Sign Tracking Device

Final Document

Team: DEC1620
Customer: Flagger Pros
Advisor: Nathan Neihart
Tristan Walters | Team Lead
Brandon Trent | Secondary Team Lead
David Dalo | Key Concept Holder
David Carlson | Communication Lead
Alex Sundholm | Secondary Communication Lead
Tyler Dahle | Webmaster

Team e-mail: dec1620@iastate.edu

Revised Project Design

Project Statement

This project is about developing a tracking device that can be inconspicuously attached to a traffic sign, and can communicate information about its location to a server via long range networks. The project also warrants the development of a web application that can communicate with the server to pinpoint where each tracking device is located and display relevant information to the user.

Project Purpose

The main purpose of the project is to locate lost or stolen road construction signs. This project could increase safety in construction zones because there would be appropriate signage to make drivers aware of workers. This tracker will also reduce replacement costs of road signs for our customer. Our customer is losing two to three road signs a month, each at the cost of approximately $300/sign. This tracker could save thousands each year by allowing them to locate these missing signs.

Project Goals

- Customer satisfaction
- Having a device that is small enough to remain hidden
- A device that can withstand harsh weather
- A device that is battery powered and can last for approximately 9 months
- Implement a server and a database that can store location information and be able to receive and send that information to the web application
- Implement a web application that will integrate with the Google API to show a map where the trackers are located based on the information received from the server.

Deliverables

- The Device
    - Microcontroller
    - GPS chip
    - Cellular chip
    - Battery
- The Database
- The Web Application

Current Design

Functional Requirements

Web Application

- While the user is not logged in, the application shall display a login prompt.
- When the user provides valid login credentials, the application shall create a new session and enable access to the main displays.
- When the user clicks logout, the application shall clear the session and return to the login prompt.
- When the user opens the main screen the application shall display all of the jobsites and unassociated flags on the map.
- When the user clicks on a specific jobsite, the application will zoom into the jobsite displaying all of the tracking devices associated with that jobsite, and display a pop up window with specific information pertaining to that jobsite.
- When the user clicks on a specific tracking device the application will display that device's ID, battery life, the last time the device updated it's location, the current time, and a short warning message in an info window pop-up to help the user realize what is wrong.
- When the user clicks on "Add Job Site" the appropriate menu for adding a Jobsite will be displayed.
- When the user clicks on "View Jobsites" all of the jobsites will display a pop up with the name of the jobsite.
- When the user clicks the "View Jobsites" pop up the application will zoom the user into the jobsite that was clicked.
- When the user clicks on "Remove Job Site" the job site will be removed and the devices that were associated with that job site will be moved to the no association category.
- When the user clicks on the "Manage Account" button the user will be directed to the account management page.
- The account management page will display fields for the user to fill out in order to add a new user to the system.
- When the user clicks on the "View Map" button the user will be directed back to the map page.
- When the user click on the "List Accounts" button the application will display a searchable list of all of the users in the system that are not the currently logged in user.
- When the user clicks on one of the accounts in the list there will be a pop up menu asking if the user wants to delete or edit the account that they clicked on.
- When the user clicks on the "Edit Account" button a form will pop up for the user to fill out to edit the selected account.
- When the user clicks on the "Delete Account" button the clicked on account will be deleted from the system.
- The application must allow only one user to be logged in per application interface.
- The application must allow login with a valid encrypted username and password.

Hardware

- The tracking device shall draw less than 15µA when in sleep mode.
- The tracking device must report its location at least three times a week to database server
- The tracking device shall enter sleep mode and power down all peripherals after successfully transmitting its location.
- All components shall use 3.3V logic.
- The tracking device shall use raw GPS and voltage data to send to the database server along with the device's identifier.

Non-Functional Requirements

Web Application

- The application must be mobile friendly.
- The application shall always run using a secure connection.
- The application must finish the login procedures within 20 seconds.
- The application must finish the logout procedures within 20 seconds.
- The application will use Bootstrap and CSS to allow the application to smoothly transition between devices.
- The application must abide by the laws and regulations of applicable jurisdictions.

Hardware

- The tracking device shall use TCP to communicate reliably with the servers.
- The charge of the battery must be able to withstand usage of up to 9 months.
- The battery itself must be able to withstand harsh environments.
- The location reported shall be accurate within 5 meters.

