Uzel Bagley, Nicholas Gerstle
Dru Harrison, Adam Marshall

Real Time Localization in the Edgar Mine using a Wireless Sensor Network

I. Abstract

The Edgar Mine, located in Idaho Springs, is an educational mine owned and operated by the Colorado School of Mines. A wireless sensor network that records temperature and lighting was deployed in one section of the mine from a previous project. However, the temperature is consistently 51° F year round. In addition, the sensors were enclosed in PVC piping that prevented them from properly detecting light levels.

The previous network has been replaced with a new sensor network containing updated software that is much quicker and records more useful information. The network tracks the location of "mobile motes", small circuit boards with sensors and a small radio antenna attached, which visitors and workers carry upon entering the mine and will be tracked by custom infrastructure. The location information and ambient sensor data is displayed on a dynamic website, and assuming the internet link to the mine is up the data will be displayed on the webpage.

The system now in place collects and refines both environmental and location data which has the potential to increase safety in the mine.

II. Introduction

Doug Hakkarinen and Alan Marchiori are graduate students at the Colorado School of Mines. They were in need of a field session group to program and setup a multi-platform network at the Edgar Experimental Mine in Idaho Springs. After recent mine collapses across the country and around the world, it quickly became evident that one of the greater challenges to mine rescue operations is locating people trapped inside.The proposed system allows individuals carrying mobile motes to be tracked as they move through the mine, recording their location, and displays their current positions on a website. The website also displays ambient sensor information from static locations throughout the mine. Since the readings from the three sensors are expected to remain fairly constant year round, any drastic changes in a reading could indicate an environmental hazard such as a fire.

III. Pre-existing System

A previous project implemented an environmental sensing network. The previous network infrastructure included ten motes, five USB-to-network interface boxes (Cisco router and a standard power-over-Ethernet adapter), and Cat-5 cabling. This network was removed, in preparation for the new network.

The ten infrastructure motes as well as undamaged Cat-5 cable were reused in the upgrade, but the previous USB-to-network interfaces were replaced with Quanmax embedded PCs. In addition to the Quanmax PCs, two servers at the Colorado School of Mines were added to the revised infrastructure.

The servers located in Golden include an application server and a database server.

IV. Requirements

The system incorporates four stages. The first stage involves several mobile motes transmitting a packet over radio, where they are picked up by stationary infrastructure motes. The infrastructure motes send both the received signal strength indicator(RSSI) and the link quality indicator(LQI) of these counters, along with measurements from attached environmental sensors, to Quanmax embedded PCs via USB. The Quanmax systems run the Debian operating system with a local MySQL database to hold data collected from the motes.

The Quanmax boxes store the information in a local MySQL database, before sending it to the servers located at Mines. The final stage consists of a localization algorithm using a nested select query that provides the most recent location of mobile motes in the mine. This information is presented in a visual form by a website with a map of the mine indicating the ten sectors that mobile motes might appear in. The most recent environmental data from all ten infrastructure motes is displayed on this page as well. (See Appendix 2, Figure 1)

Software development included:

- sending and receiving data on the motes (both mobile motes and infrastructure motes),
- entering raw data into the Quanmax databases,
- transmitting data from the Quanmax to the database server (through an open port on the application server),
- processing raw data into usable localization data,
- presenting localization and environmental sensor data on a web page updated in near real-time.

The software developed for the motes is embedded directly into the operating system, TinyOS. For this reason, existing software on the motes was completely overwritten with a "new" version of the OS with the added functionality for this project.

A. The functional requirements as specified by the clients were:

a. The system must locate at least two mobile motes in the square route of the mine installed with the network infrastructure (Miami between B-right and C-right, B-right between Miami and B-right-first, B-right-first and C-right between Miami and B-right-first tunnels) (See Appendix 2, Figure 8).
b. There must be a web-based application hosted on thor.mines.edu that displays the position of any active mobile motes in the mine.
c. The web-application must display sensor data including light, temperature, and humidity levels from the infrastructure motes (motes located in the mine at a fixed position, using different software than the mobile motes).
d. The intermediary Quanmax computers must save and buffer collected data in case of network failures.
e. The system should allow for data to be collected from the mobile motes.

f. Mobile mote software should maximize energy efficiency and preserve battery life.
g. The system should include a mechanism for automatic remote 'pulled' updates (meaning a physical visit and a direct connection to devices are not required).
h. There should be variable localization schemes, with a description of the situations each is best for.
i. The infrastructure and mobile motes should have software maximizing wireless network efficiency, dependent on the number of motes and power usage.
j. The system should be designed such that signals between mobile motes and other mobile motes can be used for relative localization.

B. Non-Functional requirements:

a. Existing mine infrastructure should not be damaged.
b. Equipment needing new software includes:
    i. 12+ telosb Tmote sky motes
        - 802.15.4 Radio
            - Chipcon CC2420
            - Integrated on-board antenna with 50m range indoors / 125m range outdoors
        - MSP430F1611 CPU
            - 10Kbyte RAM, 48Kbyte Flash
            - Very low power in periods of inactivity
    ii. 4 Quanmax PC's (embedded)
        - Atom processor
        - 1 GB ram
        - 160 GB hard drive
    iii. 2 Servers:
        - modi.mines.edu: Debian, MySQL database, behind firewall.
        - thor.mines.edu: Debian, firewall holes at ports 20, 80, 443, running the Apache web server.
c. Infrastructure:
    i. The 10 infrastructure motes will be USB powered, and attached to the Quanmax boxes, 2 or 3 per Quanmax.
    ii. The Quanmax boxes will be powered via their Cat-5 network connection, using standard power over Ethernet.
    iii. Infrastructure motes and Quanmax boxes must be packaged against moderate humidity and physical disturbance.
    iv. Infrastructure devices must be attached to existing mine infrastructure with zip-ties.
d. Project development:
    i. Use SVN as a repository for code development
    ii. Use the Trac system on modi (modi.mines.edu/projects/2010_CSM1/) to document progress, goals, programs, etc.
    iii. Use TEP-3 standards for TinyOS code in nesC.
    iv. Use appropriate language standards for all other code.

v. Install system in Edgar Mine, including Cat-5 and USB wiring.

