Team 14: Sense-Able Gym
Final Design Report

Calvin College ENGR 339/340 Senior Design Project

Daniel DeHoog, TJ DeVries, Paul Griffioen, Ryan Siekman

Wednesday, May 11, 2016

Funded in part by the Eric DeGroot Engineering Fund

Abstract

Team 14 is a group of senior electrical and computer engineers at Calvin College. The group is composed of Daniel DeHoog, TJ DeVries, Paul Griffioen, and Ryan Siekman. This group was brought together to create a senior design project for Calvin College's engineering capstone course, Engineering 339 & 340.

The Sense-Able Gym is a project that provides gyms with the ability to turn their equipment into Internet of Things (IoT) machines. The goal of this upgrade in technology is to answer a common problem for people who attend the gym. Oftentimes, when going to the gym, a person finds him or herself waiting in line for a machine that he or she would like to use. With a combination of sensors and displays (implemented using TI SensorTags), along with a single "smart hub" (referred to as the hub and implemented using a Raspberry Pi), the Sense-Able Gym gathers current use data from machines and sends that data to the gym clients' standard or mobile platforms. In addition, clients are able to reserve specific machines over the media of their choice. Through these two possible actions, gym clients have the ability to reduce their wait time by checking in online or checking the busyness of the gym without waiting in line.

The Sense-Able Gym also offers administrators the ability to track usage of machines by time, by peak volume, and by frequency to make executive decisions about their gym in a way that they were unable to before. A gym administrator may now see that the peak usage for a particular type of machine is much less than the number of machines supplied in the gym.

Contents

1 Table of Figures

2 List of Tables

3 Introduction

The goal of this project is to create a system that will allow people, both clients and administrators, to interact with gyms in a modern and smart way. This system allows users to view what machines are currently open, and it also allows users to reserve systems for personal use. The system also provides gym administrators with the ability to understand more clearly what machines get used, along with how often and when they are used.

In order to accomplish this, sensors are placed on the gym equipment to detect whether or not it is in use. Displays are also placed on the equipment to show current or future reservations. These devices connect through a mesh network which communicates the information to a hub that then sends the data to server. A mesh network is a network where each device connects to those closest to it, and each device passes on information from one device to the next until the information makes it to the final location. It is through the server that reservations can be made on a hosted website and data for the gym can be utilized.

This project is being done as the Senior Design Project for the Calvin College Engineering Department by team 14, which consists of Daniel DeHoog, Paul Griffioen, TJ DeVries, and Ryan Siekman. Senior Design is the capstone course for the Engineering degree at Calvin College and consists of a project chosen by the team, through which students learn about the development of task specifications, design process, team building, and design norms, where all of these are learned within a reformed Christian worldview. All the members of team 14 are Electrical/Computer Engineering Majors with Computer Science minors that will graduate in spring 2016.

4 Project Management

Senior Design by nature has a distributed management style, with the individual teams being self-managed for the most part. The structure which this self-management takes within the teams can vary. In team 14 it has developed into an even split with all members contributing to different parts of the project and being independently responsible for their work.

4.1 Team Organization

Team 14 consists of Daniel DeHoog, Paul Griffioen, TJ DeVries, and Ryan Siekman. The work on the project last semester was primarily doing research, and in that aspect the work was been split up primarily by the different components of the project. Daniel did the research and led the decisions concerning the displays. Paul was in charge of the sensor research. TJ worked on the server and has made the overall system diagrams for how all of these components will communicate and work together. Ryan did the research concerning the hub component. Beyond these team roles, TJ managed the schedule worksheet, Paul wrote up the submission for the Eric DeGroot Engineering fund proposal, and Ryan maintained meeting minutes and action items. All of the members of the team have been in communication with various people outside the direct team, through emails to the professors or others with whom the team has worked. These tasks have been assigned more on a basis of who the task is most relevant to, rather than having a single person do all of the email communication. Mr. Eric Walstra is the team's Industrial Mentor and Professor Michmerhuizen is the Team Advisor as well as a Course Instructor for Senior Design. Team 14's organization structure can be seen in **Figure 1**, which includes the other Course Instructors along with a Junior Engineer, Taylor Mulder, who was part of the Business 357 team.

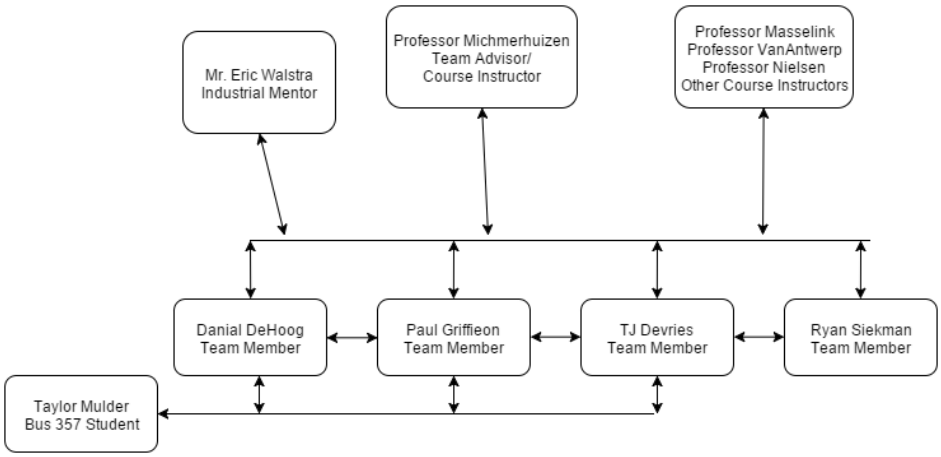

Figure 1: Organization Chart of those Involved in the Project

The work in the spring semester of Senior Design consisted of mainly coding the software for the system to work and communicate together. The sections of the code were split up and written by a similar breakdown in components, with a different team member responsible for the software for each component. Since much of the communication needs to be coordinated, there was overlap as interpretation or encoding for the information needs to be uniform across the system. TJ wrote and worked on the server software, Dan was responsible for the hub software, Paul wrote the sensor code, and Ryan worked on the display code.

Team meetings occurred every Monday afternoon from 3:30pm to 4:30pm after senior design class for the first semester. In the second semester more time was needed for meetings, so they moved to Tuesday mornings from 9:00am to 11:20am. Meetings were run by first covering what the previous week's action items were and discussing how they were completed or if they would continue to be an action item for the following week. Then any items that were due soon were addressed, and lastly all the future action items were assigned. This last action was usually done while discussing what the next steps of the project needed to be and the correct order and importance of them by referring to the schedule.