Interface Specifications

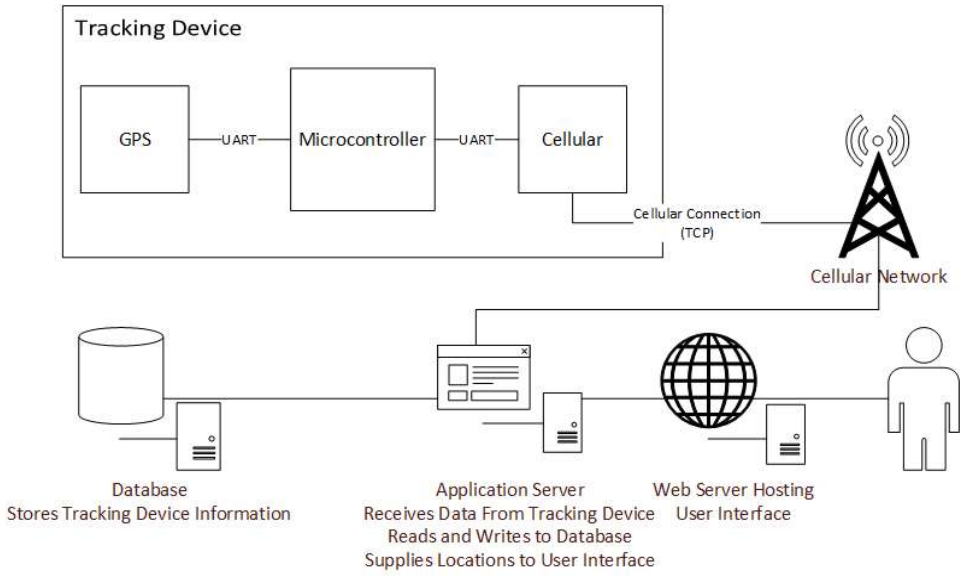

Figure 1: Overall Design View

As shown in the diagram above, our tracking device will be using UART (Asynchronous Serial Communication) to interface with all of its different components. The tracking device will then interface with the application server via the cellular network. The application server will be running background code that will convert the data received from the tracking device, then store it on the server database. The application server will also be running any php code that we would use to have the web server pull data from the server database. The web server will be where the web application will be hosted.

Results

The Web Application

The web application is able to log users in and out of the system. The application can check the credentials of the users and hide functionality that certain users are not allowed to access. The application can display information that is pulled from the database to display jobsites and tracking devices. The application is able to add and remove users from the database as well as edit user's accounts.

The Hardware

The current iteration of our device has the ability to: gather gps data, transmit gps data via cell network, track IMEI, and interrupt out of sleep mode periodically. The hardware has the capability of utilizing a function for battery life, however the code was never

produced. The circuit board has the capability of being expanded to accommodate any design changes that the client chooses to implement, as well as accommodate other power sources.

Implementation Details

Tools

Tool	Reason
phpMyAdmin	Setting up and managing SQL database
Code Composer Studio 6.2.0	Coding and Debugging on MSP430
Chrome and Firefox Debugging Tools	To test the web application
TI Launch Pad	Includes hardware Debugger for MSP430
Hologram Dash	Developing cellular communication

Table 1: Tools Used for Development

Development Process

The process that we followed to create this product is a loose version of Agile. We had weekly team meetings to report on the progress that we made over the past week as well as to come up with new goals for the next week.

Web Application

In order to access the application, the user must first login by providing valid credentials. Once the user is logged into the system, they will be presented with the main screen of the application.