All non-functional requirements are absolutely required. Functional requirements a, b, c, and d are absolutely required. Functional requirements e through h are secondary requirements, to be fulfilled, time permitting. Requirements i and j compose possible future efforts, and require more research.

V. Design

The Edgar Mine Wireless Sensor Network (WSN) consists of a number of separate but connected systems. The first two systems use the same hardware, MoteIV Tmote Sky wireless sensors, but run different software. The third system runs on Quanmax embedded PCs and includes a Java message listener and a MySQL buffer database. The fourth system contains the main MySQL database on an internal-facing server at the School of Mines and processes the raw data received from the mine. The final system hosts the public website that will display the data collected from the mine.

As seen in the Architecture diagram (Appendix 2, Figure 1), this project is best separated by the different hardware platforms and the links between them. As the tracking of the mobile motes was a key requirement, the design effort began at that level.

A. Mobile Mote

The first system of Tmote Sky sensors are called “mobile motes” because they will be carried by people in the mine. The mobile motes broadcast a packet every quarter second via radio that is received by a second system of stationary Tmote Sky motes, here after termed “infrastructure motes”. Mobile motes broadcast a simple counter, in order to allow checks for missing packets. Lastly, the lights blink the counter value (in binary) in order to assure the user that their mobile mote has batteries and is working.

To this end, the sample BlinkToRadio App from the TinyOS tutorials was slightly modified. The BlinkToRadio App sends a counter over radio to other motes running BlinkToRadio. Upon receiving a counter, the mote sets its LEDs to display this received counter in binary. Thus with two motes, each will display the counter of the other. Altering BlinkToRadio by removing the incoming radio code, and changing it to display its own counter, allowed us to use the rest of the application as the mobile mote code. (See Appendix 2, Figure 2)

B. Mobile Packet

As noted above, the packet used to exchange information is the simple packet containing only a header and the counter present in BlinkToRadio. All information concerning the source, etc, is available by accessing the header. (See Appendix 2, Figure 3)

C. Infrastructure mote

The infrastructure motes consume the radio messages to calculate the signal strength and read the mobile mote ID. A new packet is created containing the ID, as well as the signal strength indicators, from an incoming mobile mote packet, and is then forwarded to the Quanmax PCs via USB. In addition to the mobile packets, the infrastructure motes periodically take environmental sensor data (light, temperature, and humidity) and send it via USB. The infrastructure motes are still enclosed in the same PVC cases, however test data shows that there is enough variance in the received Lux values to be able to identify if a light is on in close proximity to the mote. (See Appendix 2, Figure 4)

D. Infrastructure packet

The infrastructure mote packet consists of three 16 bit fields and three 8 bit fields regardless if a particular instance contains localization data from a mobile mote or environmental data from the infrastructure mote itself. The 16 bit fields contain the mobile mote ID or sensor type, the sensor data or nothing (if localization data), and the counter corresponding to whichever type of data it is. One 8 bit field is either a 0 or 1 to help the java listener determine what type of data it is, either localization or environmental. The other two are only filled if the data is localization data. In this case, they contain the RSSI and LQI of the mobile mote. (See Appendix 2, Figure 5)

E. Quanmax

Per the client's request, the Quanmax embedded PCs run Debian Linux, with the full TinyOS toolchain, Java, vim, links, and an SSH server for easy access. They have been configured to pull and update from a specific package on thor.mines.edu via cron in order to allow updates, due to a non-existent ability to push information or commands to the machines while they're in the mine. While running, the PCs simply listen for packets and enter them into the buffer database, before sending data to thor.mines.edu.

F. Java Listener

The current Java listener - the interface between the incoming serial stream over USB from infrastructure motes and the database on the Quanmax - is an expansion upon the listener code created by School of Mines graduate student Josh Thomas. This code allows one instance to listen to several motes as specified in a configuration file, instead of requiring an instance to be spawned for each attached mote.

The program is a listener extending MessageListener provided by the TinyOS toolchain. A call to main includes a configuration file consisting of the serial interfaces the listener is then attached to. Upon reception of a packet, the program determines if it has received location or environmental data, which then is inserted into the database. (See Appendix 2, Figure 6)

G. Quanmax Database Schema

The first database that the Edgar Mine system uses is described as follows (see Appendix 1 for a more detailed schema).

entries

Id	Serves as a primary key to track number of entries
Imote_id	Infrastructure mote ID (1 – 10)
Mmote_id	Mobile mote ID (1 – X)
Rssi	Received Signal Strength Indicator value
Lqi	Link Quality Indicator value
Counter	Tracks the number of packets sent by a mobile mote
Occurrence	Time data is inserted into entries
Dirty	Acts as a boolean variable for pushing to Modi(sent/unsent)

The following is an abstract description of the tables that hold temperature, humidity, and light data, with each type of data in its own table.

temperatures/humidities/lights

Id	Serves as a primary key to track data inserted
Imote_id	Infrastructure mote ID
Value	Value of enviromental data
Counter	Tracks the number of packets sent by an infrastructure mote
Occurrence	The time data is inserted into an environmental table
Dirty	Acts as a boolean variable for pushing to Modi(sent/unsent)

H. JSONRPC

The client provided a sample python daemon which selects multiple rows from the local database, encodes them, and sends the raw data to a web service on the public-facing web server(thor.mines.edu). This server forwards the data by creating a new request to the internal database server (modi.mines.edu) using the same type of web service. The database server decodes the data and enters it into the appropriate table, either the localization entries table or one of the three environmental tables. Upon successful execution through all levels, the rows in the Quanmax local database are updated to reflect that they have been successfully sent and stored.