Documents related to the senior design project are all kept in either GitHub or Google Drive [20]. Any documents or work internal to the team is on Google Drive, while all of the documents and reports that will be turned in and all of the code written thus far for the project are kept in GitHub. The reason for keeping all of the reports and documents that will be distributed outside the team in GitHub is because these are all being written in LaTeX. This provides more flexibility in the formatting and combining of documents written by the individuals on the team, along with version control to see what has been changed and when. The shared Google Drive folder contains the team's meeting minutes, schedule, research notes, presentation, and budget. A link to share the content of the team's Google Drive folder is in the reference section, which was cited when initially mentioned in this paragraph.

4.2 Schedule

Initially in trying to find a scheduling system that would work well, various options were tested, including Asana, calendar integration with Slack, Trello, and Microsoft Project. The last one, Microsoft Project, is what the initial schedule was created with. However due to its lack of online accessibility, it was exported and saved in the Google Drive folder as a spreadsheet. From there TJ continued to add functionality to it in order for the team to be able to track hours and enable better visualization of tasks and due dates. The schedule was updated with events during the team meetings or as they were assigned. Each team member

edited the schedule as they spent more hours on tasks or completed them. The schedule determined what action items were assigned and what parts of the project were most critical to work on at any given time. When schedule issues arose, it was usually a matter of spending more time on tasks, delaying a due date, or deciding that one item must precede another. These issues were dealt with as they arose, and there were virtually no issues with task scheduling. However, one issue was ensuring that the schedule worked, as the functions within the spreadsheet that allow the hour tracking and views were difficult to maintain when items were added, moved around, or viewed on other sheets.

4.3 Budget

The budget was maintained through a spreadsheet in the team's shared Google Drive folder. It was maintained by Ryan and was updated multiple times, as the team made multiple equipment orders. It was necessary to update the budget spreadsheet as orders were made, or information on different pricing was received. The budget was not used as a management tool as it did not constrict the ability to make a prototype of the system. It only limited the size and variety of possible prototypes that could be tested.

The plan is to test the system at Calvin's gym, and the budget is large enough to monitor at least two pieces of gym equipment. Any increase in the budget would allow the team to increase the scale of this testing or to test with different components than initially selected. The team obtained an increase in budget from the Eric DeGroot Engineering Fund, enabling us to purchase more sensors than initially planned. If more machines are monitored, more data will be generated, which will help to see the accuracy and the use from an administrative side of the gym. Testing more devices would enable the team would be able to better determine what physical hardware components are more accurate or cheaper and could create a more refined final product with that knowledge.

No budget issues have arisen in the project. However, if they do then decisions will be made concerning what purchases will be more critical to our prototype than others. The primary problem which may arise being between expanding the prototype system or testing a larger variety of hardware for the system, or in the worst case, having to shrink what is currently planned for testing.

4.4 Method of Approach

This project started by trying to determine a useful product, not currently available, that all members of the team would be interested in. This led to several options, most related to the Internet of Things (IoT). The problem at this point became finding a project or potential product idea that was not currently on the market, and would be feasible for the scope of senior design. Paul initially thought of an application to show users what gym equipment is in use, as gym users can get frustrated when a machine they would like to use is constantly in use. The initial idea was to accomplish this with signal analysis of the gym through a camera, using signal processing to determine where people were located and what machines they were using. Due to the large variation in gym sizes and set ups, it was determined that this would be difficult to accomplish generically for various gyms. Therefore, it was decided to implement the system using individual sensors on each machine. This approach will still have an aspect of signal analysis, though it will be done with sensor data as opposed to visual data.

With the increase in IoT technology and the growing number of sensors being used for various tasks, using sensors to determine the equipment use seemed like a more feasible idea. The next stage was what would make this more useful to the user. Being able to not just sense whether the equipment is occupied or not does not completely solve the issue of never having open equipment, as it would then just let the user know it is always occupied. Adding a reservation aspect to the system allows the user to ensure that they will have

time on the desired equipment, and let other people know when it is reserved. In order to show in the gym whether equipment is reserved or not it now became necessary to add some type of display or notification ability to the equipment. It then became necessary to add a hub and server as the back end necessary to support the system.

The Team has utilized Slack for communication within the team. Email is used for any communication with those outside of the team. Slack is very useful and efficient as it allows for group conversations easier than on email and still includes features like document sharing and easy access from mobile devices.

5 Project Justification

Research was conducted to evaluate the uniqueness of the proposed system and to understand what similar gym reservation systems exist. Throughout this research, it became clear that the majority of reservation systems that exist are systems that reserve facilities rather than systems that reserve specific pieces of gym equipment. Most of the equipment reservation systems that exist are used for reserving sports equipment, not gym machines.

However, a few systems exist that serve as reservation systems for specific machines in a gym. Two examples include the gym reservation systems for Syracuse University [1] and Harvard University [2]. Both of these systems use machines built by Precor, a company that manufactures fitness equipment [3]. Precor uses the fitness software provided by Preva [4] to interface with their manufactured machines to provide a machine-specific reservation system for users.

Despite these few examples of gym reservation equipment and software, there are no known companies that supply a gym reservation system that is not already built in to each machine. The gym reservation systems provided by Precor and other companies are always built in to each workout machine. As a result, gym managers must buy this specific type of expensive equipment in order to have a reservation system for the gym. Buying this expensive equipment is often infeasible for a few reasons. If a gym manager is starting a new gym, the price of this expensive equipment is oftentimes too high. In the case of existing gyms, the benefit gained by introducing machinery with reservation capabilities is often outweighed by the expensive price of the new machinery and the fact that all the current machinery would need to be discarded.

The system that was designed in this project is unique in that it provides a gym reservation system that is general and is not built in to any specific piece of equipment. The system has the ability to be installed in any existing gym on however many machines the gym manager desires. In addition, by not being incorporated into any specific machine, this system is expected to be cheaper than the cost increase associated with upgrading from a normal machine to one with reservation system capabilities. As a result, the designed reservation system is unique and original with respect to its modularity and generality.

6 Requirements

The requirements for the project are broken down into requirements relating to the system as a whole, requirements for each piece of hardware, and requirements for the senior design class.

