Emergency Personal Indoor Location System

Greg Buitkus, CSE, Brett Gavin, CSE, Michael Anderson, EE, and Paul Fallacara, EE

Abstract— **We introduce a system that when complete will be able to track an individual on one known floor of a building and will be able to display the path the individual took on a GUI that will be accessed outside of the building. The system is being developed for the purpose of helping to track and locate emergency personnel and should provide ideas and techniques for other engineers who are working on a system with the same purpose.**

I. Introduction

Indoor localization is still an unsolved, underdeveloped technology today. Modern day GPS localization doesn't function indoors so individuals have to offer directions to where they are in a structure. The E-PiL is going to accurately locate the individual, whom is wearing the device, within a structure, where the floor plans are known. Indoor localization is a major problem especially with emergency personnel. Since emergency situations are sporadic and unpredictable, many injuries and deaths occur because emergency personnel cannot locate the injured and lost in time [1]-[3].

There have been many solutions, though not as efficient as needed, in the past that aided in locating personnel indoors. Walkie-talkies and radios keep personnel in constant contact as long as they are connected to the same radio frequency. Another device is the PASS (Personal Alert Safety System), worn primarily by firefighters, emits a loud, piercing noise when the firefighter is immobilized [4]. As time passed the problem has been reoccurring. Many attempts at solving this issue have been conducted with great advances, such as the PASS (stated above), GLANSER, and WASP, yet no reliable, concrete solution has been created [4]-[7]. This problem makes emergency situations harder for the original team and the rescue group due to the lack of location information. Also "stealth" missions by police or military are difficult to provide backup personnel for, especially if teams get separated by great distances and need to communicate in non-verbal communication. Along with group problems, this issue is very hazardous for individuals. When the rescuers don't know the location of the lost or wounded, there is a greater lethality for both the rescuers and the incapacitated individuals.

This paper proposes a solution that is going to be a nonrestrictive, non-inhibiting device. In order to use this solution, the device is worn on the equipment of the individual. The one portion of the system that will inhibit the team, is that beacons need to be placed throughout the building, such that members of the team will always be in line of sight of at least 3 beacons. In order to reduce the restrictiveness to the team, the system will include an app that will calculate the best locations for the beacons to be placed within the building. This way a member, or members, of the team will know before entering the building where the beacons need to be placed, which will hopefully allow the system to be much less restrictive. The overall specifications of the system are listed below in Table I.

TABLE I
System Specifications

Specification	Value
Weight	<10lbs
Real-Time Output	<30s
Application OS	Android
Battery Life	>2 hours
Location Accuracy	<3m
Cost	<$200
Location Data Availability	Always
System Output	Path user took

II. Design

A. Overview

Indoor localization has many proposed solutions that are being researched and tested. The limitations of GPS in indoor environments causes indoor localization to be a very challenging problem to solve. Every year Microsoft holds the IPSN conference where different groups compete and show off their research on the issue [8]. One of the most common solutions is the use of radio frequency to do trilateration. When radio frequencies propagate through walls, the signal has a non-homogenous drop in signal strength due to the different materials contained in the wall. This fact makes calculating range accurately using strength of signal nearly impossible. This means that these calculations can only be done when the antennas of the devices are both within line-of-sight of each other. This fact leads to the use of Bluetooth low energy beacons that could be dropped by the user. This ensures that the radio frequency calculations used to find the position of the user is indeed line-of-sight RSSI (Received Strength of Signal Indication) calculations that can fairly accurately represent ranging data from these beacons. Using the ranging data from the Bluetooth beacons the system is able use a trilateration algorithm to determine the users position relative to the beacons.

Dead reckoning is a positioning algorithm that has existed for ages. Using a person's last known position along with data about speed and orientation, the person's position can be calculated after a certain amount of time has passed. Using a gyroscope and an accelerometer the orientation of the user can be known. In order to gain velocity data to go along the orientation data, n ultrasonic range finder will be used. With a combination of the three sensors a rough estimate of the users

G. Buitkus from Reading, Ma (e-mail: gbuitkus@umass.edu). B. Gavin from Westford, Ma (e-mail: bmgav0@umass.edu).
M. Anderson from Dracut, Ma (e-mail: mjanders@umass.edu).
P. Fallacara from Marshfield, Ma (e-mail: pfallaca@umass.edu).

position can be found by using the dead reckoning approach. The largest issue with the dead reckoning approach is that once the system has calculated a position that is not accurate the next positions are going to contain that error as well. These small errors can compound into great inaccuracy. Despite the drawbacks of the dead reckoning approach, it will still give a good estimate of where the user is, especially if it can be combined with a way to prevent the errors from compounding [9].