I. Modi Database Schema

Because all of the data from Edgar Mine is pushed to Mines, the database schema on Modi matches the schema at Edgar Mine, except for a few additions including a locations table and entries history table for entries and locations that are identical to their counterparts.

The locations table is the product of the localization algorithm showing which infrastructure mote a mobile mote is closest to. This table is created using a MySQL feature called a view. A view is a table that is created at the time a query selects from it. It lets a user select a portion of data from a database, including from different tables, to view in table format. The view used for the locations table shows the most recent packet activity in the mine for each mobile mote. Once the recent activity in the mine has been gathered, the infrastructure mote which received the highest RSSI value can be calculated.

locations

Mmote_id	Mobile mote ID
Sector	Sector where the mobile mote is located
Strength	Measure of signal strength
Counter	Tracks number of packet sent by the mobile mote
Occurrence	Time at which data was entered into the database in the mine
Dirty	Boolean variable

J. Localization Algorithm

The algorithm used to approximate the location of a mobile mote in the mine is rather course. The algorithm finds the most recent packet sent by a certain mobile mote, and checks to see which infrastructure mote received a stronger signal from it. This algorithm takes the form of a complex MySQL query that is executed by the webapp when a visitor opens it in his or her web browser. This process is time consuming and only produces a low resolution localization result. This is not ideal, but due to time constraints and various issues encountered between the JSONRPC web server and MySQL triggers this was the easiest option to implement.

K. Webapp

The webapp (hosted on the public facing web server, Thor) is used to display the data from the database on Modi in an easy to read fashion. It consists of a picture of the mine, where each section is highlighted a different color, and two tables that will display environmental data and the locations of mobile motes. Each row of the mobile mote table is colored corresponding to which section that mote is in for readability. All information is updated every ten seconds by a javascript framework called prototype. Every ten seconds, an external script is called that connects to the database on Modi, executes the queries required to gather the information, and displays this to the website. (See Appendix 2, Figure 7)

VI. Implementation Details and Results

Given the requirements listed above, the design of the system was focused towards limiting its complexity. In the mobile motes, this focus manifests in longer battery time; in the infrastructure motes, this encourages independent environmental sensor and wireless transmission data

gathering; with the Quanmaxes, this encourages the use of a well understood, well documented system.

A. Mobile Mote

The mobile motes use the requisite TinyOS, which is programmed in a variant of C called nesC. The mote's task is to send a unique id and counter via packet by radio. The current version sends a packet every quarter second (250 ms), which was selected as the result of an unscientific approximation of the time needed to navigate one section of the mine as defined by different infrastructure motes. This rate is static, and may be susceptible to 'echos' in the tight mine tunnels.

Alternate schemes considered, though discarded due to their complexity, include some sort of variable rate transmission, based on the mobile motes calculated location, as drawn from the reception of IDs and comparison of the signal strengths of the closest infrastructure motes. Such a technique might allow for lower power usage during times when the mobile mote is not in motion, as well as a straightforward indication of the transition of the mobile mote between two segments of tunnel controlled by two different Quanmaxes.

The primary downside of this scheme becomes clear after the realization that for a given period of time, power needed to transmit information is nearly equal to the power required to receive information, meaning that any power saved due to fewer transmissions is more than lost when listening and calculating location. Furthermore, such a system would significantly increase the complexity of the code for the mobile mote, requiring four to ten times as much time, at best, to implement.

B. Infrastructure Motes

The infrastructure motes, using the same hardware as the mobile motes, require the same TinyOS and nesC code. The current iteration of infrastructure mote software is the result of many attempts and branches. The majority of the code is the compressed result of the RSSI demo tutorial. The RSSIBase application, a part of the RSSI demo, contains code for multiple chipsets, and the clients allowed the elimination of code required for non-telosb chipsets (the non-CC2420 radios).
Sensor data is collected from the SensirionSht11C humidity and temperature sensors, and from the HamamatsuS1087ParC light sensor.

They are all called on the firing of a timer. The rate of five minutes, was unscientifically chosen: the interval should be fast enough that relatively recent environmental data is available. However, the environment does not change quickly, especially in the current packaging (see specification document for additional commentary).

C. Infrastructure Packets

The Infrastructure packet had many revisions to keep it small and able to access all the data needed. At first, environmental data and localization data was split into two different packets of equal length. This design was changed to refrain from having to listen for two different types of packets. Then a new packet design was implemented to hold various types of data in the same packet. This consisted of three 8 bit fields and two 16 bit fields. If it was a localization packet, one 8 bit field would be zero, the other 8 bit fields would hold the infrastructure mote ID of the mote that recieved the packet and the mobile mote ID of the mote that sent it, and the two 16 bit fields would hold the RSSI value and the counter of the mobile mote. If it was environmental data that was being sent, one 8 bit field would be 1, the other 8 bit fields would be the sensor type (light, humidity, temperature) and ID of the infrastructure mote where the data originated from. The 16 bit fields consisted of the sensor value and the counter of the Infrastructure mote. The 8 bit field that only stores 0 and 1 is 8 bits simply because that is how TinyOS treats a boolean.

This seemed like it would work nicely, but some researchers have suggested that the LQI provides a better basis for localization over RSSI. Therefore a slight change was made to the above structure to implement bit shifting to fit both the RSSI and LQI into the 16 bit data field. The issue with this is that the Java listener seemed to be unable to retrieve the correct data from the field. Because of the time constraints, each type of data was simply broken into its own field. If the packet was localization data, the RSSI and LQI fields were filled and the sensor data left blank. The opposite is true for sensor data. There is still room for improvement on the packet but overall it is efficient and provides sufficient data needed for the localization algorithm.