6.1 System Requirements

General system requirements, spanning the system as a whole, are needed to ensure each piece of the system operates correctly and works well with other pieces of the system.

6.1.1 Generic

The designed system shall be able to be installed in a typical existing gym, where each sensor and each display can be installed on (most) any type of stationary equipment. If possible, each sensor should be able to be installed on free weights, weight sets, and other moving equipment. The system shall also be transferable in that the sensors and displays shall be able to be transferred from one type of machine or piece of equipment to another and still operate effectively.

6.1.2 General

The implemented system shall be able to run continuously for 24 hours without crashing, and the price of the system shall be small enough that existing gyms would be willing to invest in buying it. Gyms shall also be able to use only parts of the system if desired, and they shall be able to easily add or remove parts after the initial setup and installation.

6.1.3 Accessibility

The data collected during operation shall be accessible via a web interface (mobile device and personal computer). More detailed data about equipment use shall be available for gym managers, and if possible, fitness data should be available for personal use.

6.1.4 User-Friendly

The sensors implemented shall not impede the use of any machinery or weights, and the web interface and mobile application should be simple and easy to use.

6.1.5 Reservation System

The reservation system shall be organized according to machine, user, and time. Rules should be implemented within the reservation system about making and canceling reservations in order to protect against system abusers. The reservation information for a specific machine should appear on the display for that machine, and the user should be able to interact with the display to edit the reservation. If possible, bulk reservations (over multiple periods of time) should be available in addition to electronic calendar integration.

6.1.6 Real-Time Updates

Real-time updates of which machines are currently being used, where each update occurs within one minute, are necessary, and these updates shall be based on data acquired from the sensors, not data used from the reservation system.

6.1.7 Central Management

A smart hub, located within the gym, shall be able to collect data from the sensors and communicate data to the displays. All of this data will be collated by a central server, which manages the reservation system, the web interface, and the mobile application. If possible, the smart hub will cache daily schedules that can be sent to the displays in case the smart hub loses connection to the internet.

6.2 Hardware Requirements

The requirements for this project are divided into four parts based on the necessary hardware. The four parts of hardware necessary to implement the project include sensors used to determine whether or not a machine is in use, a display interface used to display machine reservations, a central hub placed within the gym to communicate with the sensors and display interfaces, and a server used to communicate with the hub and host the website and mobile application.

6.2.1 Sensors

The sensors used in determining whether or not a machine is in use should be movement sensors (accelerometer, gyroscope, or magnetometer) on a single board. Long battery life, on the order of years, is necessary in addition to the sensors being relatively small (less than 5 in^2). The sensors shall also have network capabilities and be able to form a mesh network, where the sensors can communicate with and pass data to one another. Lastly, the sensors shall be able to make strong yet removable physical connections to each machine.

6.2.2 Display Interfaces

The display interfaces used to display machine reservations shall have long battery life, on the order of months. They shall display information, such as a name and the reservation time, in a user-friendly manner and be able to handle sweat that may fall on the displays. They shall also have network capabilities and be able to form a mesh network, where the displays can communicate with and pass data to one another. In addition, each display shall be compatible with the mesh network, and it should be relatively small (approximately 2 in by 3 in) with at least two push buttons and one LED for making reservations.

6.2.3 Hub

The hub that is placed within the gym shall be a small yet powerful computer, such as a Raspberry Pi. It should be capable of running multiple operating systems (Linux and Windows), and it shall have an Ethernet connection so that it can be reliably connected to the internet. In addition, the hub shall be able to communicate with the mesh network of sensors and displays through some means (Wi-Fi, ZigBee, Bluetooth). If possible, the hub should be able to turn on and off the sensors and displays in order to save energy.

6.2.4 Server

The server shall be a reliable computer that is able to communicate with the hub. It also shall host the website and mobile application that users employ for reservations and seeing which machines are currently in use. In addition, the server shall store the database of information for reservations and data collected from the sensors.

6.3 Class Requirements

The deliverables for the project shall include a complete PPFS, a final report, a working prototype system, a functioning team website, a team poster, a folder of project software, and any necessary presentation materials. Each member of the team shall contribute to completing each of these deliverables according to

the work to which he is assigned. Individual tasks shall be completed on time, and tasks shall be assigned at weekly team meetings.

7 Task Specifications and Schedule

The team met at the beginning of the year to brainstorm what the project should look like and the logical steps of how to accomplish these goals. After creating a list of requirements, considering both goals of the senior design project and deliverables for the senior design class, tasks were created and assigned. These tasks were scheduled throughout the year so as to make sure there would be ample time for revision, handling any problems that arose, and ensuring quality work on all aspects of the project.

7.1 Organization of Tasks

Tasks were organized in a functional manner. As mentioned above, the group considered what the requirements of the project would be, and using those requirements continued to break down the project tasks into smaller and smaller components, until they were easily described and able to be tested if they were done or not. This allowed the team to have a clear outlook on what tasks needed to be done each week, and each team member was able to work effectively because of it.

7.1.1 Fall Semester: PPFS

The PPFS was broken down by section and each section was assigned a due date for a rough draft to be completed. Once all of the sections had been completed, the group met to work through any questions any members had on individual sections. Following this, the team had several revision meetings.

7.1.2 Spring Semester: PPFS

Similar to the Fall Semester, the team looked at what was required for the final report and tried to decide what needed to be fixed or added to the original PPFS that was turned in at the end of the first semester. Once that analysis had been done, the team assigned the pertinent sections to each group member. Finally, the team had several revision meetings.

7.1.3 Fall Semester: Research

The research was split into four different sections, one for each piece of hardware and the respective software for each section.

7.1.4 Fall Semester: Hardware

Paul researched the sensor and mesh network aspects of the project. Dan researched the display aspect of the project. Ryan researched the hub aspect of the project. TJ researched the server aspect of the project, along with contributing to the display and sensor research. After Paul and Dan completed their research, they were tasked with creating a Bill of Materials for what parts should be ordered for the project. Once the research was completed, the group had a meeting to order parts.

7.1.5 Spring Semester: Hardware

The main tasks assigned for hardware had to be assigned throughout the semester. At the beginning of the semester, the team thought that there would be no need for new hardware to be researched or bought during the semester. However, after experiencing difficulties with the SensorTags, the team reactively created tasks to handle the new hardware needs.

