UNIVERSITY OF CALGARY

ENGO 500: GIS & Land Tenure 2

Project Proposal

By: Jeremy Steward
Kathleen Ang
Ben Trodd
Harshini Nanduri
Alexandra Cummins

Supervisor: Dr. Steve Liang

DEPARTMENT OF GEOMATICS ENGINEERING

SCHULICH SCHOOL OF ENGINEERING

09/27/2013

Table of Contents

Introduction....2
Background Information....2
Internet of Things....2
Open Geospatial Consortium....3
OGC RESTful API....3
Objectives....4
Methods....6
Sensors....6
Computer Board....6
Other Sensors Considered....7
Interfacing with the Database....8
Programming Language(s)....8
Organization / Management....8
Timeline....9
Conclusion....11
References....12

Index of Figures

Figure 1: Gantt Chart of Project Milestones and Tasks....10

Index of Tables

Table 1: Specifications for Arduino Uno, and Models A and B of the Raspberry Pi. Information taken from [11,12,13]....6

Introduction

The revolution of the Internet changed the future of communication forever. As time went by, new technologies were developed which allowed us to move from the First Generation of the Internet to the Fourth Generation. In the 1970s the First Generation of Internet began through APRAnet (Advanced Research Projects Agency Network). In the 1990s the Second Generation of Internet was accompanied by AOL. In the 2000s, the third generation began and we are still in it today. This generation is dominated by social media, such as Facebook, Twitter, MySpace, Flickr, etc. Society is now in the process of transitioning from the Third to Fourth Generation. The Fourth generation of Internet is represented by the Internet of Things (IoT) [1].

IoT is a new concept that involves the networking of all physical devices to function more cohesively as one unit to assist in everyday life. It connects everything to the internet “virtually”. IoT will have a huge impact on businesses all over the world as well as the average household. Some advantages of IoT include: increase in household security, increase in energy conservation and also an increase in business sales [1]. However, the potential for such a complex network of connections also leads to many different implementations. Because of this, it is desirable to have an accepted standard for application development.

The Open Geospatial Consortium (OGC) is an international consortium with more than 445 companies, research organizations, government agencies, and universities that participate in a consensus process to develop geospatial standards that are available publicly. The OGC Standards allow technology developers to make geospatial information and services useful with any application that needs to be geospatially enabled [2]. According to Dr. Steve Liang, Sensor Web for IoT convener and director of GeoSensorWeb Laboratory at the University of Calgary, "The standards coming out of this OGC process will make it possible for developers to ensure that sensors, the observations they produce and the systems they inform will be easy to reach and control with Web services, without compromising security and data integrity” [2].

This project involves using sensors in order to develop an Internet of Things application using the Open Geospatial Consortium (OGC) standard. The project is designed within the scope of ENGO 500, Geomatics Engineering Project. Our aim for this course is to work on a design project as a group in order to increase and apply our knowledge to a real-world problem, which will prepare us to work as professional engineers in the future.

There are a few goals that we wish to fulfill by the end of the project. The first goal is to design and plan a project, and complete it successfully by working as a team. Our second goal is to produce an open-source web-service application that uses IoT to solve problems that are relevant to the population in Calgary. The third goal is to learn more about the OGC standards for IoT, and to develop a prototype that uses the specifications that were outlined by the OGC committee. Our final goal is to improve our project management skills by using Github. Github provides a way to keep track of tasks assigned to each individual and all the documentation/reports required to successfully complete this project.

Background Information

Internet of Things

One of the primary aims of this project is to develop an application of the IoT. The term ‘Internet of

Things' (IoT) refers to the network formed between tangible objects and their virtual counterparts, and is the result of linking these two representations via an internet like structure. This is usually done by equipping or embedding objects with sensors, providing them with a unique identifier and the ability to communicate without requiring human interaction [3].

Objects used in an IoT may be anything from household appliances to living creatures. In each of these cases, data from the sensor assigned to an object is transferred over a network and may be accessed in real time, in order to gain insight on temperature, noise levels or other measurable conditions of the object being observed.