D. Quanmaxes

The Quanmax embedded PCs run Debian Linux (Squeeze). Debian was selected on the advice of the clients. Debian's strong set community and significant resources in the way of documentation encourage its adoption. The well-rounded packing system provides incentive to use it as the means of "pushing" updates to the Quanmaxes remotely.

E. Quanmax Database

The Quanmax database uses MySQL by client request. The database uses four tables to keep track of movement in the mine, temperature, humidity, and lighting. All tables are using the InnoDB storage engine. This engine was chosen because it grants the user a defragmentation tool to remove 'empty' tuples which may decrease preformace with large amounts of data. Each of the tables contain atomic values so there is no need to normalize further. An occurrence field with a timestamp type was chosen for keeping track of recent packets to avoid implementing a rollover case if recent packets were selected by a counter. The tables also contain a field named dirty. This field acts as a boolean to indicate if the tuple has been sent to Mines.

F. JSONRPC

As stated previously, a python daemon running on each of the four Quamax embedded PCs periodically selects data from each of the four tables (entries, lights, humidities, and temperatures), encodes it using JSON, and sends it to a JSONRPC web service running on the public facing web server at Mines. Currently this server acts as a forwarder that consumes the data from the Quanmaxes and sends it to another JSONRPC web service on the internal database server. Initially the system was designed so that the public webserver would consume the data from the Quanmaxes and remotely insert it into the database on the internal server without having to forward the data. However, this method produced numerous errors that given the time constraint couldn't be resolved. Moving the final web service to the internal server and writing a simple forwarding web service on the public server was the quickest way to finish linking the entire system.

After running into efficiency problems with the on-insert database trigger (detailed in the Modi Database section below), it was attempting to move the history trigger to the end of the insert statement of the web service. This however caused a MySQL Operational Error exception since data was being inserted and deleted data at the "same" time. This could likely be resolved in the web service but in the interest of time and scope, it was put aside for future work.

G. Modi Database

As mentioned above, the database schema used on Modi is very similar to the schema used at Edgar Mine. The entries_history table is used to keep the entries table relatively small, reducing the time it takes to update the locations table. The locations_history table was added on client request to view when a mote was active on a specific date. Currently the database has no indexes. This is because when pushing data onto a large table, the database has to re-index depending on what column(s) the index is on, and this may have been the cause of a significant time delay when pushing data from the Edgar Mine to Mines. This delay may also have been caused by a trigger that updates the locations table after an insert of new data into the entries table. Pushing data from the Edgar Mines to Modi caused numerous problems with triggers in the database. At this time the history triggers have been removed from the project, but are included in the Appendix 1d and Appendix 1e for reference.

H. Webapp

The web application consists of two files, edgarmine.php and updateMmotes.php. The file edgarmine.php is responsible for making updateMmotes.php run on time (every 10 seconds). The timer is created using the prototype Javascript framework which provides the PeriodicalUpdater call. This allowed for easy running of the external updateMmotes.php script and an easy way to change how often the website is updated. Originally, the map was going to be updated in much the same way. Different sections of the map would light up depending on if a mote was in that section or not. However, this addition did not get completed under the current time constraints and the map is now a static picture with each section of the mine a different color. The table consisting of

data about mobile motes will have each row colored corresponding to which section the mote is in for easy reading.

VII. Project Progression

The system produced fulfills the minimum goals as laid out in the requirements section. The system is capable of tracking and listing the location of multiple mobile motes, and can record and display the results of sensor testing. This information is gathered and stored at the Quanmax and server levels, and the passing of that information between them has been demonstrated.

At this point hardware-wise, the project remains incomplete, as situations and variables beyond the scope of the field session project have arisen. Chief among them is the constraint on the length of Cat5 cable such that it can reliably transmit data. One Quanmax PC is roughly 320 feet from the switch at the main junction in the mine. Without an Ethernet repeater capable of transmitting power-over-ethernet and data, this last Quanmax PC cannot be connected to the network. However, the remaining three PCs are connected and fully functional.

Initial milestones focused on creating the necessary software that components could begin to be tested- thus starting with the infrastructure mote and Quanmax platforms. This was to be followed by assembling the mobile mote software, and testing the flow of data from the ground through the Quanmaxes.

Once the raw data was consistently being gathered, growth was intended in the upward direction: building a connection between the Quanmaxes and servers in Golden, completing the localization algorithm, and finally creating the website.

As anticipated, the entire team began with the recommended TinyOS tutorials, and spent a full week learning the operating system. Setting up the Quanmaxes with Debian and the TinyOS environment took little time, though once the system became more developed, all the Quanmaxes were rebuilt (software-wise). The largest hang-ups began to develop during the second week, when efforts were made towards receiving and processing the incoming mobile mote packets, and finding the relevant sensor interfaces, took significantly longer and proved more challenging than expected.

During week three, the team began to diversify, and attempted to begin programming many more of the components in parallel- the database and algorithms on the servers in Golden, the code used to push data to them, and the serial to database interfaces on the Quanmaxes. This was achieved, and, for the most part, successful; a result of frequent discussions on the links between platforms, and the data storage organizations of each platform. The ability to receive and process signal strength of wireless packets at the infrastructure mote level was only completed at the beginning of week four.

A basic test was achieved at the end of week 4, resulting in the majority of week 5 to be used at the mine, wiring. A successful wiring with no errors was thought to take one full day, however, incomplete information led to it taking much longer than that. Week 5 included notifications after the initial wiring that the network infrastructure cannot be attached to power lines, and the

realization that while some lines could carry power, they were incapable of carrying information, and thus had to be replaced.

Furthermore, the expectation that the incoming line from the mine entrance would work proved false, leading to significant amounts of time devoted to restringing the 900 foot entrance tunnel.

All the absolute requirements have been fulfilled within the scope of the project, and some of the optional objectives, such as remote updates, have also been completed. Other goals, such as a comparison of RSSI vs. LQI have fallen to the wayside as unexpected reschedulings and wire breaks (as well as memory errors in coding TinyOS) caused significant delays.