7.1.6 Fall Semester: Software

Research for software was performed by each of the members of the team for the respective sections they were given. The majority of the research done regarding the software was done with respect to the server and hub software. Paul and Daniel did the majority of the research for the hub software, including research into different methods to retrieve the data from the sensors and store it in a sensible fashion on the hub. TJ and Ryan did the majority of the research for the server software, including research into a Model-View-Controller Display developed in Python to inform the user about the current status of the machine equipment.

7.1.7 Spring Semester: Software

The software tasks were now about implementation rather than research in the Spring Semester. The team broke up into categories mainly related to previous experience and began work on implementing the features outlined in the requirements section.

7.1.8 Fall Semester: Oral Presentations

Ryan and TJ were tasked with doing the Oral Presentations. Initially they spent some time brainstorming the main goals of the project. Then they created the presentation for Oral Presentation I. They were also tasked with the second Oral Presentation. After examining the progress that had been made thus far in the semester, Ryan and TJ created an updated presentation. This presentation included new information about hardware that had already been ordered by the group, as well an updated scope and more information about the project as a whole.

7.1.9 Spring Semester: Oral Presentations

The whole team presented throughout the Spring Semester. There were several presentations that were made, both in front of the class as well as for Design Night and for industrial consultants. The team met together to work and plan on these presentations and held multiple practice sessions.

7.1.10 Fall Semester: Website

TJ was assigned webmaster by the team. He then was tasked with finding a good template to use. Following that task, he edited the template to meet the needs of the Senior Design team. Then he was tasked with launching the site. Later in the semester, hours were added to refine the website and add more content.

7.1.11 Spring Semester: Website

TJ was still in charge of updating the engineering design website with any new pertinent information about the team.

7.1.12 Fall Semester: Management

Each week, the entire team was tasked with a meeting on Monday afternoons. Otherwise, TJ was tasked with maintaining the schedule. Ryan was tasked with handling any paperwork or billing for project materials.

7.1.13 Spring Semester: Management

Each week the entire team was tasked with new tasks at the meeting on Tuesday mornings.

7.2 Summary of Tasks

NOTE: *THIS SECTION WILL BE UPDATED WITH FINAL HOURS FOR THE FINAL REPORT*

IT IS STILL REPRESENTATIVE OF FIRST SEMESTER

After creating functional tasks based on the requirements of the project, the team was able to do analysis on what would be expected throughout the Fall Semester. A Gantt chart was created to visualize the progress of the project and to view critically linked tasks. This Gantt chart can be found at https://drive.google.com/open?id=0BwaKZBFlZngDS25qNE5iOG1kcE0. This Gantt chart was created using Microsoft project and from the spreadsheet that can be found here: https://drive.google.com/open?id=1fMAsrZ42CqWzHr-vXf1pUO8xREHGZ6jaAe1X21Iv93o.

7.2.1 Expected Hours By Month

However, the team was able to do more than just view progress, but because of the task breakdown, each task was able to have estimated hours attached to it. When these hours were considered as a sum, it was possible to see what the expected hours per month would be.

The estimated hours per week per person was approximately 3.5 hours. The team actually slightly exceeded that at 3.7 hours per week per person.

7.2.2 Actual Progress By Week

While the average progress per week was expected to be around 3.5 hours, it is clear in **Figure 2** that the work was not always very evenly spread out. Even so, in the end the team accomplished all the tasks and completed almost all of them on time. Please reference either the Gantt Chart or the Google Spreadsheet to find a particular task if necessary.

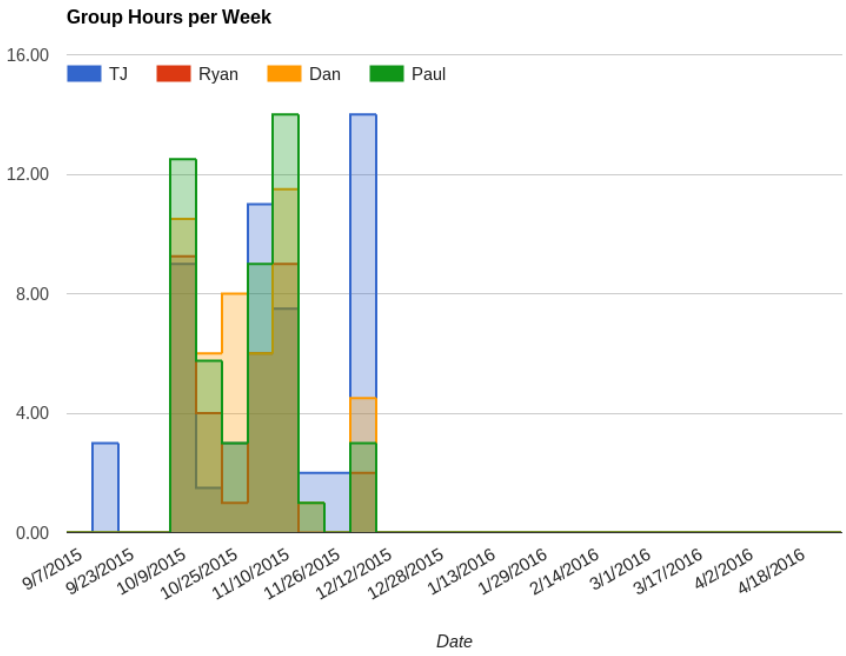

Figure 2: Work Done by Week

8 System Architecture

The system architecture is split into four main categories: Sensors, Displays, Hub, and Server. The overview of the architecture can be seen in **Figure 3**. The system was designed in order to be as low cost as possible, while still retaining generality for many different types of gyms. This architecture attempts to address problems of range, variable amounts of gym equipment and scalability.

