Indoor People Tracking System (IPTS)

Daniel DeFazio, Brandon Tuero, Matthew Rhodes

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* -- GPS tracking systems have become very common place in the world today. They have been placed into cars, cellphones, and even gaming systems; however, one large weakness of these types of systems is when a user enters a building. The entire GPS system depends on a very stable signal to determine the user's current location. When one enters a building, the walls, floors etc., make the task of determining location very difficult. In our project we utilize a grid of radios placed throughout the building to determine the location of a tag. This approach gets around many of the limitations that GPS brings to the table while still maintaining a good degree of accuracy.**

***Index* -- Radiofrequency Identification, RFID tags, Radar tracking, Radiofrequency integrated circuits, embedded software.**

I. Introduction

This project was conceived and developed to incorporate low-cost, off-the-shelf, S-band radio transceivers with smart detection software to implement a practical indoor position tracking system. This complete, high level electrical and computer engineering design project utilizes the ultra high frequency radio's mesh networking capabilities to actively track the relative position of designated tag modules. This system is capable of tracking the approximate location of guest and/or personal within a facility containing classified/secret room locations based on relative signal strength. The system will automatically notify authorized personal when a security breach event takes place. This happens by predefining specific room locations as "secure", and distinguishing tag modules as authorized "security tags" and unauthorized "guest tags". A graphical user interface, developed in C#, is used to display the position of both security and guest tags against a custom map representing the layout of the area under observation. The point of observation requires a computer system with a serial connection to a bridge node. The bridge node will send and receive serial data to the network, connecting the network to the observation point. The guest tags are capable of identifying their location within the facility and sending that information out to an LCD screen for the user to observe. The guest tags are also capable of alerting authorized personal to an emergency event by forwarding its location to the security tag and setting-off a short burst through a simple piezo buzzer. The locations of the security breach and emergency distress call will be viewable on an LCD display on each security tag module as well as the observation terminal. For added convenience, the system is capable of detecting a low battery output to the portable tags. This is indicated by turning on a red LED located on the tag modules and setting a low battery indicator on the GUI. This system will not track the specific location of people within a given room. The system will determine whether a room, hallway, or space is occupied, only. The relative location within that space is unknown.

II. System Overview

The system was intentionally designed to easily scale up or down to allow for integration into a building of any size. With a maximum line-of-sight communication rang of up to 3 miles, this prototype is capable of servicing a multi-building complex with a single observation terminal. Because the system operates on a mesh network, the bridge node connecting the observation terminal to the network is required to be within range of only one other reader node. This allows for a higher degree of flexibility for placement and installation into its environment. To demonstrate proof of concept and system functionality, the current prototype is capable of tracking location in only two rooms (of any reasonable size) and the adjoining hallway. This limitation is only a result of the limited number of reader nodes. The number of people that can be tracked in this case is two; the authorized person and the guest (or unauthorized person). Again, this is only a limitation of the number of tags. If a gap in surveillance can be tolerated, meaning only specific room occupation is of interest, the system can be set to report only when a person has entered any three rooms of a building or complex. To help demonstrate the intended layout of the system, Fig 1 shows a generic map of two rooms and the adjoining hallway. One reader node is placed in each room, ideally centrally located, and one reader is placed in the hallway.

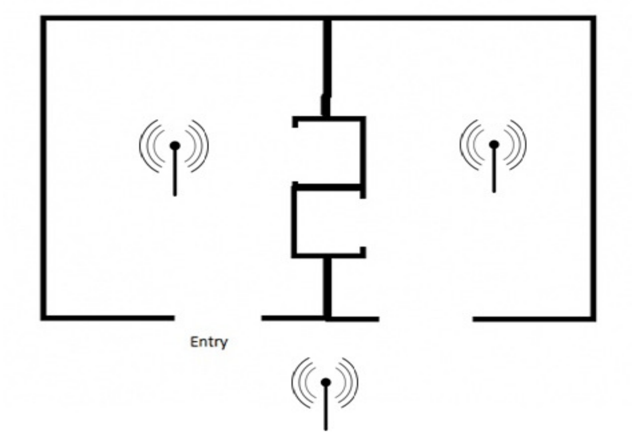


Fig 1. Generic room layout and node placement for intended operation.