Many solutions rely on simply using RF to do some sort of localization. These solutions require the unit to set up outdoor RF transmitters. Building these transmitters and getting accurate location data from these is very difficult and very expensive. Bluetooth, although very limited in range, can give the same effect when placed indoors. This idea stems of a system called “GLANZER” [5], which drops RF devices periodically, meaning that at least one RF device will always be in line of site of the responders. This system is incredibly expensive, so much so that very few units have been sold based solely on their price. Using Bluetooth beacons and the dead reckoning hardware, a fairly accurate location can be calculated without a large overhead cost. The combination of the two approaches not only give a good location estimation but, it also allows for the user to never go into the dark, meaning that there is never a spot in the building where they can not be seen. This is very important, as losing the firefighter or policemen could be potentially lethal. Creating a combined system will give us the accuracy we want out of our system, without driving the price up.

The Emergency Personal Indoor Location System includes 4 fundamental subsystems powered by a 9W lithium-ion battery that are integrated to calculate the user’s indoor location. The system’s block diagram shown in Figure 1 shows the system design. An Arduino Uno reads the dead reckoning sensors and sends that data to an Android based phone or tablet. This Android device will calculate the RSSI of the beacons that are within sight and package that data along with the Arduino data. This data is then transmitted to a computer over a wireless network. Using a trilateration algorithm along with a localization algorithm the users position can be determined.

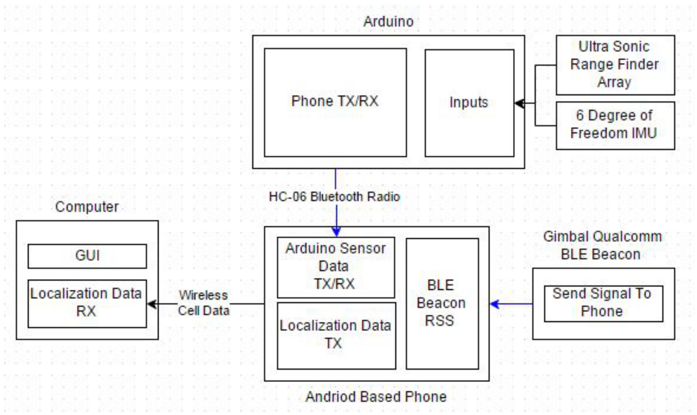


Fig. 1: System Block Diagram

An important part of a system like this, is that due to the nature of the situation when it is being used, it needs to work reliably. This is true because if the system fails, lives could possibly be in danger. Due to this fact, there is clearly an ethical responsibility of the system, as well as the team developing the system to provide a reliable product. The team should not be tempted to take any shortcuts when it comes to research or development of the system, especially when it comes to the reliability of the product. The team also needs to be honest about any potential situations it may find that could cause the system to fail. Thus far, the team believes it has be thorough in its research and design of a system that will be as reliable as possible.

B. Ultra Sonic Sensors and IMU

Block 1 in the system's block diagram consists of two of the three inputs to our system. The first of these inputs is a combination of two distances determined by two XL-MaxSonar High Performance Ultrasonic Rangefinders [10]. These two ultrasonic rangefinders will be placed on the individual to calculate the distance from the individual to the wall directly in from as well as the distance between the individual and a wall that is located to either side of them.

These ultrasonic sensors measure the time of flight for sound that has been transmitted by the sensors and reflected back from nearby objects. That time of flight is then used to output a range reading, which can be updated every 100 milliseconds. The ultrasonic range finders have three separate outputs that update the range data simultaneously: analog, pulse width, and serial. These outputs are then connected to the Arduino that contains a program to interpret the output of the rangefinders in a user friendly way [11].

The second of two inputs for this block is IMU (inertial measurement unit) data, which is a combination of an accelerometer and gyroscope. The SparkFun 6 Degrees of Freedom IMU Digital Combo Board [12][13] combines the accelerometer and gyroscope to account for the pitfalls of one another. The accelerometer can determine the orientation of an object, but lacks the ability to measure correctly during movement and rotation, which is where the gyroscope becomes critical [14]. The data and clock outputs of the IMU are connected to two analog pins of the Arduino, which contains a program to interpret the IMU data and display it in a user friendly way as well. The IMU data will be useful for two purposes. The accelerometer data should stay almost constant, with the user only experiencing a gravitational force of -1g in the z direction if the IMU is placed on an object exactly as shown.

If a drastic change in the orientation occurs and that change remains constant, the object has changed orientation. If an emergency personnel is wearing the device and their orientation has changed drastically, they may have fallen down and need immediate assistance.

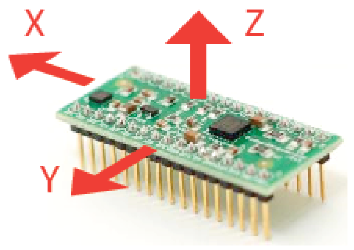


Fig. 2: IMU (Showing chip orientation)

The gyroscope data, measured in degrees per second, will be used for determining if an individual has turned left, right or around. When the x, y, or z component has drastically changed and short amount of time and returned to origin, we know the individual has changed direction and we can use this data, along with the rangefinders, to display the path an individual has taken.