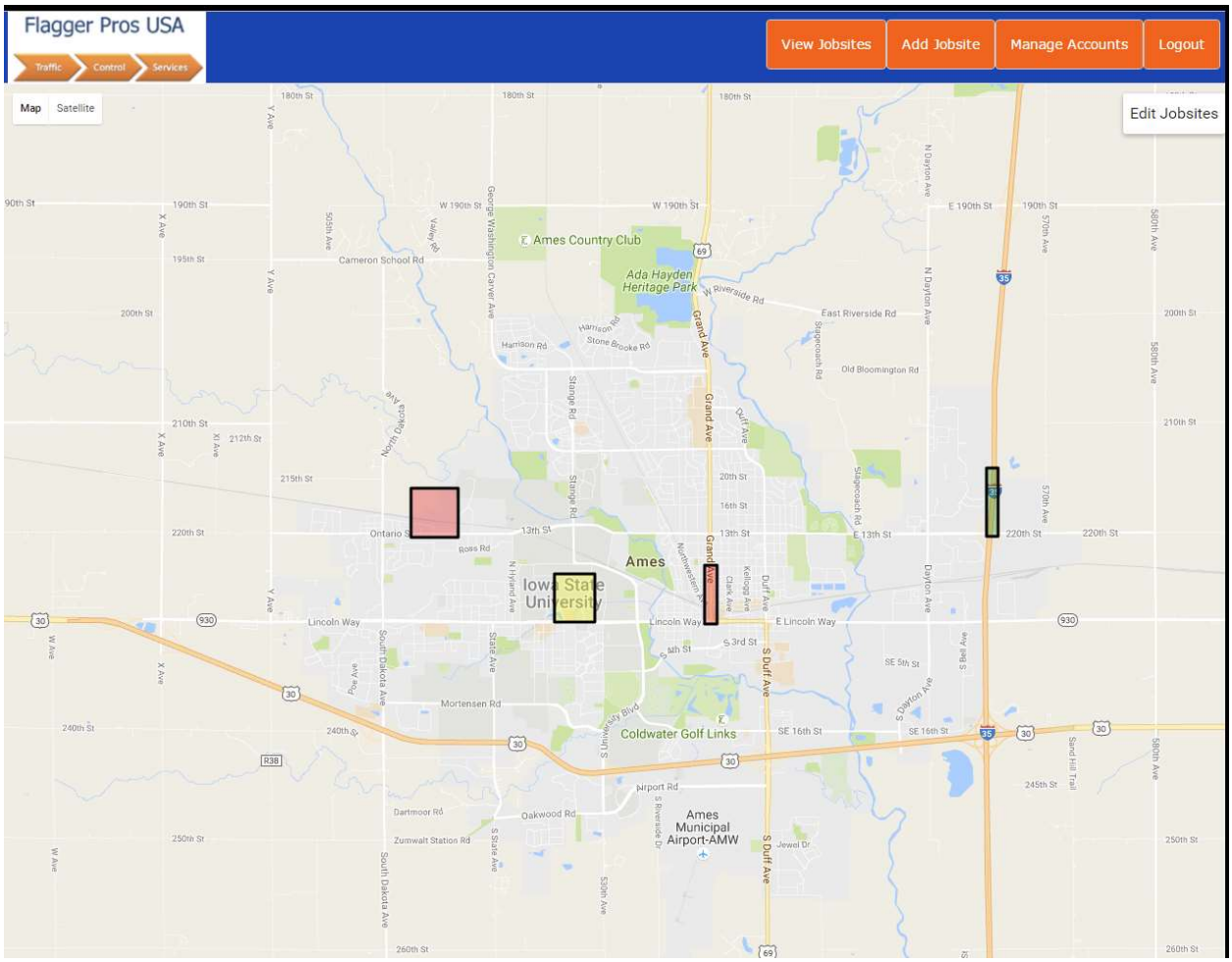

Figure 2: Web Application Main View

On this screen, the user will be presented with four buttons in the header. One of these buttons is there to add a new jobsite. Clicking on this button will bring up a pop up menu that will give the options of setting up a new jobsite. Another button is the “View Jobsites” button. When the user clicks on this button, all of the jobsites will pop up an info bubble displaying the name of each jobsite so that it is easier for the user to see and interact with each jobsite. Clicking on one of the jobsites will zoom in to that particular jobsite’s location and display that jobsite’s information in an info window. The user can also delete a jobsite from the info window if they are an admin. Clicking on the name of the jobsite inside the info bubble that pops up when the user hits the “View Jobsites” button, they will be zoomed into the jobsite and recentered at the center of the jobsite, just like if they clicked on the jobsite.