Notice that the bridge node (also referred to as the base node) and the observation terminal that will run the GUI are not depicted in Fig 1. These devices are not required to be in the immediate vicinity of the area under observation and can be set-up in a remote location within building or complex.

In total, the current system utilizes 3 stationary reader nodes, 1 stationary base node, 2 portable tags, and a computer system to power the GUI. The two portable tags contain additional electronics to control the security breach detection and emergency distress call functions. These components will be discussed in detail in later sections.

III. System Components

The system is comprised of three major hardware components that are interfaced to create the final project. This section provides a semi-technical description of each of these components.

A. *Radio Transponders*

The Synapse RF Engine, models RF100PC6 and RF100PD6, are used for all wireless communication in this project. These radio transceivers are ideal for this project for several reasons. These devices are equipped with 2 UART ports with HW flow control which are required for serial communication to a secondary microcontroller. They have an outdoor LOS range of up to 3 miles with a data rate of 250 kbps. Low power consumption is critical for the portable, battery powered tags. Given 3.3 V supply, these devices have a current draw of only 115 mA for transmit and 1.6 μA in LP mode. Each RF engine combines a microcontroller, an 802.15.4 radio, an external power amplifier, and an antenna. The RF100PC6 model, used for all reader and bridge nodes, includes an integrated F antenna. The RF100PD6 model, used for all tag modules, includes an SMA connection for an external antenna. A standard 4" duck antenna with 2 dBi gain is used to increase the maximum range and signal stability. Figure 2 shows a block diagram of the major subsystems comprising the RF100.

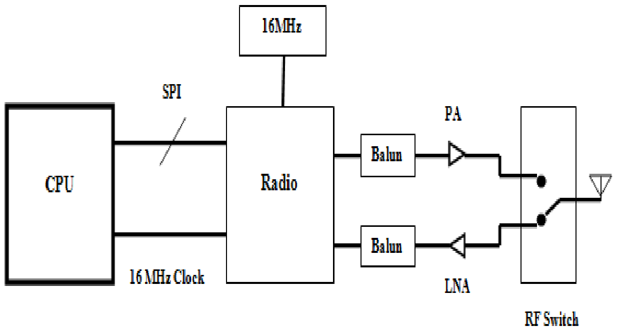


Fig 2. Block diagram showing the major subsystems comprising the RF100

Carrier modulation used by these devices is a direct spread (DSSS) with offset quadrature phase shift keying (OQPSK) at 2.4 GHz. All RF100 modules comply with Part 15 of the FCC rules and regulations. A maximum of 5 dBi gain can be achieved with an external antenna and still maintain compliance.

The radio transceivers are loaded with SNAP network operating system that is the protocol spoken by all wireless devices used in this project. They include an on-board Python interpreter that is used for application development. These devices are capable of over-the-air programming that allows convent customization for any environment.

B. *Microcontroller*

The microcontroller that is used for both the guest and security tags is the Texas Instruments MSP430G2231. This microcontroller has several key features that made it the perfect choice for this project. The microcontroller has a low supply voltage range of between 1.8 and 3.6 volts. Its power consumption is also very low with an active mode current draw of 220 μA and an off mode current draw of only 0.1 μA. The controller also has 5 different power-saving modes in which this project utilizes its lowest, denoted as low power mode 4. In this low power mode the CPU is disabled as well as almost all of the clocks, and the controller only draws around 0.2 μA. The controller can wake-up from any of its low power modes in only 1 μs. Since the guest and security tags are battery powered, an ultra-low power controller is essential to the design.