VIII. Conclusion and Future Work

The Edgar Mine Wireless Sensor Network project is overall a success, fulfilling the requests of the client. In the current implementation, RSSI is used for locating mobile motes, but LQI is also sent to the database(s) to allow for improved localization algorithm. Having only worked on the project for six weeks, a few tasks are yet to be completed.

More efficiency could be achieved on the mote level if the various IDs of the motes were stored in non-volatile memory as to keep from needing to re-specify the ID everytime install occurs.

The wiring in the mine is currently not completed. The wiring in C-Right remains broken due to possible length and soldering issue.

No calibration was done when it came to determining if packets are being sent often enough or too often and if the channel they are sent on makes a difference. Also, no optimization was done on how much power the radio actually needs to send the packets a good distance.

One of the more interesting items to still figure out is if LQI or RSSI is better for localization. Both pieces of data are stored in the database but the algorithm for localization uses RSSI at the moment and no tests were run on LQI.

The infrastructure mote packet could still be optimized a little more and not waste so much space.

With the current Quanmax software, the data in the database will grow until there is no more space on the hard drive. There was not enough time to address this issue either. One other future upgrade for the database on modi.mines.edu would be an item that is planned to be released with mysql 5.1.6. MySQL plans to add scheduled events which allow users to run queries or statements periodically.

This project provided many unique challenges and a many great lessons were learned:

a. A mine provides many unique circumstances that much be taken into account when placing wires and embedded PCs. Certain guidelines must be followed to ensure that Cat5 cable is not attached to power lines or pipes which may cause interference or even damage to the

system. Also, as power is limited in a mine, it was important to figure out how to handle the situation where the PCs were unplugged to make room for drills and other commonly used tools. A mine is also not the most friendly environment for people. Between the self rescuer that saves lives 2% of the time it is required and the blasting of new tunnels, the mine provided many obstacles.

b. Running Cat-5 cables presented many challenges throughout the fifth week of the project which was spent entirely in the mine. Every cable that was brought down from the previous system had to be tested, and many of the ends had to be replaced. There was some experience with terminating RJ45 plugs in the group, but by the end of the week everyone had extensive experience. In addition to creating ends, the group learned first hand the length constraints of Cat5 for ethernet connections. After many unsuccessful attempts, the final section of the mine (tunnel C-Right) is still unconnected. The distance from the switch at the B junction to the Quanmax in C-Right is over 320 feet, which is extremely close to the rated limit of Cat5 cables, and well above the suggestion of 250 feet for the maximum length of cable. Without an ethernet repeater that can handle power-over-ethernet this last section cannot be completed.

c. Wireshark is a very handy tool when it comes to figuring out where in the network a connection may be broken. Being able to track ARP requests and replies enabled the team to locate the Quanmaxes on the network so they could be accessed via ssh. Wireshark was also instrumental when debugging the DSL connection in the mine.

d. Errors that have to do with bad packets/packets too long etc, are very difficult to troubleshoot and fix. These errors still pop up from time to time with seemingly little reason to why. Most of the time, a simple reinstalling of the mobile mote software would fix the problem but no permanent fix has been found.

e. Website programming can become very messy when dealing with more than just php and html. Trying to get multiple frameworks to see different variables and calling multiple scripts led to hard times and a lack of website functionality that would have made it complete.

f. The first week of the project was spent learning TinyOS, nesC, etc. It soon became apparent that working in the Alamode lab at the School of Mines was not going to provide the needed support since installing software on the motes required sudo access. Thus, a decison was quickly made to work locally on the team's own laptops. To standardize the environments, the most current version of Ubuntu was installed in a dual boot manner on two Apple MacBooks and an HP tablet. The fourth member ran Ubuntu virtually using the VMWare Player. Each method of running Ubuntu had it's advantages and disadvantages. Running virtually made it simple to move back and forth to applications that needed to run on Windows (or OSX), but passing USB connections to the virtual host proved cumbersome. The final step in setting up the development environment was to use an Eclipse plugin created for TinyOS. This required a good amount of configuration so that Eclipse could correctly interface with the motes.

g. While overall nesC is not a difficult language, various small items led to issues in early development. One such issue was the fact that all variables must be declared at the top of a scope. This is good practice but proved irritating when first trying to program the motes.

h. In the entire mine infrastructure there are three DSL modems. The first two sit in the office at the surface of the mine. One is a Qwest provided modem that brings internet access to the office. Connected to the router in the office like any other piece of equipment is another DSL modem that converts the ethernet data from the router back to a phone line. The phone line that runs into the mine over 900 feet in length. At the other end, at the B junction in the mine, the final DSL modem converts the phone line back to ethernet which is then connected to the switch, and thus the four Quanmaxes. The team did not realize that modems could be used in this manner. It proved absolutely necessary however due to the Cat5 length constraints mentioned above. The phone line carries a different voltage that allows it to be transmitted great lengths, thus able to provide internet access 800 feet underground.

i. Mines offers a database class here that teaches students basic database efficiency and queries. Select statements and efficiency is a much larger subject when applying it to more than just a database of movies and actors. Knowing basics and looking up instructions and tips online helped out a lot when trying to implement the localization algorithm with a select statement. The team discovered that group-by and views are extremely powerful tools when working with databases.

j. During one of the team transitions between infrastructure packet design, the RSSI and LQI values were in the same 16-bit field. It is very difficult to extract a portion of the field when Java insists on keeping values without letting the user perform a sign extended shift of bits.

k. Creating a Debian package, while not simple, greatly simplifies the install process on the Quanmaxes. Once a suitable guide was found, iterative changes allowed the production of a package that, after networking was installed, setup all the necessary production tools with minimal interaction.

l. It's extremely helpful to have good and frequent contact with the server administrater. Be sure all relevent php, python, python-JSONRPC, and python-mysql interface packages are installed and correctly configured.