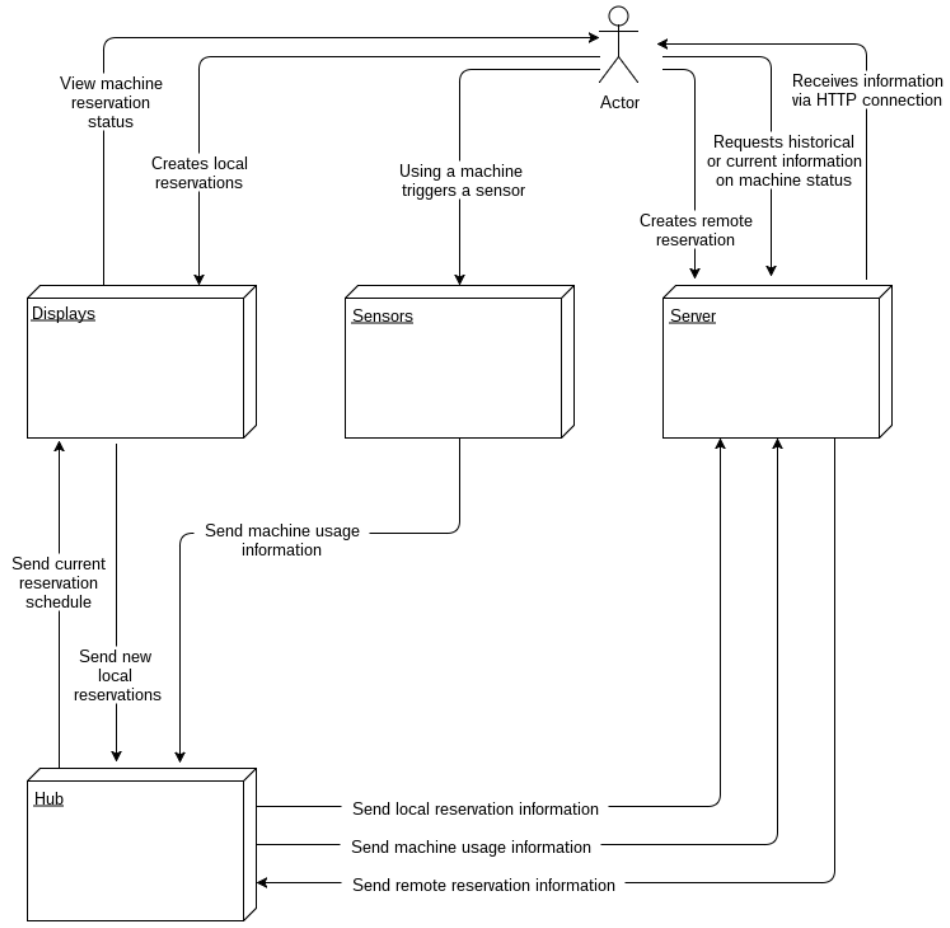

Figure 3: UML Diagram of the Sensor Architecture

The displays and sensors would be placed on individual gym equipment, so that the user would have a by-machine understanding of the reservations upcoming for a given machine. The sensors' main goal is to send information back to the hub, for collation by the server. The goal of the hub is to gather the data from the sensors and the display and send it to the server, as well as distribute remote reservation data to the displays. Finally, the server interacts with the user and sends information as requested to the user. It also handles any new reservation requests.

One way the user interacts with the system via HTTP requests to the server, primarily for requesting information or setting reservations. Another way the user will interact with the system is using the gym equipment. By doing so, they will activate the sensors, who will update the hub. The final way the user interacts with the system is by reading from the displays and finding out the information about reservations for that machine.

8.1 Sensors

To accomplish the goal of having a sensor network with a wide range, but long battery life, it was decided that a mesh network system would be necessary. In general, to make a signal have a longer range, more power is given to a bigger antenna. This was not an option for sensors where the expected battery life must be greater than one year. However, a small range was not acceptable to fulfill the requirement of a

generic system that would fit most any gym, so it was out of the question to settle for a small range. Another option is to use sensors that are capable of communicating to each other and then funneling that information through the other sensors to a centralized hub that would collect this data: a mesh network.

The basic UML diagram in **Figure 4** details the connections between the sensors, between the sensor and the hub, and between the sensor and the user.

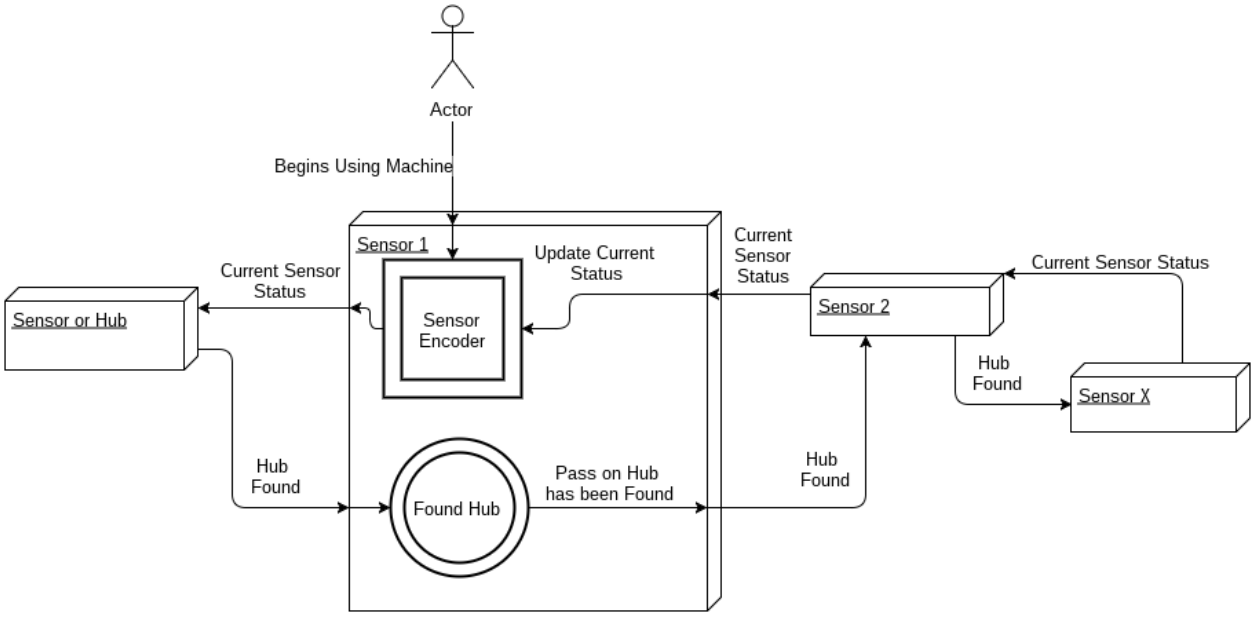

Figure 4: UML Diagram of the Sensor Architecture

The sensor architecture can be thought of as funnel. The further out from the hub a sensor is, the further away it is from the center of the funnel. To communicate with the hub, each sensor will filter its information down to the hub through other, closer, sensors.