The MSP430G2231 has 10 general purpose I/O and supports both SPI and I2C Universal Serial Interfaces. It also has 2 KB of flash memory and 128 B of SRAM. The internal clock runs at 16 MHz but the controller is capable of being interfaced with an external crystal. While the MSP430 comes in a variety of packages the 14 pin PDIP package was chosen for ease of handling, bread board testing, and final PCB assembly.

C. LCD Display

The LCD display on both the guest and security tags is what will provide the user with the information they need to know. In order to keep the size of the tags to a minimum, an LCD display that is capable of 2 lines with 16 characters each is used. The display operates at 3.3 volts with the current draw dependent on the brightness of the backlight. During testing with the backlight at 80%, the screen drew around 20 mA which again is important for a portable, battery powered device.

The LCD display is already attached to a serial "back pack" which means the display can be directly interfaced with the MSP430 microcontroller. The display is capable of serial communication with a baud rate between 2400 and 38400bps, but this project utilizes its default setting of 9600pbs. The incoming buffer can also store up to 80 characters. Shown in Fig 3 is the LCD display which is made by Sparkfun Electronics.

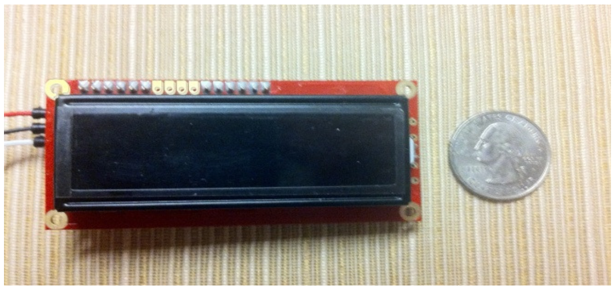


Fig. 3 Sparkfun Electronics 16X2 LCD Display used for guest and security tags

IV. Security Tag Detail

The security tag is a simple device that will be broken down into a detail of the overall component of operation, a brief hardware discussion, and a detail of the embedded programming.

A. Operation

The security tag is meant to be carried by security personnel or anyone who needs to monitor the system for trouble. The security tag alerts the user if there is a security breach within the building or if a user with a guest tag has pressed their emergency button. The security tag consists of the three main components listed previously, the radio transponder, the microcontroller, and the LCD display. The security tag also contains two LED indicators on the top, one green for power on, and one red for low battery. There is also a momentary push button mounted on the top that acts as the "trouble acknowledge" button. Like the guest tag, the security tag's location will be traceable on the GUI.

When the security tag is first powered on, the LCD will display a message saying "No Security Issues to Report". Settings within the GUI will allow certain rooms to be defined as "off limits" or restricted access. If a guest tag enters one of these rooms, the security tag will sound a buzzer for 1 second and the LCD will display a message saying "SECURITY BREACH IN ROOM X" with X being the room in which the breach occurred. The message will stay on the screen until either another breach occurs, or the user presses the "trouble acknowledge" push button. The second function the security tag monitors is a user emergency. If someone with a guest tag presses their emergency button, the security tag will sound a buzzer for 1 second and display a message saying "EMERGENCY IN ROOM X" with X being the room in which the guest tag that pressed the button is located. As before, pressing the "trouble acknowledge" button will clear the message from the screen.

B. Power

The power system of the security tag needed to be carefully considered since the unit will be battery powered and should last at least a normal work day of 8 hours. The LCD display requires 3.3 volts and because the microcontroller and radio both have input voltage ranges that include 3.3, this is the value that was chosen. The voltage regulator for the security tag, as well as the guest tag, is an MCP1700 3.3 volt regulator. The regulator has an output current of 250 mA and a drop out voltage of 178 mV. The total current draw of the security tag as around 120 mA, so the regulator will power it without a problem. The battery that powers the portable tags is a Sparkfun Electronics 2000 mAh polymer lithium ion battery which outputs a nominal voltage of 3.7 volts. Using the battery life formula (1) we can calculate the expected battery life to be approximately 11 hours which meets our goal of 8 hours.

$$Battery\ life = \frac{Capacity\ (mAh)}{Consumption\ (mA)} * 0.7 \qquad (1)$$

As with any battery powered device, a low battery indicator is essential. The low battery circuit we designed for our tags is based off of a voltage divider network. The output of the battery will go into a voltage divider which is then connected to two PNP transistors. When the battery voltage drops below the value we set, approximately 3.3 volts, a red LED will turn on, and a pin will be set high on the radio. This will allow the user to see the battery is low and will also display a message on the GUI showing that a specific tags battery is low. Shown below in Fig 4 is the low battery circuit utilized by the tags.