IX. Glossary

- ARP - Address Resolution Protocol is used to map an IP address to a physical machine address.
- Atom - a low power x86 processor developed by Intel.
- Cat-5 - the RJ-45 (8P-8C) wiring specification, commonly used for wired computer networks.
- Debian - a Linux based operating system. Free.
- DSL - Digital Subscriber Line: internet access
- Infrastructure mote (Imote) – Stationary mote that is mounted to the mine ceiling.
- JSON - JavaScript Object Notation is a lightweight data-interchange format.
- JSONRPC - Remote Procedure Call implemented in JSON that allows bi-directional communication between the server and client.
- Mobile mote (Mmote) - mote carried with an employee or visitor for tracking.

- modi.mines.edu- a server located in Stratton hall at the Colorado School of Mines, Debian based, several services, including the internal project Trac wiki, svn repositories, and MySQL database. It is administered by Alan Marchiori and Nicholas Gerstle.
- Mote - a small, battery (or USB) powered sensing device, with minimal computing and storage capabilities, operating in a wireless, or mixed wired/wireless network. This project used telosb motes.
- Power over Ethernet - a specification allowing the transmission of power over Cat-5 and higher RJ-45 wiring, promising at least 12.95 watts of power at the terminal end.
- Quanmax - one of Quanmax computers listed in the requirements section.
- RJ45 - Short for Registered Jack 45, an eight-wire connector used commonly to connect computers onto a network.
- thor.mines.edu- a server located in Stratton hall at the Colorado School of Mines, Debian based, running Apache. It is administered by Alan Marchiori.
- TinyOS - a minimal operating system where specific program logic is embedded in the OS kernal.
- Ubuntu - A popular free version of Linux
- USB network interface - the actual hardware that was used included several Tmote Connects.
- VMWare Player - virtual machine hosting software developed by VMWare.
- WSN - wireless sensor network

X. References

[1]Kannan Srinivasan and Philip Lewis. RSSI is Under Appreciated.

[2]Jun-geun Park, Ben Charrow, Dorothy Curtis, Jonathan Battat, Einat Minkov, Jamey Hicks, Seth Teller, Jonathan Ledlie. Growing an Organic Indoor Localization System.

XI. Appendix

1. Detailed database schema

1a. entries/entries_history

Id integer primary key auto_increment
Imote_id tinyint not null
Mmote_id tinyint not null
Rssi smallint not null
Lqi small int not null
Counter integer unsigned not null
Occurrence datetime not null
Dirty tinyint unsigned not null default 0
type = InnoDB

1b. temperatures, lights, humidities (all three tables have the following format)

Id integer primary key auto_increment
Imote_id tinyint not null
Value smallint not null
Counter int unsigned not null
Occurrence datetime not null
Dirty tinyint unsigned not null
type = InnoDB

1c. locations / locations_history

Mmote_id tinyint unique not null
Sector tinyint not null
Strength smallint not null
Counter integer
Occurrence datetime not null
Dirty tinyint default 0

1d. This is the trigger on the locations table to track movement in the mine. The data is kept in a locations history table.

```
create trigger fill_locations_history after insert on locations
for each row
insert into locations_history select * from locations;
```

1e. Trigger is the localization method. This updates the motes' latest position in the mine displayed in the locations table. Take note that this is a multiple statement query. A delimiter must be set to something other than a semi colon. This trigger also populates the entries_history table and takes away old data from the entries table to let the localization preform quicker.

```
delimiter //

create trigger locator after insert on entries
for each row begin
replace into locations(Mmote_id, Sector, Strength, Counter, Occurrence)
select Mmote_id, Imote_id, maxRssi, Counter, maxOccurrence from
(select *, max(Rssi) as maxRssi from (select Mmote_id, Imote_id, Rssi, Counter,
maxOccurrence from recent as k where k.maxOccurrence = (select max(maxOccurrence)
as t
from recent where k.Mmote_id=recent.Mmote_id group by recent.Mmote_id))
as sub group by Mmote_id, Imote_id, Counter) as theMagic group by Mmote_id;
insert into entries_history select * from entries where Occurrence < now() - INTERVAL
10 MINUTE;
```

```
delete from entries where Occurrence < now() - INTERVAL 10 MINUTE;
end;
//
delimiter ;
```