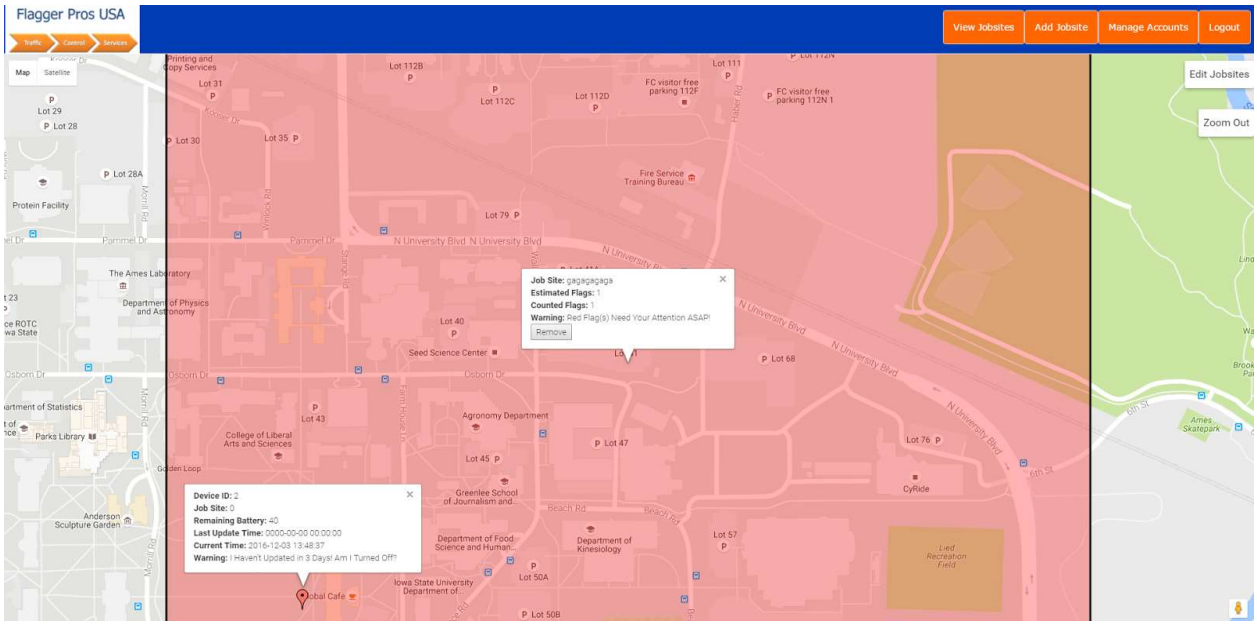

Figure 3: Web Application Jobsite View

The flag markers that are inside a jobsite will be hidden until the user clicks on the jobsite and is zoomed into it. Once they are visible, the user can click on the marker to bring up information about that marker (ID, jobsite, battery, last update time, health, etc).

Database

This project will have three different main tables in our database. They are DeviceInfo, Sites, and UserInfo. The DeviceInfo table will hold the information for each device. Each device will be associated with a jobsite. If a device is not within a jobsite, it will be visible from any zoom level on the map and have no association. The UserInfo table will hold the information for valid users. This table will control the login functionality of the application.

