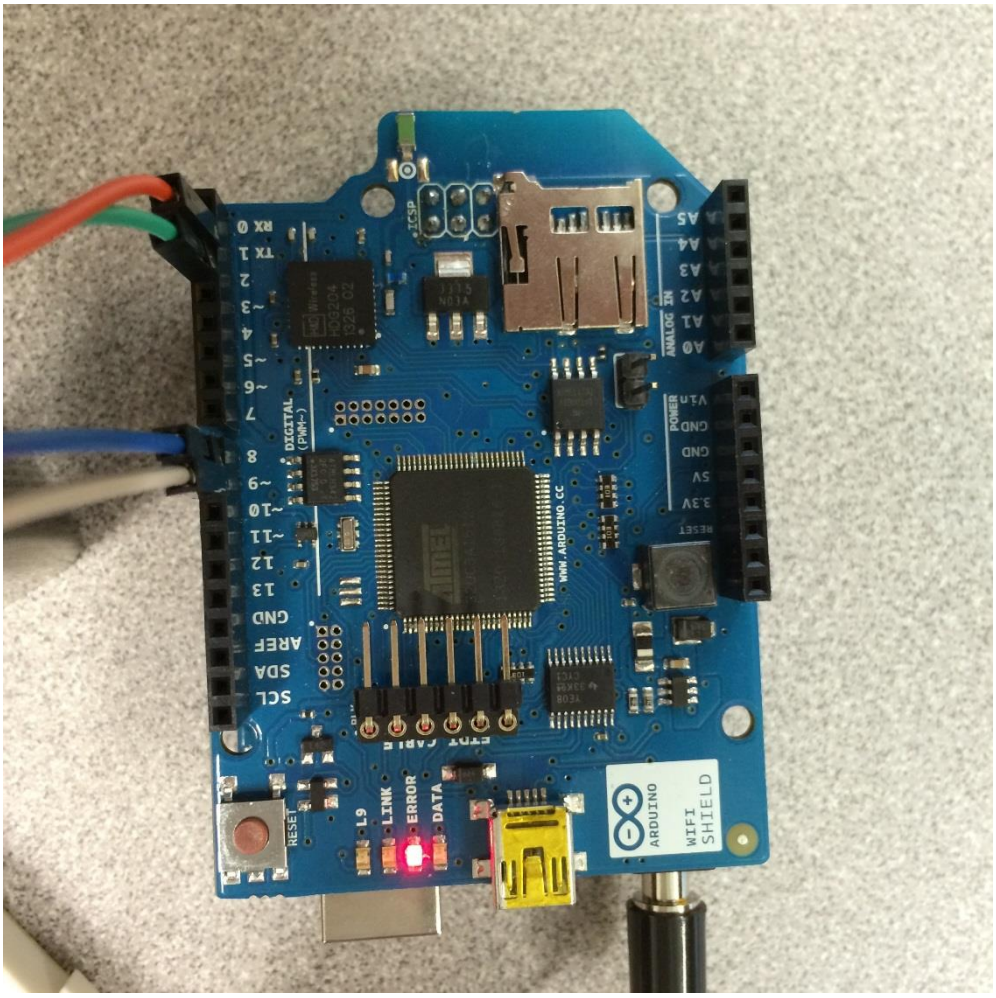


Figure A3. The Arduino Uno and Arduino WiFi Shield

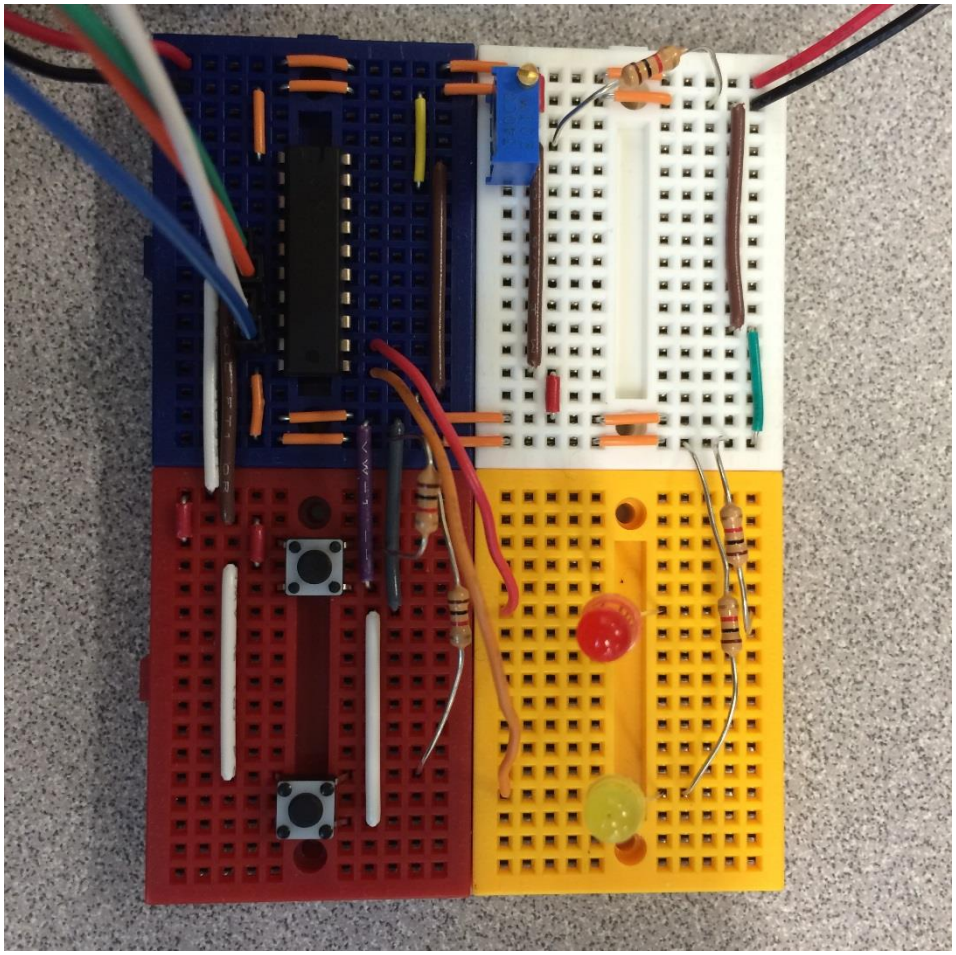


Figure A4. The user interface of the device