The main technique used to construct this block involves programming and reading data sheets. Both the ultrasonic rangefinders and the IMU require extensive reading on both data sheets to learn all aspects of the devices. With this knowledge, a circuit can then be constructed with the Arduino, which contains codes for interpreting the sensor's data [15].

Along with learning all about the sensors via data sheets, the math used to interpret the IMU data was extensively researched and used to build the Arduino program [16].

The experiments required to test this block involve walking around with the IMU and rangefinders to see if they perform as we expect based on an individual walking around a room. Once we know exactly what to expect from the sensors from certain movements, we can program that information into our GUI so when those movements occur, the path can be drawn properly.

C. *Bluetooth Low Energy Beacons*

This block uses Bluetooth Low Energy (BLE) beacons for ranging/positioning data. Due to BLEs' low energy characteristics they are efficient in power consumption and usage. In addition, they are proven to be reliable in short-range systems [17][18], which is the best solution to the indoor location problem. As the individual navigates through a building, they will drop these beacons in line of sight in corners of every room they enter, every doorway they pass through, and roughly every 5 meters in wide-open spaces. Then the signal strength of these beacons will be received by the android app and sent to the computer for localization calculation. The BLEs the E-PiL uses are Gimbal Series 10 Proximity Beacons [19]. These beacons are small (28mmx40mmx5.6mm), lightweight, and cheap to replace ($5 per beacon). These beacons also have an effective range of 0-50 meters. They use a CR2032 battery, which are easily replaceable and have a battery life of many months or up to a year.

To test the BLEs, we first measured signal strength along a 0.2-meter radius around the phone and found that orientation of the phone contributes to the data (Figure 3).

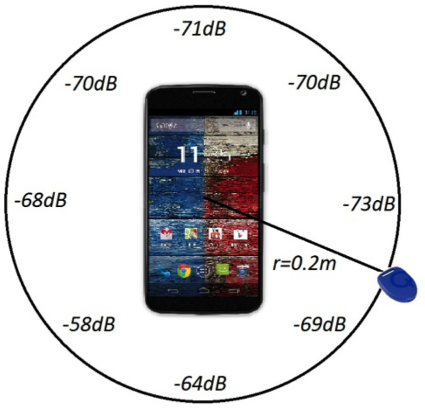


Fig. 3. Signal Strength with respect to phone circumference. Given a 0.2-meter radius, the strongest signal occurs in the bottom left of the phone. This is most likely due to the location of the Bluetooth chip inside the phone.

We then took measurements outside. During this test, the conclusion of more accurate data when the back of the phone was facing the BLEs was proven (Figure 4). This is because the Bluetooth adapter faces the back of the phone to keep the RF waves away from the phones user's head.

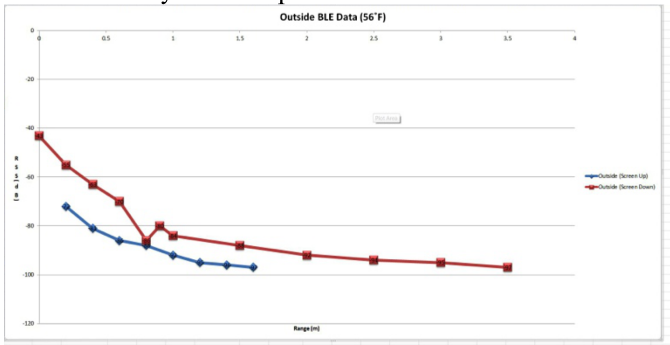


Fig. 4. Plot of the measurements taken outside. The top line represents when the screen was facing away from the BLEs and went out of range at 3.5 meters. The bottom line represents when the screen was facing toward the BLEs and went out of range around 1.5 meters.

Testing inside resulted in a greater range of data than the outside measurements (Figure 5).

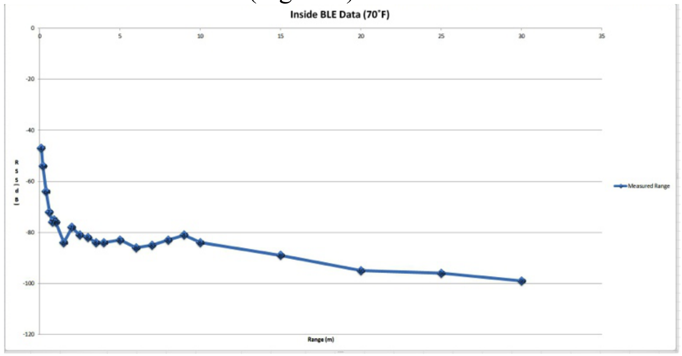


Fig. 5. Plot of the measurements taken inside. The range measured to was 30 meters before the device went out of range.