Open Geospatial Consortium

The Open Geospatial Consortium (OGC) is a non-profit international organization, and exists as a medium though which more than 400 organizations worldwide may collaborate to encourage development and implementation of open standards for geospatial services, data sharing and geographic information system (GIS) data processing [4]. Founded in 1994, the main focus of the OGC is the creation of a set of technical documents known as OGC Standards. These documents detail encodings, which ensure complex spatial information remains available and relevant to various applications. OGC Standards are developed by a unique consensus process, which ultimately allow geoprocessing technologies to interoperate, thus guaranteeing compatibility between two or more separate products [5].

OGC RESTful API

There are two different types of web services, the first being Simple Object Access Protocol (SOAP) based services, and the second being Representational State Transfer (REST) based services. The primary difference between the two types of services is that SOAP-based services transfer data in the form of XML documents through a specified schema, while REST-based systems transfer data by dynamically generating web pages (e.g. through some Common Gateway Interface (CGI) based application) [6]. An Application Programming Interface (API) exists to aid developers in writing applications by providing them with an abstraction layer that specifies how software components such as data or processes should request services from the underlying hardware or software layers [7]. In the context of web services such as SOAP or a RESTful API, as we intend to use in this project, it usually consists of a set of classes that can format and make requests from the web service, and thus simplifies the use of the service [7]. Effectively this provides us a library or a suite of functions and objects with which to interact with the web service.

This particular project will likely use multiple devices or sensors. While the developers of these sensors or devices may have provided a unique API for each sensor, this will likely bring difficulties when integrating more than one sensor or data set. Therefore, there is a need for a single, standards-compliant API, that can access and interact with any sensor we may choose to use. Linking everything through a common API will facilitate the project in a number of ways: it will centralize and ease the connection process to and from the sensors (this includes reducing connection and execution time of the application, as well as necessary power), it will simplify the code (thereby reducing the number of various languages and models required for developers to learn in order to use any specific sensor), and finally it will ease the task of maintaining security for one web service rather than several. Finally, it will be easier for us as the developers to implement one API rather than several over the duration of the project.

To implement our project, we are planning to use the RESTful Web API design model, or RESTful

web service. This is an architectural-style web API, implemented using HTTP and REST principles. A RESTful API service requires several necessary aspects, including the media type of the data supported by the web API, and a set of operations (such as GET, PUT, or DELETE) using HTTP methods as mentioned above [8]. In addition, these aspects should include some base Uniform Resource Identifier (URI) string for the Web API, as we want to be able to identify our data as we are accessing it [9].

Objectives

The overall goal of this project is to use the Internet of Things (IoT) to tangibly assist the citizens of the city of Calgary in a practical and user-friendly way. In so doing, the end product should utilize the OGC RESTful API and should also be open source. In order to accomplish this goal, several smaller objectives and sub-objectives have been set out. The intention of these objectives is to set out clear, achievable tasks which will lead to a useful application and also fulfill the requirements of this course. However, because these are being outlined at a very early stage in the project, it is anticipated that some adjustments will occur as the project progresses. The following list describes each of the main objectives and their subsequent sub-objectives.

1. Determine a location-based application of the IoT which is relevant to the city of Calgary. This requires defining a problem relevant to Calgarians (or a particular sub-group of citizens, such as civil servants, university students, environmentalists, etc.), specifically identifying a component which can be measured and the importance of its spatial properties. The scope of the application should be outlined, including what area it will cover and what value it will bring. In order to continue with the project, the application should be selected by October 4, 2013.

2. Design a sensor setup which can measure data suitable for the application determined in objective 1.

    a) Determine the technical requirements necessary for data measurement, such as the sensor(s) required, and other periphery elements needed. These requirements should be written into a technical specification report with an accompanying sketch of the sensor setup.

    b) Create a working prototype of a sensor which could be deployed in the city. The prototype’s technical specifications should follow the technical requirements detailed in sub-objective 2a.

    c) Test the prototype in a lab setting, in order to understand functionality and limitations. Simulated environments should be used so that adjustments can be made before testing in a field setting. All test data should be saved and stored in a secure format, and should be accompanied with detailed experimental notes (such as dates of testing, differences in conditions, etc.)