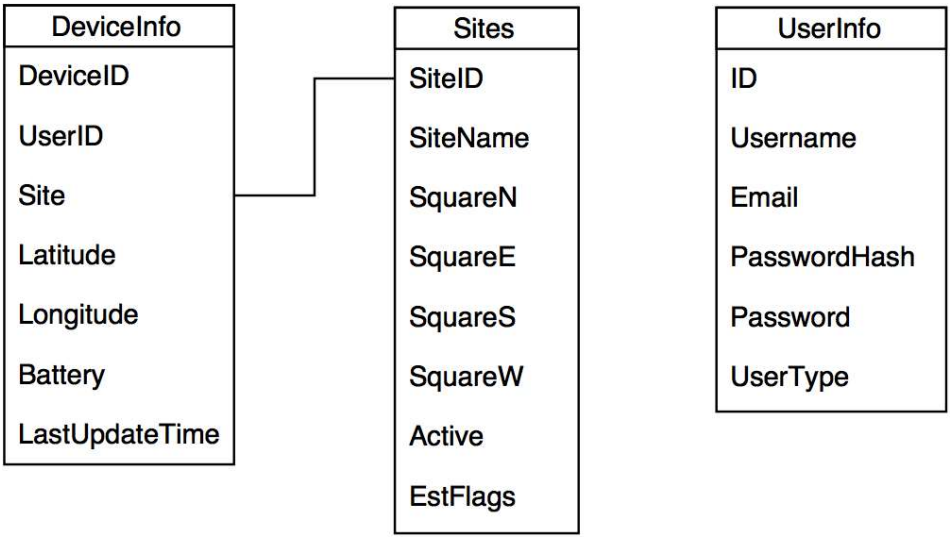

Figure 4: Database Design

Hardware

The circuit diagram on the next page outlines the parts used and how they were connected in our final product. Each part came with headers installed and we created a backplane to mate nicely with each part, making them swappable for testing. This also leaves a fairly clean looking final product. The code that is running on the MSP430FR2433 when first powered on turns the GPS on and watches the UART for usable data. Upon reading in a good location, it saves the string to a variable and cuts power to the GPS. The next step turns the cellular modem on and watches the UART for the modem's unique IMEI that is stored to a variable. Once the IMEI has been read in and saved, the packet of data that will be sent is put together and sent out on the UART to the cellular modem. Finally, the code watches for a success message from the cellular modem and then cuts power before entering low power mode 3.5 (LPM3.5), also known as sleep. It sleeps for two days before waking up and resetting to run the code again. In addition to the functions above, we implemented timeout functions for both the GPS and cellular modem to prevent wasting power in cases where they cannot complete their job due to environmental conditions.