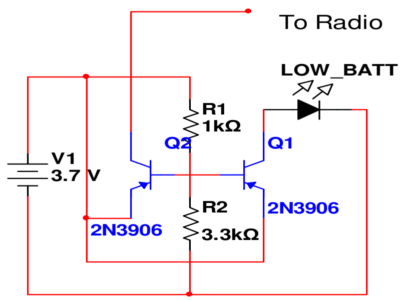


Fig. 4. Low battery detection circuit consisting of a voltage divider and 2 PNP transistors.

C. Microcontroller Code

The MSP430 microcontroller handles all of the print outs to the LCD display by reading in data from the radio. After the tag is first turned on and displays the initial message, the controller then goes into low power mode 4 which is the lowest power mode of the MSP430. At this point the controller can be awakened by one of 2 interrupts from the radio, the security breach or emergency signals. Once the microcontroller receives an interrupt from the radio, it then runs the room function. This function simply checks the status of three input pins that are connected to the radio. The three pins act as three binary bits to represent room numbers from 0 to 7. Once the room number has been read in, the LCD display is updated with the correct message and room number. The microcontroller then immediately drops back into lower power mode 4. At this point the LCD displays the message until another interrupt is received and the process starts over, or the user presses the "trouble acknowledge" button. The "trouble acknowledge" button it directly connected to the microcontroller's reset pin so when it is pressed, the controller resets, as do all variables within the program. Shown in Fig 5 is a flow chart of the microcontroller code

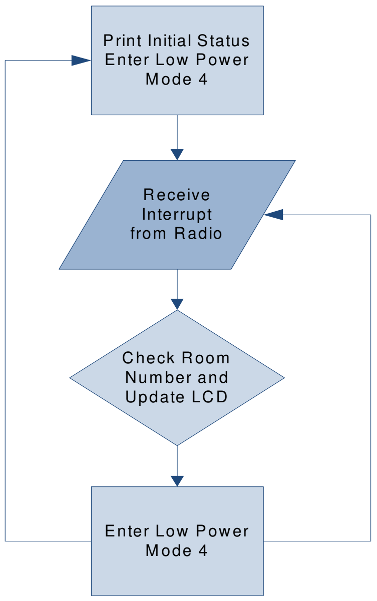


Fig. 5. Flow chart of the code running on the MSP430 microcontroller

V. Guest Tag Detail

The guest tag, in terms of hardware make-up and embedded programming, is very similar to the security tag discussed above. Its only function is to actively transmit its address in an infinite loop until it is asked to do one of two things by the user. The guest tag includes two push buttons that allow the user to a) identify their current room location and b) signal an emergency distress call to an authorized person if needed. Fig 6 shows a simple block diagram of the hardware interface of the essential subsystems of the guest tag.

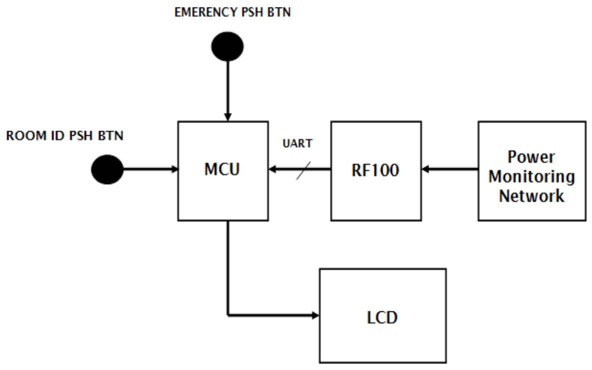


Fig 6. Block diagram of the guest tag module subsystems.