Along with these inside data points, measurements were also taken when the BLEs were not in line of sight (around a corner). As expected, the signal strength decreases as the beacon moves farther out of line of sight (Figure 6).

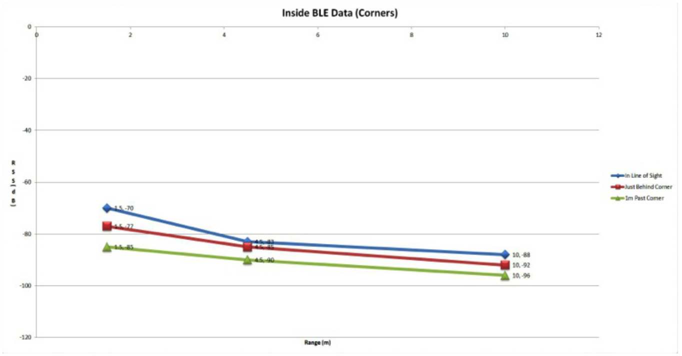


Fig. 6. Plot of the measurements taken inside around corners. The top line represents when the BLEs were in line of sight. The middle line represents when the BLEs were just around the corner. The bottom line represents when the BLEs were 1 meter past the corner. As shown, the signal strength decreases as the BLEs move farther out of sight.

In order to use the device to locate personnel, the system needs to use the BLEs signal strength to calculate the individual's range from the BLE. To do this, use a signal strength (RSS) equation [proposed by Prof. Janaswamy]:

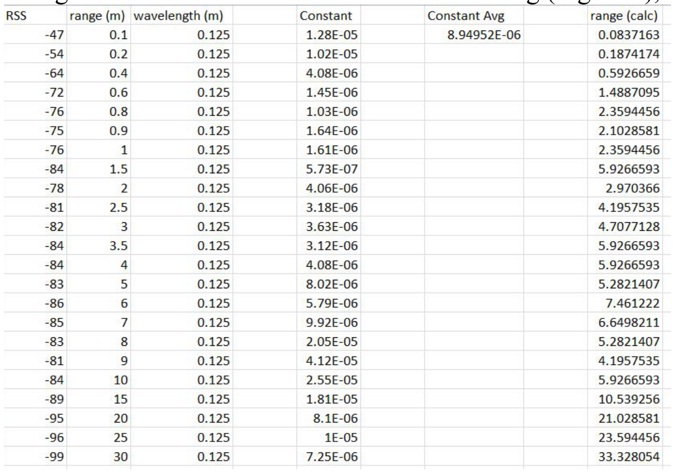
$$RSS = 10\log\left(\frac{G_t G_r}{4\pi^2}\right) + 20\log\left(\frac{\lambda}{range}\right)$$

(1) where G_t is a bias in the Bluetooth beacon towards a particular direction, G_r is a bias in the phones Bluetooth reciever towards a particular direction, λ is transmission wavelength, and range is the distance the phone is away from the given beacon

To solve for the signal constant $\frac{G_t G_r}{4\pi^2}$:

$$\frac{G_t G_r}{4\pi^2} = e^{\frac{RSS - 20\log\left(\frac{\lambda}{range}\right)}{10}}$$

Using Excel and the measured values from testing (Figure 7),



RSS	range (m)	wavelength (m)	Constant	Constant Avg	range (calc)
-47	0.1	0.125	1.28E-05	8.94952E-06	0.0837163
-54	0.2	0.125	1.02E-05		0.1874174
-64	0.4	0.125	4.08E-06		0.5926659
-72	0.6	0.125	1.45E-06		1.4887095
-76	0.8	0.125	1.03E-06		2.3594456
-75	0.9	0.125	1.64E-06		2.1028581
-76	1	0.125	1.61E-06		2.3594456
-84	1.5	0.125	5.73E-07		5.9266593
-78	2	0.125	4.06E-06		2.970366
-81	2.5	0.125	3.18E-06		4.1957535
-82	3	0.125	3.63E-06		4.7077128
-84	3.5	0.125	3.12E-06		5.9266593
-84	4	0.125	4.08E-06		5.9266593
-83	5	0.125	8.02E-06		5.2821407
-86	6	0.125	5.79E-06		7.461222
-85	7	0.125	9.92E-06		6.6498211
-83	8	0.125	2.05E-05		5.2821407
-81	9	0.125	4.12E-05		4.1957535
-84	10	0.125	2.55E-05		5.9266593
-89	15	0.125	1.81E-05		10.539256
-95	20	0.125	8.1E-06		21.028581
-96	25	0.125	1E-05		23.594456
-99	30	0.125	7.25E-06		33.328054

Fig. 7. This is the range and signal constant data for certain RSS.