2. UML Diagrams, Packet Diagrams, and Architecture Diagrams

Figure 1. Architecture Diagram

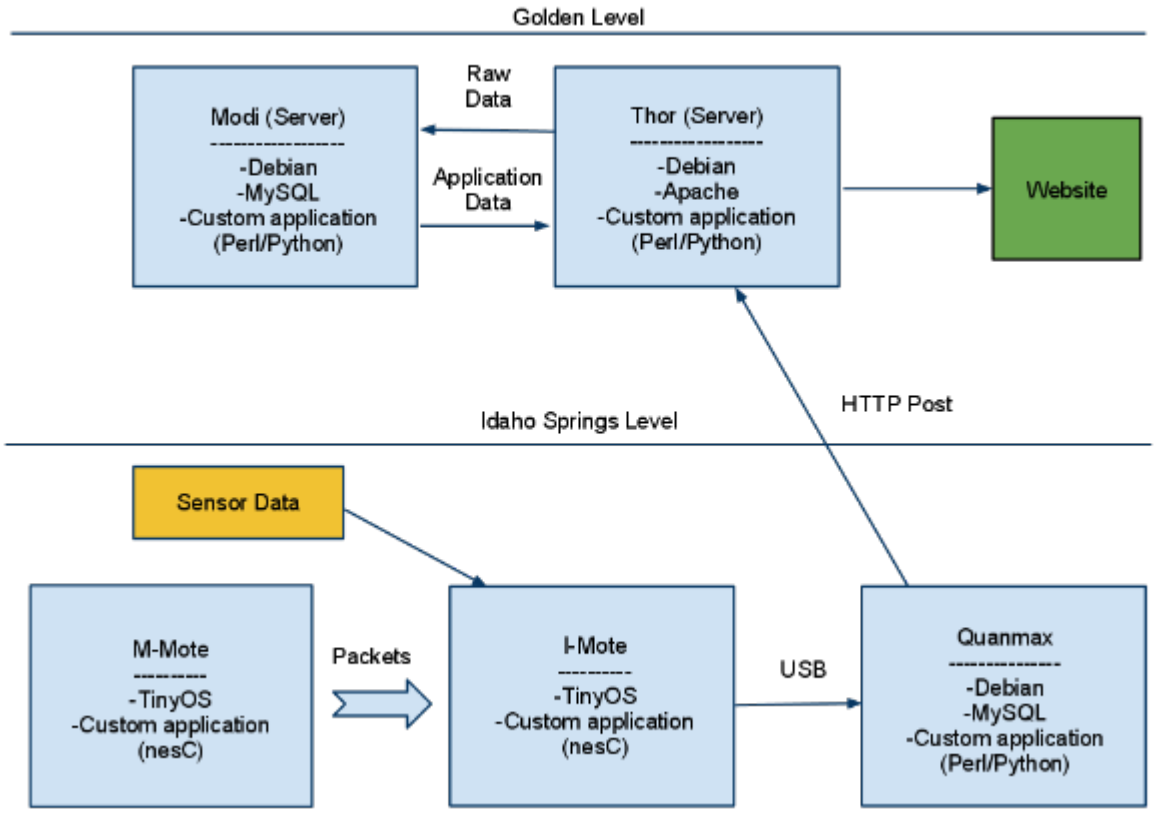


Figure 2. Mobile Mote Diagram

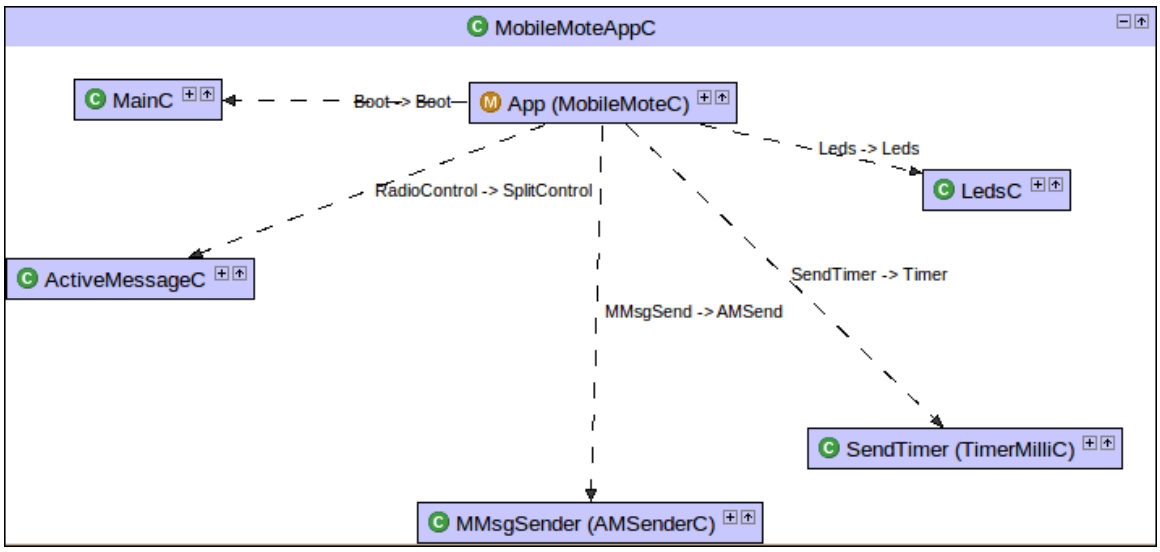


Figure 3. Mobile Mote Packet Diagram

Figure 4. Infrastructure Mote Diagram

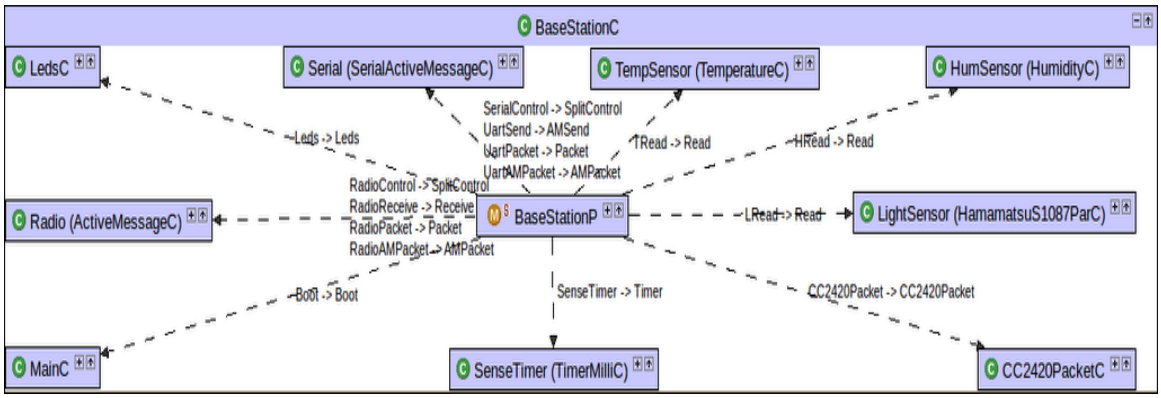


Figure 5. Infrastructure Mote Packet Diagram

Location Packet

Header	0	Mmote_id (16 bits)	Blank (16 bits)	RSSI (8 bits)	LQI (8 bits)	Counter (16 bit)	Footer