Testing has proven that there is a significant variance in the stability of the raw data being sent to the GUI for interpretation based on the environmental conditions of the room itself. Once the raw data reaches the GUI, various techniques are used to compensate for these signal strength variations. These methods are discussed in subsequent sections. However, the Python script controlling the radio module requests a broadcast about every one second. Once the transmit routine is complete, the radio module is set into a low power mode before entering into the next transmit routine. While in LP mode, the current draw of the radio from the battery is only about 1.6 μA compared to 115 mA during transmit. The system was specifically designed to have the portable, battery operated devices have the capability of powering down while the stationary devices are always actively listening. Certainly, increasing the transmit interval will conserve battery life; however the accuracy of the system will decrease. Sending updates to the GUI more frequently allows the averaging and error correcting to be more effective. The transmit frequency can be easily adjusted over-the-air to optimize battery life and accuracy for a particular environment. If a lower level of accuracy can be tolerated, the radios can be set to 3 seconds or less between broadcasts for longer battery life.

Similar to the guest tag, the secondary microcontroller remains in LP mode and waits for user input. The program flow chart shown in Fig 7 describes the primary operation of the external MCU.

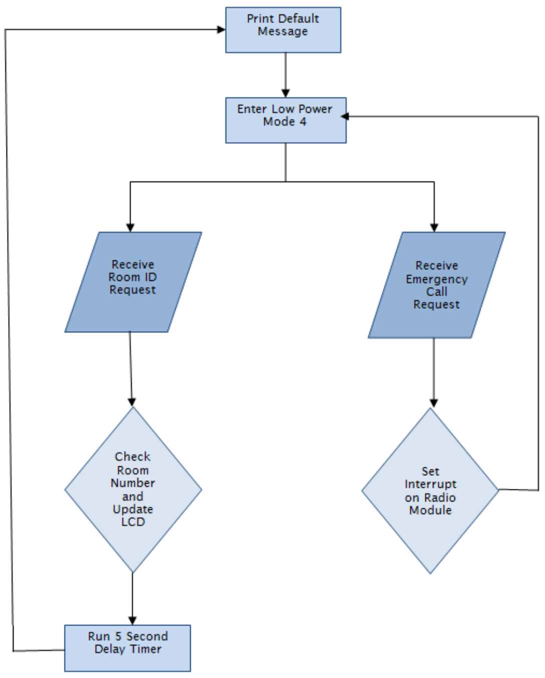

Print Default Message
Enter Low Power Mode 4
Receive Room ID Request
Receive Emergency Call Request
Check Room Number and Update LCD
Set Interrupt on Radio Module
Run 5 Second Delay Timer

Fig 7. Program flow of external, secondary microcontroller controlling interrupt request from the guest tag.

VI. Board and Packaging Detail

The schematic as well as PCB board designs were done with Eagle software. The boards were not overly complex and we laid them out in a manner which suited each of the tag designs the best. We tried keeping the board size to a minimum because the tags should be as small and portable as possible. The PCB for the guest and security tag are each 2 layer boards. All of the parts were chosen to be through-hole packages so that no special equipment was needed to assembly the boards such as what would be needed for surface mount parts.

The size of the case was largely determined by the size of the LCD display. The case for the security and guest tags is black ABS plastic and is 4.25"X 3"X 1.5" in size. The serial board attached to the back of the LCD display makes the display nearly 1 inch in height so a case was chosen that could accommodate this. The front main face of the case is where the LCD display is mounted while the indicator LEDs and acknowledge switch are located on the top for the security tag. On the left side of the case is a small rocker switch that is used to power on the tag. All of the internal connections to the PCB are done via pin headers and wires. The LCD display, power switch, reset switch, and buzzer are all detachable from the PCB for ease of disassembly and trouble shooting.