We get an average signal constant of $8.95\text{x}10^{-6}$. Now using this constant, solve equation (1) for the range:

$$range = \frac{\lambda}{e^{\frac{RSS - 10\log\left(\frac{G_t G_r}{4\pi^2}\right)}{20}}} \qquad (3)$$

Using Excel and equation (3), the range values corresponding to certain RSS can now be calculated (Figure 8).

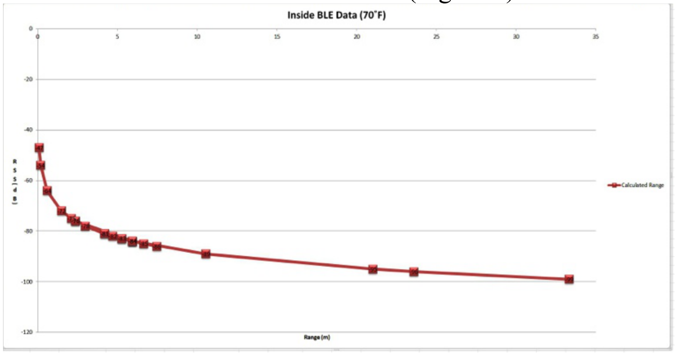


Fig. 8. Plot of the calculated ranges. The calculated range before the device went out of range was 33 meters.

Comparing these plots, we can see how precise the correlation is (Figure 9).

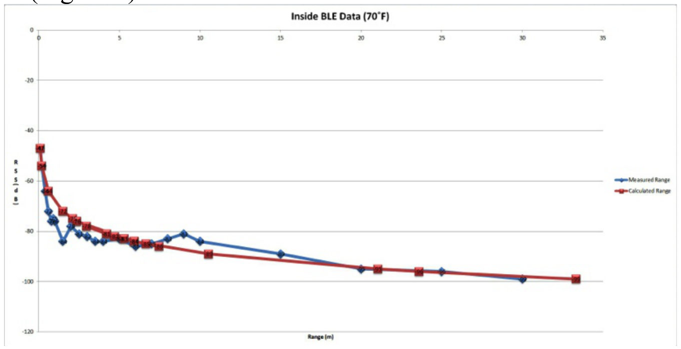


Fig. 9. This is an overlapping plot of the measured range plot and the calculated range plot. As shown, a correlation between the two sets of data can be seen.

The knowledge of Bluetooth and radio frequencies used in this block was gained in Fields and Waves. RF propagation and strength of signal calculations were taught to the group in an advising meeting with Professor Janaswamy.

D. Android Phone Application

Sending data back to the users outside from the Arduino poses a big connectivity issue. Antennas that would be used to transmit this data would be bulky so using the already present cellular data system is an easy solution to this data problem. Using Android OS, data can be sent to the computer running the user-tracking interface. This creates a link between the Arduino sensors and the computer that will be running the localization algorithm.

Along with getting the data out to the tracking user interface so the users location can be determined and displayed, the Android application must also collect the data from the Arduino. Once the data is collected, it is sent over a Bluetooth connection between the Android phone's Bluetooth adapter and an HC-06 Bluetooth serial interface which is hooked up to the Arduino. This connection is set up where the Android is the master sending commands to the Arduino, the slave, which responds to these commands with the data from all the sensors [20]. Once the application has collected the data from the Arduino the Arduino can resume collecting data until the HC-06 is ready for more data to be pushed into its buffer. This ensures that there is constant data from the IMU and the range finders being fed to the localization algorithm.

The communication between the phone and computer also involves sending the data from the Bluetooth beacons. This data is in the form of the RSSI data of each beacon, which will be used to do trilateration. This data is also collected by the android application. Using the API from Gimbal, the RSSI of each of the beacons is determined in the application [21]. With the RSSI from the beacons all the information needed to run the localization algorithm can be sent to the computer.

The communication pipeline loosing connectivity is a very big problem because the data would no longer be able to be sent, which would leave the user in the dark. The communication between the Arduino and the phone will always be reliable because the strength of signal for both Bluetooth receivers will always be high enough. This is because both devices will be on the user, meaning that the two devices can't possibly be far enough apart that they couldn't communicate with each other. The other connection in the data pipeline, the connection between the phone and the tracking user interface, uses wireless data. It is assumed that the buildings that the users are entering are in areas with cellular signal. Using the Bluetooth adapter of the phone along with the wireless data connection the data will always be sent to the computer to do the localization algorithm. This will be tested in smoky environments, high temperature environments, and situations where the system is moving to ensure the connection does not fail. With the communication system in place Block 1 and 2 are now ready to be used in localization algorithm of block 4.

The Android program is written in Java using the Android 20 API (Kit Kat 4.4) [22]. In order to write the Java code, techniques were used from both CompSci121 as well as experience gained during a summer internship. Along with the Android programing knowledge the knowledge of serial communication for the app was retained from Computer Systems Lab 1 and 2. The Arduino serial communication portion of the app was written in Arduino based C and was learned through IEEE's Micromouse competition.