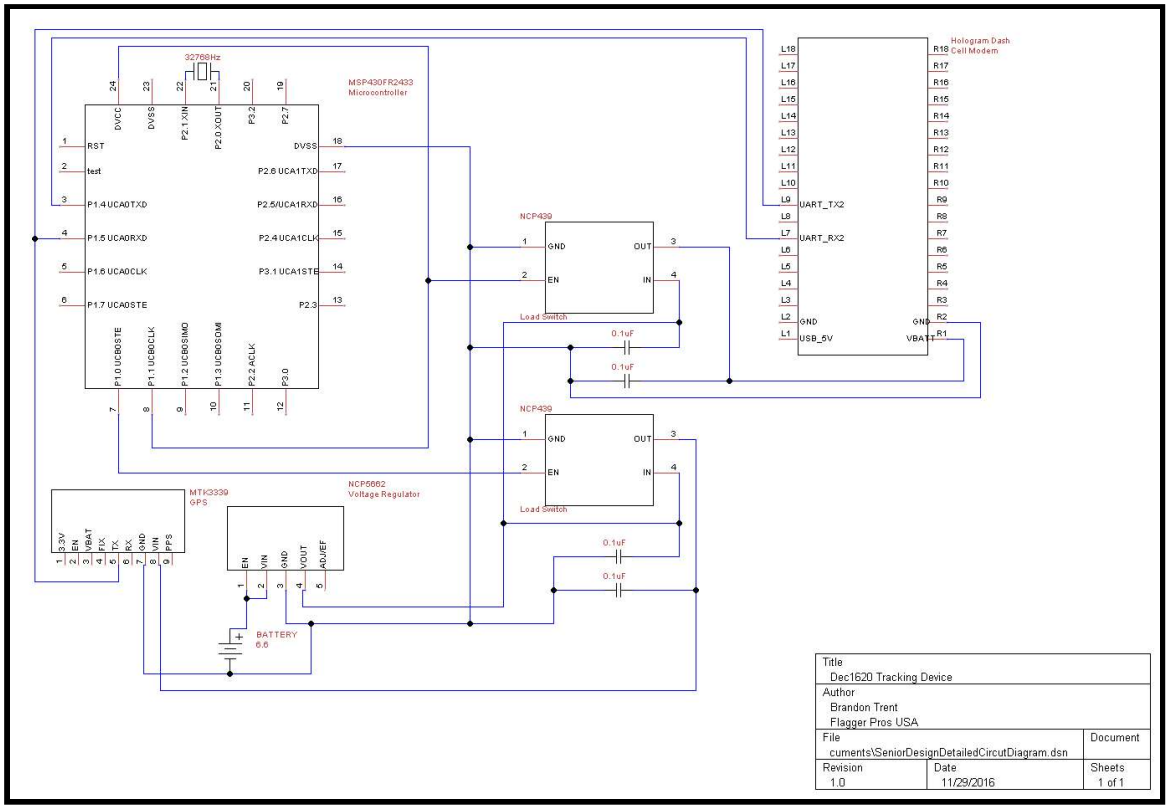

Figure 5: Circuit Diagram

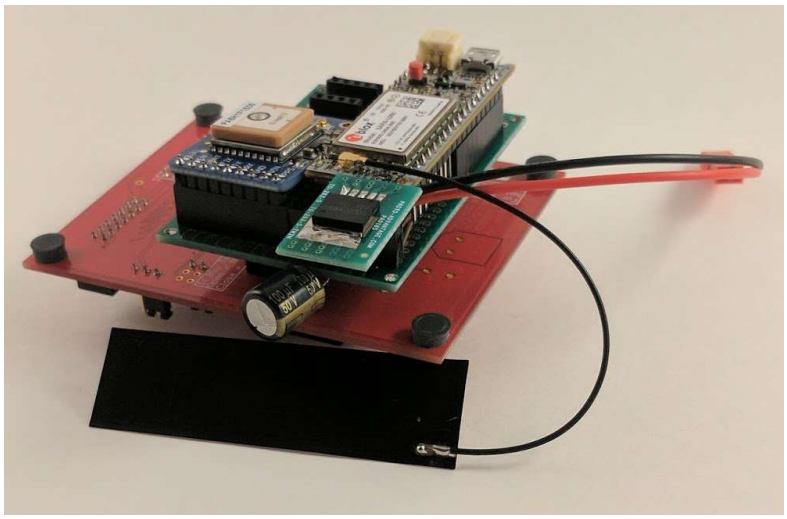

Figure 6a: Final Product

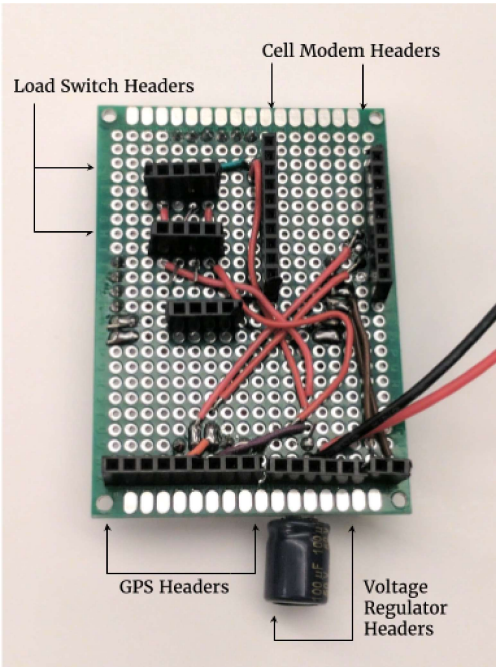

Figure 6b: Header Locations

Testing Process and Testing Results

Web Application

To test the web application we have implemented a test as you go methodology. Since we are using a loose version of Agile development, once we finished a specific functionality we would test that functionality on multiple browsers and multiple devices to make sure that the functionality works anywhere the client could use it. Along with a test as you go methodology, we also gave the url to our client to get their feedback on our current design and also to let them try to break it and find some use cases that we did not think of during development.

Hardware

The hardware had to be tested for reliability and power consumption. In order to test reliability, we ran the device in three locations simulating the various conditions the device might be placed in related to GPS and cellular signal availability. The three locations included the senior design lab that has no view of the sky and marginal cell phone reception, an apartment on the top floor to simulate an open warehouse, and finally open sky in town. The cellular modem performed well in all cases, as it works in any place AT&T service is offered. The GPS did not find a location inside the senior design lab, and after a two and a half minute timeout, moved on to tell our servers we could not find our location. In the Apartment setting, the GPS was able to acquire location in an average of two minutes. Finally, in an open sky environment the GPS found location in an average of 40 seconds. The results of our reliability tests were as we expected when we purchased the parts, so we were fortunate to have no surprises here.

The power consumption test was performed in the senior design lab setting to calculate our worst case power consumption for each run cycle. This resulted in the GPS running for two and a half minutes and the cellular modem running on average of four minutes. The device as a whole pulled 28mA during the GPS portion of run time. For the first 17 seconds and five seconds when data was transmitting the cellular modem pulled, on average, 100mA and the rest of the run time pulled 38mA. In total for worst case the device runs for six and a half minutes and consumes 4.0875mAh of power each run cycle and in sleep mode for one hour we consume 0.0008mAh. Since we run every other day or on average 3.5 times a week we consume 14.44065mAh of power from the battery in a worst case scenario. From this calculation with the 1.5Ah battery we selected we will use ⅔ of the battery in 69 weeks almost doubling the 9 month requirement we were given. Note these calculations do not account for leakage from the voltage regulator or environmental conditions which is why we did not increase the frequency of checking in.