VII. Reader Mesh and Base Detail

A. Base

The base unit functions as the main node hub in our project. Its primary responsibility is to manage the data going into the GUI, and to route the data that the GUI sends out to the mesh of nodes. To simplify the coding in our project we decided to use ASCII values for all transmissions rather than having a mixture of different styles. The nodes in our project will be continuously sending updates regarding the most recent tag hits they receive. The security tag will receive signals to indicate that either a secured area has been breached of is a visitor tag has pressed their emergency button to indicate they are in need of assistance. The security tag will also be able to send an "all clear" signal back to the GUI to indicate that the emergency situation has been dealt with and there no longer a need for a security response. The protocol for a radio to communicate with the base is very straightforward and consists of an op-command followed by two to three arguments as shown in Fig 8.

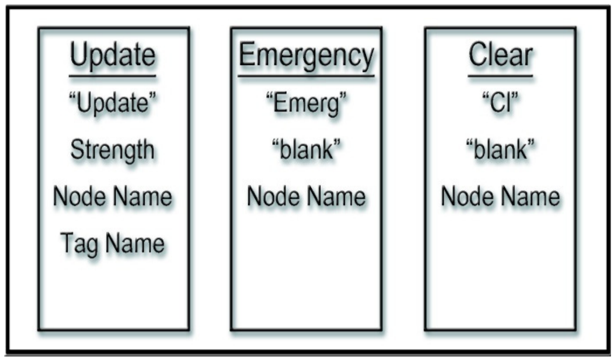


Fig 8. This image shows the protocol for a radio to communicate with the base or bridge node.

The "Update" op-code sends a strength value as its first argument. This is the value of the signal strength of the tag that most recently hit it. The next argument is the node name. For simplicity we established that nodes would be known by their address. The final argument is the tag name. Since the system can support many tags, this value is required to distinguish what tag was the original source of the communication with the node.

The "Emerg" op-code sends a "blank" as its first argument. This purpose of this blank is simply to help keep the processing of the various op codes consistent. The second argument is the node name. This is the name of the node that the visitor tag was located at when they pressed their emergency button.

The "Clear" op-code is similar to the "Emerg" op-code. Its first argument is a blank. The second argument is again, the node name. Since the security tag doesn't have to be standing in the exact location that the initial emergency was indicated, the node name is saved in the radio and when the clear command is sent this info is included.

B. Mesh Network

The Synapse RF100 radios that we are using in our project automatically form a mesh with other radios that are within range. This mesh works to extend the range of the radios by forwarding signals sent by radios at one end of the mesh through intermediate nodes and finally onto its destination. If a node in the mesh should fail (due to power failure etc...) the mesh routes around this dead node.

The code in the GUI is structured to account for dead nodes and takes this node out of future calculations. Since we intended to make the project robust enough to work in very large buildings this particular feature we felt was vital to our success. While the radios have a very large range outdoors with few obstructions, the distance is significantly lower inside a building. The large amount of walls, ceilings, floors etc., work to attenuate the signal and if it wasn't accounted for would result in a range that was almost useless for any real world applications. The following diagram shown in Fig 9 illustrates how the forwarding works.

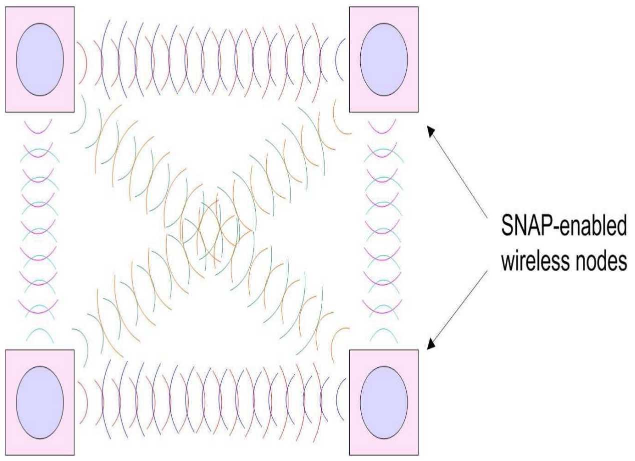


Fig 9. This Image shows the data forwarding capabilities of the mesh network.