E. Output GUI

Simply put, the output GUI is the way that the system displays the location of the user. In order to accomplish this, the GUI is meant to run on a laptop that is outside of the emergency situation. The idea is that a team lead would be able to monitor his team members' movements while they participate in an emergency situation, from a safe location.

Once E-PiL is finished, the GUI will take in location information from the phone app, and then display the results on screen. In order to test that the GUI is working correctly before the entire system is completed, a set of simulated data was created. The way that this works is that the program reads in a set of numbers from a text file, which correspond to information about the floor plan of a simulated building. Once the system has finished reading in the floor plan, the system will then try to contact a server to gather the location data. This server is simply another java program that is reading in simulated location data from a text file, and sending the data to the GUI through a TCP socket. In this case the GUI is considered to be the master, while the server is the slave. This means that the server isn't going to send any data until the GUI asks for it. The way that the GUI is set up is that, once it has finished reading in the floor plan, it will try to open a connection to the server. Once the connection is established, it will send a packet requesting location information from the server. When the server receives this request it will respond with the current location of the user. When the GUI receives this location information, it will process and display it, and then request new location information from the server. This cycle will then continue either until the user ends the program, or the server sends a message saying that it is no longer receiving information from the sensors.

The reason why this idea of a server was added into the GUI was for testing purposes; the connection with the server closely resembles what a connection with the phone app would be. Therefore, setting up a connection between the GUI and a simulated server proves that a connection with the app is possible once the app is ready to send data to the GUI.

For the purposes of MDR, the goal was simply to show that the program could display a set of location data points in an easy to follow path. This was achieved by simply displaying the data points gotten from the server on top on the floor plan of the simulated building. The result of this was that a clear path was shown of how the simulated user was moving through the simulated building. An example of how the user's path is display can be seen below.

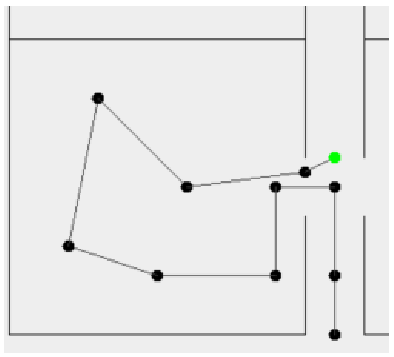


Figure 10: Example of a generated user path

As seen in the above diagram, the path of the user is displayed clearly. It is clear to the team lead, or whoever is monitoring the situation, that the green dot is the user's current location. It is also clear that the black dots represent the users previous locations, thus showing the path that the user took to get to their current location. This way the team lead will be able to follow the users path to get to where they are, should the need to do so arise.

The GUI program, as well as the simulated server program are both written in Java. In order to write the Java code, techniques were used from both CompSci121 as well as ECE242. On top of the basic Java syntax and data structures used, the creation of the socket between the GUI and the server also applied knowledge of TCP/IP learned in ECE374. The only part of this program that wasn't learned in classes, and as such needed to be learned elsewhere, was how to actually display the GUI using Java.

III. Project Management

Table II
MDR Deliverables

Input	**Status**
Demonstrate working IMU	Completed
Functioning ranging data from ultrasonics	Completed, with some bugs
Functioning ranging data from Bluetooth beacons	Completed
Data Delivery	
Bare bones app to collect and send data from Bluetooth	Completed, with some bugs
Output	
Create a GUI to display location data	Completed
GUI should be able to get dummy data from a server	Completed

Above in Table II the MDR deliverables are shown. At MDR the IMU and Range finders were displaying correct orientation and accelerations. The range finders distance calculations were not correct because of a power issue that scaled the distances incorrectly for the code written to collect the data. These units still need to be used to create data that can be used to determine the position of the user, using the dead reckoning approach. The Arduino connection to the phone was working correctly but floating-point numbers could not be sent. Making this connection more robust is needed to ensure the system is working correctly. The Bluetooth beacons were giving correct RSSI data that could be used to calculate the distance the beacon was from the phone. Using this we can use a trilateration algorithm to determine the users location in the building. The final deliverable for MDR, the user interface showing the users path with dummy data, was also completed. This simulated location data was sent from a server running on a local computer to the GUI program when the GUI requested the data. This was done to show that the connection to the phone would be able to communicate the data needed to display the path. The GUI now needs to run the localization algorithm on the incoming data to display the path.