To be able to fulfill this design architecture, each sensor will contain two main processes. The first process is *Hub Found*. The *Hub Found* signal will alert other sensors further away from the hub that the destination is "down this way." This will allow sensors, if this attribute is detected, to not continue to search for other devices to make connections to, but to focus on only one (or perhaps two for redundancy's sake) and thus conserve power. The second process is *Sensor Encoder*. This process will encode the information recorded by that sensor can be linked to that sensor. This allows the sensor to pass on useful information about itself, along with being able to send the data from previous sensors. This ensures that the information is kept separate and particular to each machine. This is integral to determining which machines are in use and maintaining realistic reporting for both real-time and historic services. This process is activated by an outside actor using the equipment.

The information contained in *Current Sensor Status* signal is the encoded data that will – eventually – be sent to the hub. After arriving at the hub, that data will be sent to the server for processing.

8.2 Displays

The displays are the answer to the problem of showing the user if a machine is reserved or will be reserved shortly. They also will allow the user to create a reservation locally to ensure that they will not be asked to leave the machine before they are ready to leave.

With the need to learn about remote reservations and push local reservations, the displays encountered a similar problem as the sensors: low power and long distance communication. With that similar problem came a similar solution: displays with mesh networking capability. In the **Figure 5**, one can see that the displays are designed to communicate with each other as well as the hub.

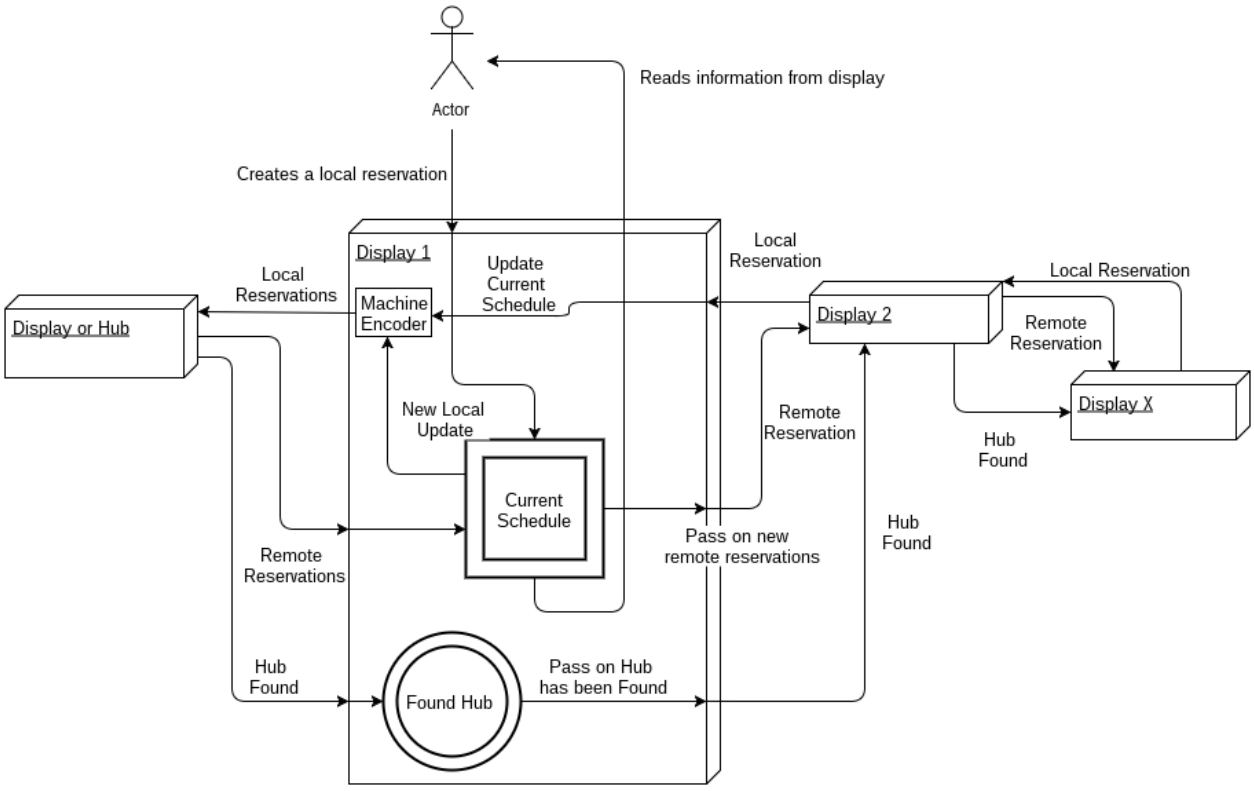

Figure 5: UML Diagram Detailing the Display Architecture

Extending the analogy presented for sensors, the displays will funnel any information about local reservations down to the hub through other display units. They will also pass a *Hub Found* signal up the chain of displays to communicate when searching is not required any longer. However, displays must also retrieve information about remote reservations and thus have an additional connection when compared to the sensors. Due to the three different streams of information, the display contains three distinct entities within itself.

The first entity is the *Hub Found* entity, which completes the same function as the *Hub Found* entity within the sensors. The second entity is the *Current Schedule* entity. This is a basic database that contains all the information regarding the reservation schedule for the current day for this machine. It takes in a remote reservation signal and decides if it is pertinent to this display. If the reservation is pertinent, it updates the *Current Schedule*. If the reservation is not pertinent, it continues sending the encoded information to the next sensor, or "up the funnel." The third entity is the *Machine Encoder*, which will only send information if a local reservation has been initiated by an outside actor. It will also encode any other local reservations made by machines up the funnel and then send that down the funnel. The funnel here being the same type of mesh network that the sensors use, they can be their own network or simply be part the same network as the sensors. This is similar to the encoding of the signal in the sensors to ensure that the data received by the hub can be matched to a specific piece of equipment.

8.3 Hub

Now that all the information from the sensors and the displays has been "funneled" into the hub, the hub's first purpose is to transfer this information up to the server for processing. The other purpose is to transfer any new information to the displays, along with transferring the *Hub Found* signal to the displays and sensors directly connected to the hub. This is illustrated in the **Figure 6**.