VIII. GUI Detail

The main purpose of the GUI is simply to display graphical data to the end user – in this case, security personal sitting a monitoring station. The screen will display the current location of any active tags, as well as their current battery status. As a tag moves throughout the mesh of nodes the location of the tag will be updated on the screen to view. The type of tag – in this case either a visitor or security tag – will be indicated by the type of icon that is displayed. In the event that an emergency alarm is raised an icon will appear at the node itself on the screen. If a security breach is detected the icon of the visitor tag that is located in the restricted area will change to reflect the breach.

The choice in icons for this project is very important. Since the project rests upon presenting data to security personal in a manner that is easy to read and grabs their attention for situations in which they need to react quickly we chose the following representations of icons to display in the GUI.

The icons that show a visitor tag that is within an area that they are allowed to be in is fairly mundane – a black dot. When the visitor enters a restricted area the icon changes to a bull's eye. The sudden changes in color in the different areas of the bull's eye are more noticeable to the human eye and help security personal pick out serious issue quickly. The icon to indicate an emergency is a black and yellow spiral. Here, as with the bull's icon, the motivation was to use contrasting colors to quickly bring attention to the situation. The security icon is very similar to the visitor icon and varies only in color – in this case green with a red dot. We wanted the security icon to stand out a little more than visitor one since it can help security personal notice where their coverage might be a bit thin if a large number of visitors are present. With these choices in icons security personal should be able to keep a general eye on the visitor tags that are moving about the building, while at the same time respond quickly to situations that are deemed abnormal and require security personal to respond in person.

A. Node Detection

One purpose of the GUI is to determine which node a particular tag is closest too and place the icon that represents that tag at that location for the user to view. This is accomplished by evaluating the most recent hits from all of the nodes and calculating which signal was the strongest. As the tag moves throughout the mesh it continually pings the nodes. The data from these pings is then sent back to the base. As new data comes in for each node, it overwrites the previous data. In the best case situation, each node is able to report on these pings during each iteration of a location check; however, this does not always happen. If a node misses an update the old data is simply used. The thought process is the data will not change dramatically between one or even two iterations. If the node continues to not update, the node is then taken out of the calculation until it's determined that it is now functioning properly.

Ideally the walls, floor, and ceiling of a room will greatly attenuate this signal, making detection a bit easier; however, since our project is especially vulnerable to a variety of noise sources we developed a few strategies to help take them into account. These are discussed in the following sections. One last consideration is on where the tag is determined to be once we've established which node it is closest to. While a project using trilateration would give a more specific location, we felt placing the location of the tag at the spot of the closest node was more than sufficient for location detection within a building. When setting up the mesh, the user will want to be aware of this fact. If the system is being used in a very large room, more than one node may be desirable for detection in this situation.

B. Thresholding

In our initial testing we found that there was a large amount of "ping-ponging" that took place when a tag approached the mid-point of two or nodes. This was most prevalent when near large noise sources, but there was still some variance in signal strength even in an ideal setting. The result of this was the icon indicating where the tag was located to jump very rapidly between the competing nodes. This particular problem doesn't impact the accuracy of our project – one could technically consider the location to be either node at this point – but it does make the GUI more difficult to use for the end user. The solution that we utilize helps eliminate this issue while not sacrificing accuracy.

The approach we took to help account for the ambiguity was to establish a threshold level. This works by only allowing the GUI to consider a tag to have moved to a new node only after the current one's value has been exceed by a set amount. After a move has taken place the value of the new location is considered checked at each iteration of the check to determine if the threshold has been met again.

This approach greatly reduces any ping-pong type effects that occur when nearing the midpoint between two nodes. Since all the nodes have slightly different levels of variance based on their proximity to noise sources and other factors, this value is able to be set at any time from the GUI. This helps ease the setup and fine-tuning process.