3. Develop a library which links the sensor setup and the OGC RESTful API.

    a) Learn necessary language(s) required for writing a library which will work with each sensor in the chosen sensor setup. Proficiency is needed to a point where it will be possible to write a working library; as such, learning should have a concentrated focus.

    b) Collect set of test data with multiple sensors in a working/usable configuration in the

city. The configuration may be sparser than actual deployment, but should be a reasonable (e.g. at least 5 sensors) network. All test data should be recorded and stored in a secure format. Experimental notes should be kept, including details such as date(s) of experimentation and times, pictures of location, weather conditions, etc.

4. Create a user interface (UI) such as a web site or mobile application which makes use of the OGC RESTful API as provided by Dr. Liang's research group to display the sensor data.

    a) Design a user-friendly graphical interface which is tested and found intuitive by at least 80% of test subjects. Testing will be based on hands-on experience and will be evaluated by means of a standardized survey. Feedback collected will be used to improve the final version of the user interface.

    b) Represent collected data (in real time) using the created user interface and create a video of its usage. The video should show both how to use the developed UI, as well as prove its functionality.

5. Fulfill all deliverable obligations as explained in the ENGO 500 course outline punctually and professionally. These deliverables will also serve as one means for tangible documentation and as a measure of progress.

    a) Write a project proposal which outlines background information, project objectives, methods and expected outcomes by September 27, 2013. This proposal should include some initial research into the topic, as well as planned objectives and methods for meeting those objectives. A Gantt chart showing the planned breakdown of tasks should also be included.

    b) Complete a literature review which thoroughly explores the history of the topic, current state-of-the-art and applications. This should be completed by November 18, 2013.

    c) Present a report of technical deliverables in two forms: written and oral. The written report component should be accomplished by December 6, 2013 and the oral component should be presented at an appointed time between November 22 and December 6, 2013. This report should cover the technical deliverables which are to be accomplished in the previous four objectives.

    d) Give a progress report in two formats: written and oral. The written report should be completed by February 14, 2014 and the oral report should be given at an assigned time between January 31 and February 14, 2014. The progress report will communicate work done towards the final goal.

    e) Create a final project report in two formats: written and oral. The first draft of the written final report will be completed by April 4, 2014. An oral report, given as a defense of the written report, will take place at an appointed time between April 7-11, 2014. Based on feedback from the oral defense, a revised copy of the final report will be finished by April 28, 2014. The final report represents the culmination of all the project work throughout the school year.

    f) Present at the Capstone design fair. The date has yet to be announced, but the expected date should fall somewhere around the end of March or April. This will be used to showcase the project's final products.

g) Upload and copy documentation as well as all necessary work onto the group's Github repository [10]. This repository will serve as a means of management for the group, and is also fulfills the goal of being open source. This will serve to assist any future work that may be performed using this project as a base.

Methods

Sensors

Computer Board

All sensors will be interfaced using a single board computer. These computers are well suited for do-it-yourself (DIY), educational, and prototyping projects due to their low cost and the wide variety of available sensors. The boards considered for the project are the Raspberry Pi and the Arduino Uno. Although they are similar in size and cost, their intended purposes differ slightly. The Raspberry Pi is a fully operational computer running a Linux operating system [11]. Having a full OS gives the Raspberry Pi the luxury of communicating with higher level devices such as a USB WiFi dongle or sensors with digital output. Conversely, the Arduino Uno is a micro-controller designed for embedded applications, meaning there is no operating system [12]. The Arduino is well suited to communicating with low level sensors which give either analog or digital signals. Table 1 below highlights their differences:

Table 1: Specifications for Arduino Uno, and Models A and B of the Raspberry Pi. Information taken from [11,12,13]

Name	Arduino Uno	Raspberry Pi Model A	Raspberry Pi Model B
Cost (USD)	$29.95	Discontinued	$35.00
Size	2.5”x2.10”	3.37”x2.125”	3.37”x2.125”
Processor	ATMega 328 @ 16MHz	ARM11 @ 750MHz	ARM11 @ 750MHz
SRAM	2 KB	256 MB (Shared w GPU)	512 MB (Shared w GPU)
Flash Memory	32 KB	SD CARD	SD CARD
Ethernet	None	None	Ethernet
WiFi	Adapter available	3rd party adapter	3rd party adapter
Digital GP I/O	14	8	8
Analog Input	6	None	None
I²C	2	1	1
SPI	1	1	1
USB	None	1	2
Video Out	None	HDMI, Composite RCA	HDMI, Composite RCA
Audio Out	None	HDMI, 3.5 mm jack Analog	HDMI, 3.5 mm jack Analog
Input Voltage	7-12v	5v	5v
Power Requirements	42 mA	300 mA	700 mA

Below we have listed some of the advantages of each of the boards.

Raspberry Pi:

- Internet connectivity: The Raspberry Pi includes a full GNU/Linux OS and therefore includes advanced networking capabilities. It provides an Ethernet port as well as plug and

play support for some USB Wifi & 3G dongles.

- Performance: While the Raspberry Pi does have the overhead of running an OS, it boasts a 750 MHz processor which can be overclocked up to 1GHz with either 256 MB or 512 MB (shared with GPU) of RAM available. The flash memory is also larger (depending on the size of SD card used).

- Cost: While the board itself is comparable to the Arduino, being able to use a standard USB WiFi dongle is a great advantage over the Arduino WiFi shield (~$10 vs $70).

- Output: If the sensor needs to be controlled in-situ, common video and audio output formats will help accomplish this.

- Programming Flexibility: With a full GNU/Linux OS, any language that compiles on ARMv6 architecture can be used. This opens up the possibility of using many languages or a mix of languages.

Arduino Uno:

- Power requirements: If the application requires batteries as the power source, the Arduino uses significantly less power than the Raspberry Pi.

- Sensor compatibility: The Arduino offers more ports to interface with sensors outputting both digital and analog signals. Comparatively the Raspberry Pi cannot communicate with analog output sensors.

- Large development community: The Arduino development community is more mature than the Raspberry Pi's with a large number of tutorial and example projects [13].

While other computer boards may be considered depending on the application we decide upon, many are comparable to these two boards, and as such only these boards will be explored for the remainder of this report.

Other Sensors Considered

For the location of the device, the sensor set-up will likely be one of two situations:

- If the hardware was designed to be used in an application where the sensor is moving, a GPS receiver will likely be included. Adding a GPS receiver will increase project cost by means of the receiver as well as its power supply.

- If the sensor is designed to gather spatial data (i.e. image based) from a discrete location, it could be possible to avoid including a GPS receiver if the location of the sensors does not have to be known to a high precision.

The other sensors to be included on the device depend on the application chosen. There are a wide variety of sensors available, many of which are inexpensive. Examples of other sensors that could be include are: image, video, sound, temperature, pressure, humidity, altitude, distance, magnetometer, accelerometer, force sensitive resistor, tilt, infrared, luminosity, motion, radiation, liquid flow, stretch and more [14].

Interfacing with the Database

The sensor assembly will have network access of some kind, with which it will send sensor data to a database designed specifically for IoT applications. This server is maintained by the Sensor Web Interface for IoT Standard working group [15]. The sensor assembly will require a library or set of libraries to interface with the database's OGC RESTful API, which will be developed to adhere to the upcoming OGC Standard Sensor Web interface for IOT (SWIOT) [16] standard. This project will likely be one of the first to use the SWIOT standard, and it is hoped that the project helps the standard get recognized and used by other developers.