Appendix I: Operation Manual

Web Application

When a user visits the web application page, if they are not logged in they will be presented with a login screen. Enter user credentials and log in. After clicking the login button, the user will be taken to a new page with a header that holds useful buttons we will discuss in a bit, and the google map (which takes up most of the page).

On the map, any flags that are working and have reported to the database and are not contained within a jobsite will be visible right away. The color of the flag indicates their "health." If you click on a flag marker, you can see the warning message to get more specific details as what is wrong with the flag if it is not green.

Flag Health Levels:
Green: Flag has over 40% battery, and last updated within 3 days.
Yellow: Flag battery is between 30% and 40%, but has updated within 3 days.
Red: Flag battery is below 30%, or it has not updated within 3 days, indicating a problem with the device (it was turned off or it died).

If the user has jobsites on the map, they will be rectangles that also are color coded the same way as flags to indicate their health. They count the number of flags inside, and if the number is correct, it is colored as the worst colored flag. If more or less flags are counted than expected, it is automatically red. Again, the jobsite will give you a warning message to indicate a specific problem. If you click on a jobsite, you will zoom into the jobsite and will be able to see all flags inside the jobsite. A popup will also be displayed after clicking the jobsite that will give you information like its name, estimated flags, counted flags, and a message about its health. There will also be a remove button if the user is an admin and has admin privileges. Clicking remove will refresh the page, and the jobsite will no longer be there. If there were any flags inside, they will now be unassociated with any jobsite and be visible at all zoom levels of the map to warn the user there are unassociated flags floating around. If you zoom back out, all flags within the jobsite are hidden from view to keep the map clean from clutter and keep flags organized.

In the header, the user will find four buttons: View Jobsites, Create Jobsite, Manage Accounts, and Log Out.

View Jobsites: Clicking this button at any zoom level on the map will pop up an info bubble over each jobsite. This info bubble will be colored as the health of the jobsite, providing the user with a quick view of all their jobsites and their health to pinpoint problem sites quickly. The info bubble will also contain the jobsite name, if the user clicks the name, they will be zoomed into the jobsite. This helps zooming into small jobsites that are hard to click when zoomed out on the map. There are close boxes on the upper left of the info bubbles that will close the individual info bubbles. Clicking this button again after the bubbles are open will close them all.

Create Jobsite: Clicking this button will bring up a small pop up that will allow the user to input data into two fields. The desired jobsite name and the estimated number of flags that will be sent to the jobsite. Once the user enters these, they click create and they will notice their cursor turns into a cross on the map. The user will then click and drag over the area they want the jobsite to be associated with. Once they release, the jobsite will be created and the page will refresh and the new jobsite will be in its place.

Manage Accounts: Clicking this button will take the user to the page where they can create a new account, edit a pre-existing account, or delete a pre-existing account if the currently signed in user has admin credentials. The first form that the user will see on this page will be the form that allows the user to create a new user. In order to create a new user the user must fill out all of the information. If the user doesn't fill out the information correctly the user will not be able to create the account and will be given a reason as to what they did wrong. If the user wants to edit or delete a pre-existing account the user must click on the "List Accounts" button. Clicking this button will display all of the users in the system that are not the currently logged in user. The currently logged in user must then click the user in the list of users that they would like to edit or delete. Doing this will bring up a popup window with the option to edit or delete a user. If the user clicks delete then they user that they clicked on will be deleted from the system. If the user clicks edit, another popup menu will come up with the form used to edit user accounts. In this form the only information that they user needs to fill out is the username of the user that they wish to edit, all of the other fields are optional. So if the field is filled out the database will update the information but if the field is empty then no change is made to the database. Clicking on the "View Map" button will take the user back to the map page. Clicking on the "Logout" button will log the user out of the system. Clicking on the "Create New User" button will take the user back to the create new user form.

Logout: Clicking this button will log the user out and take them back to the login page.

Database

To set up the database, the database UI software phpMyAdmin is needed. This is the easiest GUI to set up the database quickly and efficiently with the proper configuration. Using either the phpMyAdmin UI interface buttons or the SQL queries provided, create the database as shown in Figure 4.

In order to use the application for the first time, a dummy admin user will have to be created. The credentials for this user need to be username: admin, password: admin, and user type: 0. This creates a user with full admin capabilities, however, it isn't a secure account, so our suggestion is to use this account to create a new admin user and delete the old one. The new admin account will have a hashed and salted password so it will be much more secure.