C. Successive Hits

While testing the stability of the strength of signals between nodes and tags we noticed that we would occasionally get a large spike in signal strength. If we did not deal with this particular issue the icon indicating the current location would occasionally move to some random node before moving back when the spike was over. Since our project includes the ability to detect when a tag has moved into a restricted area and notify a person carrying a security tag, this would create a very serious issue for the end user.

The approach we took to deal with these spikes was to only allow the GUI to change node locations once a number of successive hits have been made. By only counting a set amount of successive hits we greatly eliminated the majority of these spikes. While it may seem like this would impact the accuracy of the system, it didn't have any real noticeable impact on what we saw – in fact it ended up being much smoother. The reason for this is the speed at which we sampled signal strength from the mesh. The speed of this sampling is so much faster than the pace human movement that it did not present a problem for us. This technique also utilizes the thresholding that was previously mentioned. These two combined together allowed for very smooth transitions with a great deal of accuracy. As with the thresholding approach, we wanted to be able to change the amount of successive hits that we consider a "success." This functionality was built into the GUI and is generally used when initially setting up the mesh.

D. Serial Data

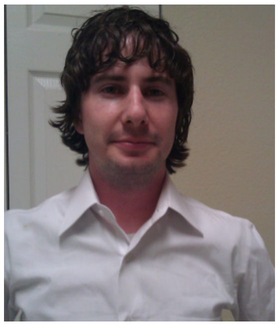
The GUI also manages the serial data that comes into and leaves the PC. Due to the amount and speed that the data is being received we decided to run the portion of the program that fetches the serial data in a separate thread. This resulted in a much smoother experience.

As the number of nodes increases the probability that some of the data will be received out of order increases. To help deal with this the GUI will automatically flush the serial data buffer if an error is detected. While the accuracy of the current location will suffer when this takes place, the speed of the updates being received from the nodes will make this effect minimal.

In addition if the computer the GUI is being run on has many processes running at once, it's possible that the buffer could become clogged with data that is now old. When this takes place the data is also flushed and the process started anew.

The three signals we don't want to have lost are the security breach, emergency, and clear. We dealt with this situation by having the tags continually send the signals until they are dealt with.

THE ENGINEERS



Daniel DeFazio is a senior Electrical Engineering student at the University of Central Florida. Daniel plans on graduating in December 2011 with his Bachelor's in Electrical Engineering. He is currently working for Staples as a Master Computer Technician. His interests include space system engineering, communication and signal processing, and embedded systems development.

Brandon Tuero is a senior Electrical Engineering student at the University of Central Florida. Brandon plans on graduating in December 2011 with his Bachelor's degree. He is currently working for Universal Studios Creative where they work on new attraction development. He plans to continue working at Universal upon graduation.

Matthew Rhodes is a senior Computer Engineering student at the University of Central Florida. He plans on graduating in December of 2011. His interests include embedded systems and software development.

ACKNOWLEDGEMENT

The authors wish to acknowledge the contribution and support of the EECS faculty especially Dr. Richie who gave us valuable advice and direction in critical moments of the project.

Also, we want to thank and express our gratitude to the professors who showed interest in our project and decided to be in our review committee.

REFERENCES

[1] Synapse Technology Incorporated. (2011) RF100 Data Sheet 43014-01B. Retrieved 15 August 2011, World Wide Web: http://www.synapse-wireless.com/documents/products/Synapse_RF_Engine_RF100PC6_RF100PD6_Data_Sheet.pdf

[2] Sparkfun Electronics. (2007) SerLCD v2.5 Data Sheet. Retrieved 10 September 2011, World Wide Web: http://www.sparkfun.com/datasheets/LCD/SerLCD_V2_5.PDF

[3] Texas Instruments. (2011) MSP430G2x21 Mixed Signal Microcontroller. Retrieved 02 July 2011, World Wide Web: http://www.ti.com/product/msp430g2231

[4] T. Igoe, "Wireless Communication," in *Making Things Talk*, 8th ed. Sebastopol, CA: O'Reilly Media, 2007, Ch. 6. pp. 198 - 207