Environmental Packet

Header	1	sensor type (16 bits)	Sensor Data (16 bits)	Blank (8 bits)	Blank (8 bits)	Counter (16 bit)	Footer

Figure 6. JallBase UML Diagram

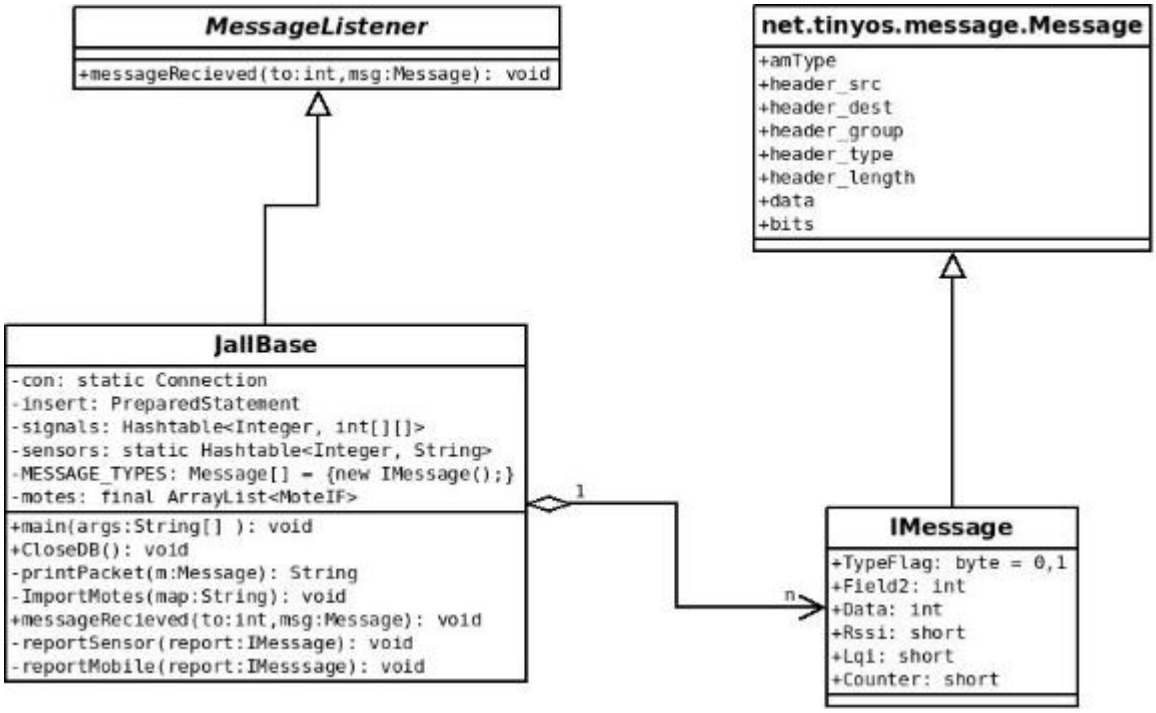
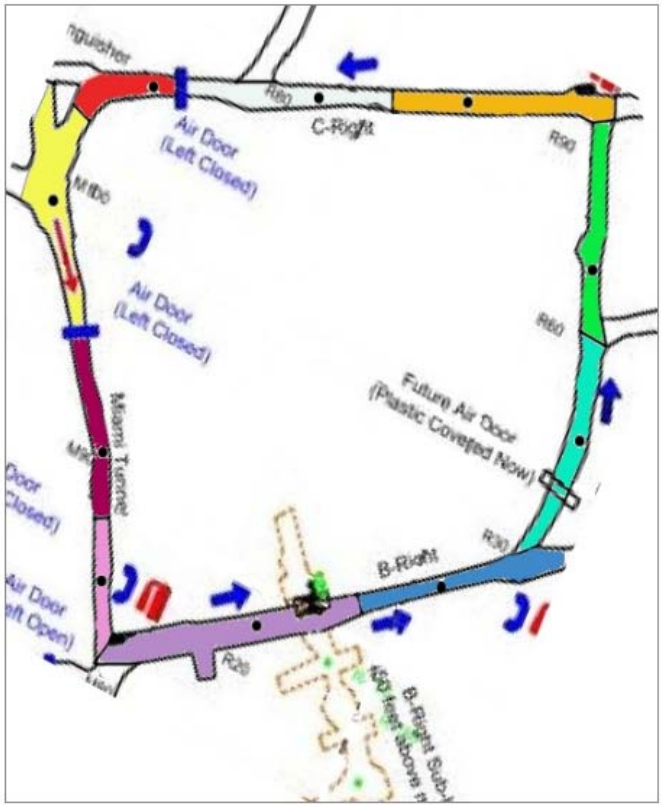


Figure 7. Website Layout



IMote	Light	Humidity	Temperature	Date (Y-M-D H:M:S)
1	100 Lux	73.48 %	63.66 F	2010-06-22 15:47:21
3	100 Lux	71.11 %	62.49 F	2010-06-22 7:56:39
4	200 Lux	73.44 %	62.87 F	2010-06-22 7:56:39
7	400 Lux	70.64 %	63.79 F	2010-06-22 7:45:12

MMote	Sector	Signal	Counter	Date(Y-M-D H:M:S)
4	4	200	9663	2010-06-22 7:51:24
1	1	209	4455	2010-06-23 10:43:19
2	2	201	7734	2010-06-23 10:43:19
3	3	311	5678	2010-06-23 10:43:19
5	5	205	7574	2010-06-23 10:43:19
6	6	200	5768	2010-06-23 10:43:19
7	7	200	5345	2010-06-23 10:43:19
8	8	206	5749	2010-06-23 10:43:19
9	9	209	8794	2010-06-23 10:43:19
10	10	310	7384	2010-06-23 10:43:19

Figure 8