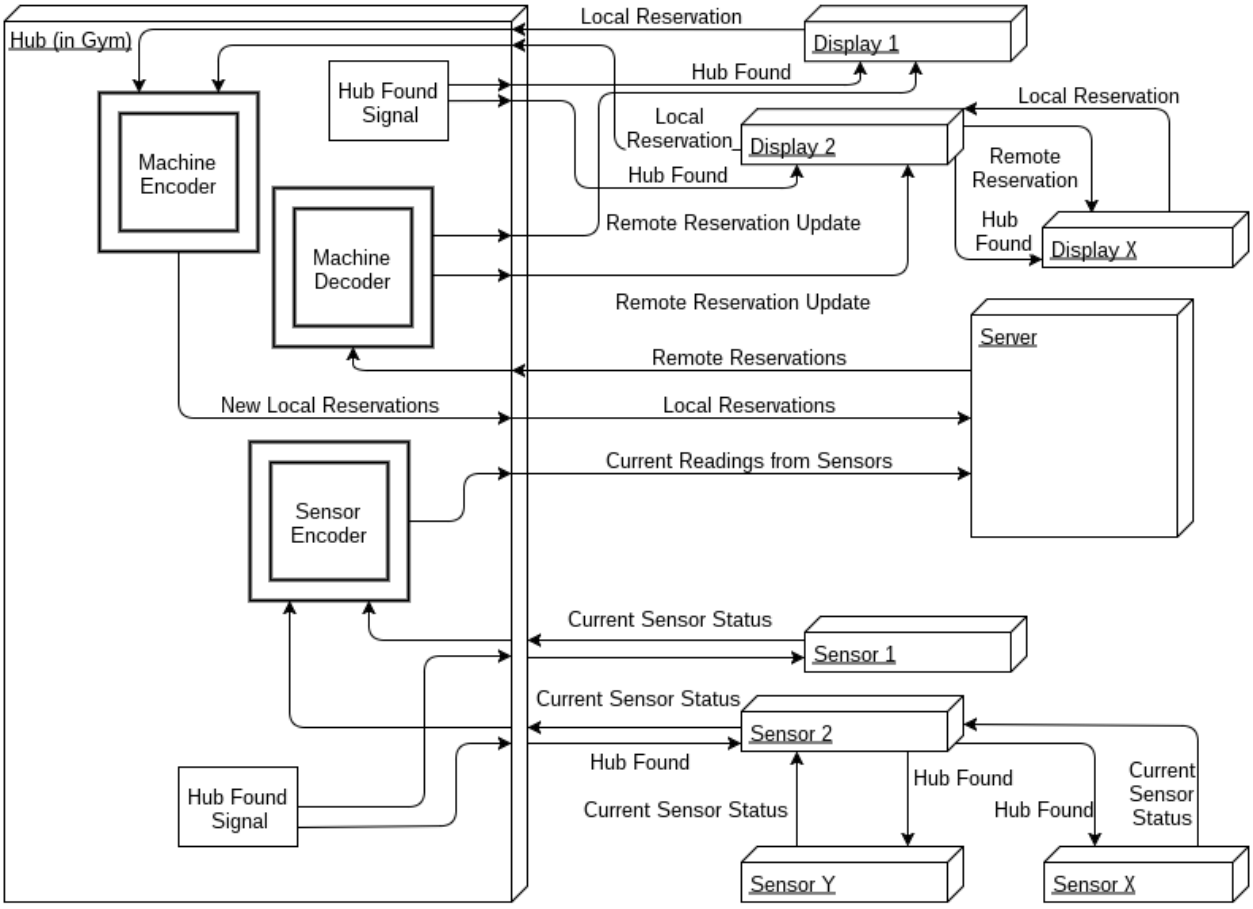

Figure 6: UML Diagram Detailing the Hub Architecture

The hub has several key processes that affect both the sensors and the displays. It must encode all of the signals that it gets from the sensors into something that it is able to send to the server for the processing of the signals. This will most likely include taking the information and forming a packet that is sent to the server. Because it can connect to several different sensors, it must have its own *Sensor Encoder* that encodes which sensors sent which data into the packet that will be sent to the server. The hub will also send a *Hub Found* signal to any connected sensors to ensure the sensors are funneling their readings to the correct location.

The hub also handles all of the connections from the displays. A *Machine Decoder* is necessary in the hub because it disseminates remote reservation information from the server to the displays. The *Machine Decoder* is where the hub will take the information from the server and decode that data into display-specific data. It also receives information from the displays, so it must have a *Machine Encoder* that takes the information received from the displays about local reservations and then passes that along to the server. After the information is encoded and sent, the server can maintain an up-to-date schedule. Once again, the hub sends

a *Hub Found* signal to the displays so that each display can confirm it is sending the data to the correct location.

The hub has three basic types of connections. The first type of connection is when the hub makes a connection to the server which carries information about local reservations made by actors in the gym. The second type of connection is when the sensors create a connection to the hub. On this connection the sensors send information on the current status by sensor. This signal is properly encoded. The third connection is when the hub receives information about remote reservations. This connection provides the information that will later be passed, after decoding, to the displays for up-to-date representation of the current schedules by machine.

8.4 Server

The server's main purpose is to track the information both from sensors and from remote users. It will take the information from the sensors, process the data, and then be able to represent what machines are currently in use. It will pass that information to three primary locations: *Current Reservation Database*, *Historical Use Database*, and *Current Use Database*.

The *Current Reservation Database* will contain recent reservations made both locally and remotely. Additionally, if a machine is currently in use the entity will prevent someone from reserving that piece of equipment for a set amount of time. This prevents a user from being kicked off a machine unexpectedly. Because of this, the database will be updated from HTTP requests (from the World Wide Web actor), local reservation requests (human actor in the gym), and from current status readings (made by sensors in the gym). The database will forward any new reservations to the machines by sending the information to the hub. The *Current Reservation Database* will contain all the reservations that have not yet been completed for the machines. It was designed as a separate database so that sending information to the machines would be simpler and quicker, as a very large (and ever-growing) database would not have to be queried to send up-to-date information about reservations.

The *Historical Use Database* will exclusively track the information by machine that is received from the hub. It will act as the pool of information that administrators can draw upon to gain insight into the use of the machines and that clients will draw upon to gain health and habit insights on. It is planned to be a log of all past-use by machine. It will also contain information about reservations made, including if they were fulfilled, fulfilled on time, and proportion of time the machine was used with and without reservation.

The *Current Use Database* is a simpler version of the Historical Use Database in that it will not log use of the past machines, but it will keep a record of whether a machine is currently in use or not. This system is shown in **Figure 7**.