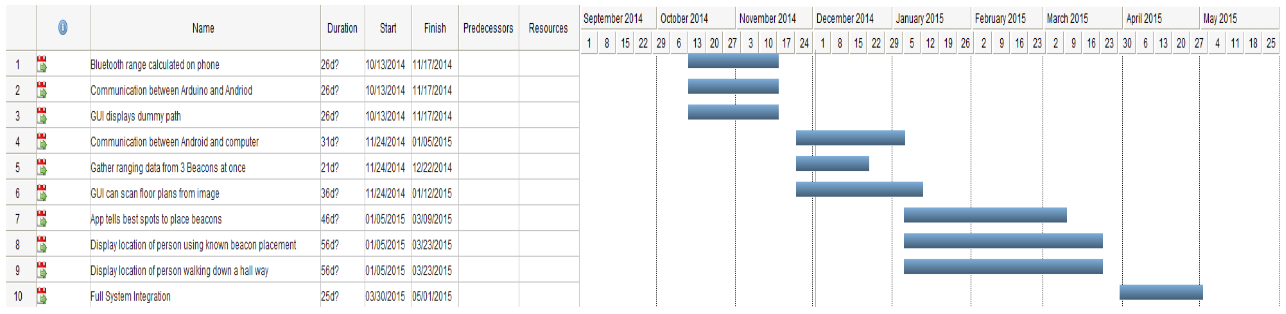
To get to the present status of project the group has worked closely together. Brett worked alone on the GUI and will continue to work on the high level programing of the project, including the improvement of the GUI along with the localization algorithm. Greg works on the low level embedded programming of the Android and Arduino subsystems, but he will also be helping Brett on the localization algorithm in the future. Mike specializes in the RF triangulation using Bluetooth beacons along with the power system. Paul worked on getting the IMU and range finders to determine the movement of the user. Each group member worked on their own specialized parts but also help the other group members whenever possible. As a group we meet every week to discuss our goals for the week, what has been accomplished, and when everyone will be working so we can easily work together. Along with these meetings we are in constant contact because of our class schedules and a group text thread. Overall the group dynamic has worked to keep our project moving along.

IV. Conclusion

Overall, all four subsystems are working for the most part. As was stated in the project management section, there are still some bugs, specifically with the IMU and ultra sonic sensors, however the group is generally ready to move forward with starting to put everything together into one system.

Moving forward, the main goal will be in developing and implementing an algorithm to keep track of where the user is at all times using all the collected data. In order to accomplish this, the group plans to be able to track a user in an average sized room with known pre-placed Bluetooth beacons. In this scenario it is very important that the user is constantly in sight of at least three beacons. This way it will be possible to use trilateration to locate the user. Doing this the group will be able to prove that the beacons are working correctly and can in fact locate the users.

While the team is working on tracking a users movement in a room with known Bluetooth beacons, the team will also be



	Name	Duration	Start	Finish	Predecessors	Resources
1	Bluetooth range calculated on phone	26d?	10/13/2014	11/17/2014		
2	Communication between Arduino and Andriod	26d?	10/13/2014	11/17/2014		
3	GUI displays dummy path	26d?	10/13/2014	11/17/2014		
4	Communication between Android and computer	31d?	11/24/2014	01/05/2015		
5	Gather ranging data from 3 Beacons at once	21d?	11/24/2014	12/22/2014		
6	GUI can scan floor plans from image	36d?	11/24/2014	01/12/2015		
7	App tells best spots to place beacons	46d?	01/05/2015	03/09/2015		
8	Display location of person using known beacon placement	56d?	01/05/2015	03/23/2015		
9	Display location of person walking down a hall way	56d?	01/05/2015	03/23/2015		
10	Full System Integration	25d?	03/30/2015	05/01/2015		

Figure 11: Gantt Chart

working on developing an app that will indicate to the user where they should be putting beacons. The thought behind this is that, once it is shown that the system can track a user with known Bluetooth beacons, developing an app that tells a user where to put beacons is a good way for the system to known exactly where the beacons will be. The good thing about this app is that it can help to ensure that the user will always be in sight of three beacons, allowing the system to use trilateration.

On the hardware side of the project, the main goal, apart from gathering more accurate data, is to combine all of the sensors into a system that is wearable for a user. The two main parts to this include creating a printed circuit board for the components, as well as developing a power system that will allow the system to run on its own.

Another goal of the system will be to allow the GUI to scan in a floor plan from an image file. This is because it is much more likely that an image will be available than a text file describing the floor plan. Scanning in an image will also eliminate the need for a text file to be in the correct format, which will make the system more reliable.

As stated in the project management section, Brett will be leading the efforts in developing and implementing the location algorithm. He will also be the one adding the image scanning functionality into the GUI program. Greg will focus much of his time on developing the beacon placement app, as well as added functionality for the phone app to send location data to the GUI. Greg will also be assisting Brett in developing the location algorithm. Mike will be the one who develops the power system, and along with Paul will help to put all of the sensors together into one system.