Programming Language(s)

The Raspberry Pi is configured to support Python by default, but it is easy to use other languages such as C/C++ or anything else that will compile on ARMv6 architecture. Advantages of using Python are that as high level programming language it allows for easy development and smaller source size. Due to its interpreted nature, it is slower and uses more resources than C++. Advantages of using C/C++ are that as an intermediate programming language it is faster and more flexible than Python. Development of C++ programs is more complex, which takes more time and code is typically 5-10x larger in C++ [17].

The Arduino offers its own integrated development environment (IDE) where the device can be programmed using the Arduino Programming Language, based on C/C++. It also has many libraries available by default for connecting to standard sensors [18].

Organization / Management

The project will be managed through Github which will serve as a means to plan & document the project during its evolution. Github has many tools to aid projects along their development and is a good choice for the project due to its open source nature. The Github tools that will be used throughout the project's lifetime are:

- **Milestones:** For any deliverables required or goals set during the project, a milestone will be created. A milestone can be given a due date, and will include one or more tasks that need to be accomplished.

- **Issues:** Any time there is a task that needs doing, an Issue will be created for it. The issue's description will outline background information, objectives, and can be associated with a milestone. Issues can be assigned to users or users can volunteer by assigning an issue to themselves. Once an issue is assigned to a user, other user's know not to work on it without addressing the issue’s owner beforehand. Issues can be labeled and sorted in various ways to help manage the backlog, and will serve as the main organizational system for our day to day work.

- **Wiki:** The Github wiki will serve as the main repository for all the knowledge amassed throughout the project. It is readable by anyone but only project collaborators (our group) can add to it for the time being. By compiling all data in one place all team members can stay current with the project's evolution without needing to consult multiple sources.

- **Pulling/Forking:** Github uses the content tracking system git to track changes to files (usually code), often by means of *forking* and *pulling*. A user can fork (or copy) a repository

(collection of files) and then subsequently create a local copy on their machine. From here they can make any edits they desire, and if they want to commit the changes, they create a pull request. This pull request is reviewed by other teammates, specifically the group leader, and if accepted then these changes are merged into the master branch of the main repository. The group will likely only work on the master branch for the duration of the project, but if necessary other branches might be developed and explored.

By utilizing these tools the group hopes to manage the project in an open, detailed, and organized manner in which all group members are accountable and progress is maintained.

Timeline

See the Gantt chart in Figure 1 on the next page to see the task breakdown and expected timeline for milestones and objectives throughout the project. Note that the project runs from week 0 (week of September 13th, 2013) until week 32 (week of April 25th, 2013).