Every time the database is moved, both of these steps will have to be redone. In the case that the database gets wiped but the tables are still in tact, a new basic admin will have to be created in order to create more users.

Hardware

To program the microcontroller, open Code Composer Studio and connect the MCU board via the Texas Instruments JTAG into the JTAG port on the development board. Make sure that the VCC jumper beside the JTAG port is connected in the correct orientation (internally powered vs. externally powered). To program, ‘int’ should be connected to ‘Vcc.’ Inside CCS, load the program needed and build the project using the Build command (Ctrl+B) and run the program by going to the Run menu and selecting Free Run. This ensures that the JTAG is not involved in the running of the code, which can interfere with the real time clock. The JTAG can now be disconnected from the board. To run, move the jumper to connect ‘ext’ and ‘Vcc’ and connect an external power source.

The device can be dismantled if necessary. Reference the assembled photo for general locations of each device. The GPS module fits into the headers on the left side of the PCB. The load switches, which are interchangeable, slot in above the GPS module, with the text in the same orientation as that of the GPS. The cell modem should be oriented so the USB port should be at the top of the PCB, and the battery wires at the bottom. There is only one orientation for the PCB to fit into the microcontroller.

Appendix II: Alternative/Initial Versions of Project

Web Application

The earliest iteration and design of the website didn't include the jobsites. So it was just going to be a map full of flags with no organization at all. The flags were still going to be colored based on what state they were in, whether it be green for good, yellow for needs maintenance soon, or red for needs attention now. There would be many flags, however, and they would be scattered around without a clear idea of what jobsite they were belonging to, or if they were outside of a jobsite. That is why we came up with the idea to have both a jobsite and a warehouse that these flags could be inside. This idea was to help make the map less cluttered and more readable. We then decided to get rid of the warehouses because they were just the same as the jobsites and there was really no functional difference to them, so we decided to just make them jobsites that you could name warehouse. The different iterations of the database followed this as well. We first had an iteration with no jobsites and warehouses then we had one with jobsites and warehouses and then the final one is just with jobsites. We also considered making this a mobile application as well as a web application. We were only able to get the web application done, but have it mobile friendly. This gives our client easy access from the office, or on the job with their mobile phone despite it not being a mobile application. Our main target browser for development was Google Chrome, as that was commonly used by our client, however we tested and made sure it worked on Firefox and Safari as well. We knew our client and some of his employees had iPhones that may be using Safari instead of Chrome, so we made sure it was functional for them on their mobile device. Also instead of a side menu, which we originally discussed, to house the list of jobsites and flags for each jobsite, we went with simple pop ups over each jobsite to indicate their health and allow easy navigation into the jobsite to see which flag is the problem. This prevented a sidebar from shrinking the viewable map further on smaller screens like mobile phones.

Hardware

The first iteration of our product was very piecemeal, with each individual component passing its output to the next manually rather than integrated into one piece of code. Rather than using our final MCU (MSP430FR2433), we worked with the MSP430G2553 Development Board. The differences between the two changed our product drastically. The FR2433 added Real-Time Clock functionality, a new low-power mode, and increased stability.

Appendix III: Other Considerations

The hardware side of the device had many considerations during the design process. For the microcontroller we really only had one series of processors that could be an optimal choice. More specifically, we needed a processor which could handle being asleep for multiple days at a time. The TI MSP ultralow power line of processors has a specific feature which allows the microcontroller to be fully powered off and draw a minimal amount of power, because of this function and ultralow power consumption we chose the flavor of processor which had just enough memory to contain our program. The next main consideration was other power sources for the device. There are many different compositions of batteries that we could have used for the device such as NiCad, lead-acid, and other Lithium based compositions. We went with LiFePo4 (Lithium Iron Phosphate) to satisfy our temperature tolerance, as well as longevity. The actual size of the battery could have been in a custom form, but instead we chose a single cell case in order to reduce overall cost. The load switch we chose to go with was the most optimal switch for power consumption, leakage, and size for the current we would be running at. Included in the design documents is an optimal voltage regulator, but unfortunately due to production lead times the regulator wouldn't have been ready in time for development. Instead we chose a generic replacement that was not optimized for current leakage for development. Both the gps module, and cellular module were chosen for their cost effectiveness. Other modules were multiple times the price of the ones we chose and offered very little to no benefit.