While every portion of the system is bound to encounter problems along the way, the group anticipates the biggest and hardest problems to come from the location algorithm. The task of combining the Bluetooth data with the ultrasonic and IMU data in a way that will be displayable to the user is a daunting one for sure. This task gets even more intimidating when it has to be incorporated with the beacon placement app. The beacon placement app causes problems because the team needs to figure out what to do in the scenario that the user doesn't place the beacons where they are supposed to. This causes all kinds of problems, because the Bluetooth beacons will be much less useful at correcting the dead reckoning errors from using only the ultrasonic and IMU data.

In summary, there is much work to be done, however the group is confident that they have developed a plan that will help them to achieve their goal of producing a system that will track emergency personnel effectively.

Acknowledgment

The authors would like to thank professor Ramakrishna Janaswamy for advising the project and assisting in brainstorming various ideas used in the project.

References

[1] D. Munson. (2012). *New Technologies in the Fire Service Series: Firefighter Tracking Systems* [Online]. Available: http://www.northnettraining.com/detail.aspx?cid=38

[2] D. Chauhan. (2014, May 20). *Holy Grail of firefighter tracking on the horizon* [Online]. Available: http://www.firerescue1.com/fire-products/communications/articles/1916468-Holy-Grail-of-firefighter-tracking-on-the-horizon/

[3] M. Luttrell, H. Bray. (2010, August 4). *New "Missing Firefighter" Location Technology Unveiled At Worcester Polytechnic Institute Conference* [Online]. Available:

http://www.firefighterclosecalls.com/news/fullstory/newsid/113460

[4] M. van der Feyst. (2014, September 2). *Firefighter PASS device: Like a 911 call* [Online]. Available: http://www.firerescue1.com/fire-attack/articles/1977281-Firefighter-PASS-device-Like-a-911- call/

[5] Gimbal, "Proximity Beacons," Series 10 datasheet, [Revised Nov. 2014].

[6] J. Mapar. (2011, Aug. 1). *GLANSER: A Scalable Emergency Responder Locator System* [Online]. Available: http://www.wpi.edu/Images/CMS/ECE/GLANSER_-_WPI_PPL_2011_-_AmitKulkarni-Aug1%281%29.pdf

[7] M. Mordecai. (2012). *WASP: Wearable Advanced Sensor Platform* [Online]. Available: http://www.globeturnoutgear.com/innovations/wasp

[8] I. Sharp, K. Yu, Sensor-based dead-reckoning for indoor positioning. Physical Communications (2013). http://dx.doi.org/10.1016/j.phycom.2013.11.013

[9] I. Sharp, K. Yu, Sensor-based dead-reckoning for indoor positioning. Physical Communications (2013). http://dx.doi.org/10.1016/j.phycom.2013.11.013

[10] "XL MaxSonar EZ Datasheet". Retrieved November , 2014 Available: http://www.maxbotix.com/documents/XL-MaxSonar-EZ_Datasheet.pdf

[11] Allen, B, "Distance Detection with MaxSonar ultrasonic rangefinder". Retrieved November, 2014 Available: http://playground.arduino.cc/Main/MaxSonar

[12] "ITG-3200 Product Specification Revision 1.4". Retrieved November , 2014 Available: https://www.sparkfun.com/datasheets/Sensors/Gyro/PS-ITG-3200-00-01.4.pdf

[13] "Digital Accelerometer". Retrieved November , 2014 Available: https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf

[14] Meyer, A, "Stable Orientation – Digital IMU 6DOF + Arduino". Retrieved November, 2014 Available: http://bildr.org/2012/03/stable-orientation-digital-imu-6dof-arduino/

[15] Versano, F, "My first 6 DOF IMU Sensors Fusion Implementation: ADXL345, ITG3200, Arduino and Processing". Retrieved November, 2014 Available: http://www.varesano.net/blog/fabio/my-first-6-dof-imu-sensors-fusion-implementation-adxl345-itg3200-arduino-and-processing

[16] Starlino, "A Guide to using IMU in Embedded Applications". Retrieved November, 2014 Available: http://www.starlino.com/imu_guide.html

[17] Mikhaylov, K.; Plevritakis, N.; Tervonen, J. Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI. *J. Sensors Actuator Networks,* vol. 2, no. 3, pp. 589- 613, Aug., 2013.

[18] Siekkinen, M.; *et al*. "How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4," in *Wireless Communications and Networking Conf. Workshops*, Paris, 2012, pp. 232-237

[19] Gimbal, "Proximity Beacons," Series 10 datasheet, [Revised Nov. 2014].

[20] HC -06 Serial Bluetooth Products "Bluetooth serial interface module and Bluetooth adapter" HC-06 datasheet

[21] Gimbal, "Proximity Overview". [Online]. Available: https://gimbal.com/doc/proximity_overview.html [Accessed: Dec. 2 2014].

[22] Android, "Android 4.4 API's". [Online]. Available: https://gimbal.com/doc/proximity_overview.html [Accessed: Dec. 2 2014].