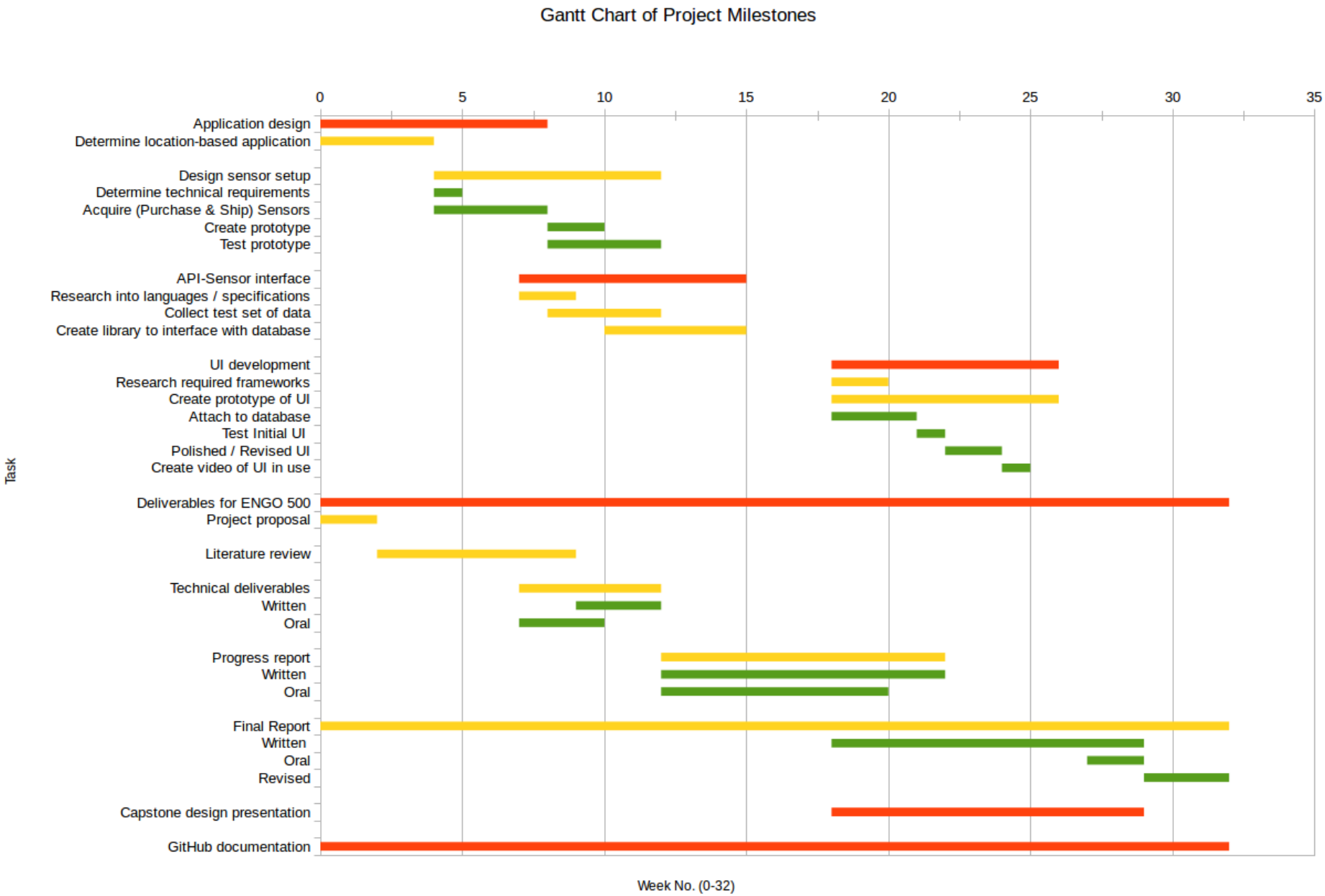

Figure 1: Gantt Chart of Project Milestones and Tasks

Conclusion

The desired outcomes of this project are to:

1. Design and plan a project, and to carry the project through to completion working in a team environment [19].

2. Produce an open source web-service application (web site or mobile application) that uses the Internet of Things (IoT) to solve a problem relevant to us locally (i.e. in Calgary). By open sourcing this application, we hope to provide some meaningful benefit to either Calgarians or any other entity that may wish to take over the project upon its completion at the end of the year.

3. Learn more about the Open Geospatial Consortium (OGC) standards for IoT, and develop a prototype application that uses the upcoming standard specification as outlined by the OGC committee. This application, which should match our second outcome, should ideally serve as an example for those who wish to use or implement the specification in the future, once it has been fully published.

4. Learn about project management skills, in particular managing projects using Github.

Ideally, by meeting these outcomes, we will fulfill a greater outcome, that is, to have worked on an independent project as a group to further our own knowledge and prepare ourselves to work as professional engineers. Likewise, while many smaller outcomes may be fulfilled, such as learning more about programming, sensor integration through micro-controller boards, and others, these are merely subsets of the main outcomes listed above.

In order to realise these outcomes, we will work over the course of 32 weeks (both fall and winter semesters) on developing a web site or mobile application that integrates sensor data using the OGC IoT standard. To accomplish this, we will first determine a relevant problem to us locally, and develop a sensor network of some fashion to use as a means to solve this problem. We will then use data from the sensors and develop an API layer to communicate with the OGC standards-compliant database that Dr. Liang has provided. Finally, we will use the database as the basis to develop our web site or mobile application, as listed in outcome 2, which should either manipulate or provide information from the sensors that targets our problem.

In the process of completing all of this, we will seek to improve our project management skills and documentation abilities by keeping track of the project through Github and completing the required reports for the course. By working under constrained deadlines, and by providing documentation as we develop the project, we will ultimately be working closer towards becoming professional engineers ourselves.

References

[1] Suny Cortland, "The Internet of Things," [Online]. Available: https://sites.google.com/a/cortland.edu/the-internet-of-things/home. [Accessed 22 September 2013].

[2] "The OGC Forms "Sensor Web for IoT" Standards Working Group," [Online]. Available: http://www.opengeospatial.org/node/1650. [Accessed 22 September 2013].

[3] I. Wigmore, "Internet Of Things (IoT)," July 2013. [Online]. Available: http://whatis.techtarget.com/definition/Internet-of-Things. [Accessed 23 September 2013].

[4] "About OGC," Open Geospatial Consortium, 1994. [Online]. Available: http://www.opengeospatial.org/ogc. [Accessed 20 September 2013].

[5] "OGC Standards and Supporting Documents," Open Geospatial Consortium, 1994. [Online]. Available: http://www.opengeospatial.org/standards/. [Accessed 20 September 2013].

[6] B. Spies, “Web Services Part 1: SOAP vs. REST,” May 2013. [Online]. Available: http://ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest. [Accessed 24 September 2013].

[7] D. Orenstein, “Application Programming Interface – Computerworld,” January 2000. [Online] Available: https://www.computerworld.com/s/article/43487/Application_Programming_Interface?pageNumber=1. [Accessed 24 September 2013].

[8] W3C, “HTTP – Hyper Text Transfer Protocol Overview,” July 2013. [Online] Available: http://www.w3.org/Protocols/. [Accessed 24 September 2013].

[9] W3C, “URIs, URLs, and URNs: Clarifications and Recommendations 1.0,” September 2001. [Online] Available: http://www.w3.org/TR/uri-clarification/. [Accessed 24 September 2013].

[10] J. Steward, “ThatGeoGuy/ENGO500,” [Online]. Available: https://github.com/ThatGeoGuy/ENGO500. [Accessed 24 September 2013].

[11] “FAQs | Raspberry Pi,” [Online]. Available: http://www.raspberrypi.org/faqs. [Accessed 22 September 2013].

[12] “Arduino – Arduino Board Uno,” [Online]. Available: http://arduino.cc/en/Main/arduinoBoardUno. [Accessed 22 September 2013].

[13] “Arduino vs. BeagleBone vs. Raspberry Pi | MAKE,” April 2013. [Online]. Available: http://makezine.com/2013/04/15/arduino-uno-vs-beaglebone-vs-raspberry-pi/. [Accessed 22 September 2013].

[14] “Sensors / Parts, Adafruit Industries,” [Online]. Available: http://www.adafruit.com/category/35. [Accessed 23 September 2013].

[15] “Sensor Web Interface for IoT SWG,” Open Geospatial Consortium, 2013. [Online]. Available: http://www.opengeospatial.org/projects/groups/sweiotswg. [Accessed 22 September 2013].

[16] “Internet of Things (IoT) and Web of Things (WoT),” Open Geospatial Consortium Network. [Online]. Available: http://www.ogcnetwork.net/IoT. [Accessed 22 September 2013].

[17] G. van Rossum, “Comparing Python to Other Languages,” 1997. [Online]. Available: http://www.python.org/doc/essays/comparisons.html. [Accessed 22 September 2013]

[18] “Arduino – Reference,” [Online]. Available: http://arduino.cc/en/Reference/HomePage. [Accessed 23 September 2013].

[19] B. Teskey, “ENGO 500 Course Outline,” September 2013. Available through Blackboard.