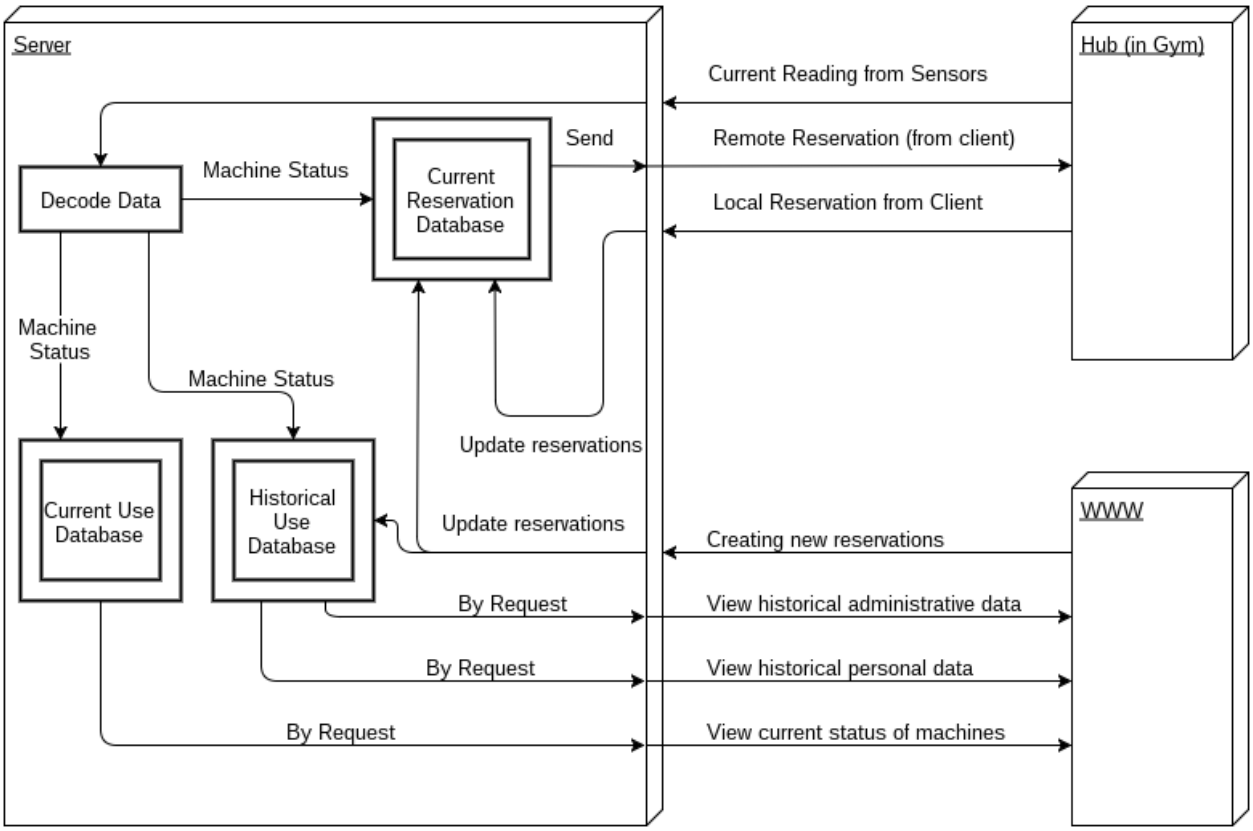

Figure 7: UML Diagram Detailing the Server Architecture

The server's main purpose will be to capture and receive the current use from the hub, while sending any remote reservations gained via HTTP requests. It will also log and display current use of the machines for personal and administrative use.

9 Design Criteria, Alternatives, and Decisions

The design criteria, alternatives, and decisions for the project were divided into many parts based on the main software and hardware components of the gym reservation system (see **Figure 3**). These main components are described below.

9.1 Sensor Systems

The sensor systems constitute the means by which data is gathered from gym machines as to whether or not they are currently in use.

9.1.1 Design Criteria

The design criteria for the sensors used to gather machine usage information include a few characteristics. One criterion is that the sensors are capable of gathering movement data (like an accelerometer, gyroscope, or magnetometer) because each sensor will be mounted on a moving part of the machine. This data must be

acquired reliably, which reflects the design norm of trust. Another criterion is that the sensors be relatively small (less than 5 in^2) so that they do not impede the use of the machine. Long battery life, on the order of years, is also a criterion for the sensors because it becomes inconvenient to replace the sensors for a whole gym. Through the sensors being small and having a long battery life, the design norm of delightful harmony will be incorporated. Additionally, a criterion is the ability of the sensors to operate in a mesh network, since this will decrease the amount of communication the hub has to do and will allow for shorter data transmission to and from each sensor. Lastly, the price of the sensor network must not be too high so that a gym owner could feasibly purchase it. By keeping the price within a reasonable range, the design norm of caring will be incorporated by considering the customer's needs.

9.1.2 Design Alternatives

The sensors chosen for the project must satisfy the design criteria set forth above. In researching different types of sensors, it became apparent that usually only two out of three desired characteristics exist in most sensors. These three main characteristics are the ability to use the sensors in a mesh network, the low cost of the sensors, and the ability to run on battery power for long periods of time. The final choice of sensors possessed all three characteristics, and is one of the main reasons these sensors were chosen for implementation in the design.

Much research has been conducted recently regarding sensors and sensor nets, but there are not many commercial products that are available. Research has been conducted at the University of California-Berkeley [5, 6] and the University of California-Los Angeles [7] regarding sensor systems and sensor networks. In addition, much research has been conducted at Stanford regarding the TinyOS operating system that runs on such sensor networks [8]. All of this research has demonstrated the ability to construct a sensor network that operates reliably and at very low power, but few companies have advanced this research into a commercial product.

Despite the lack of commercial products, a few places do sell sensor networks. One of them is MEMSIC, which sells wireless sensor networks for a variety of applications, including educational, industrial monitoring, research and development, and location tracking. However, the boards that serve as wireless sensor nodes are a little expensive and are larger than desired for the project. [9]

Linear Technology is another company that sells wireless sensor networks. It provides nodes, or motes, at a low price that operate at very low power. However, these motes lack the sensors desired for the project, so additional sensors would need to be bought. [10]

Developing or buying an Arduino board with the desired sensors is also an option, but the difficulty is found in finding an Arduino board that is capable of both mesh networking and sensor systems, is relatively cheap, and has long battery life. [11] There did not appear to be any public mesh networking libraries available for use that were built for Arduinos.

Bluetooth SensorBugs are also an option. They contain the desired sensors, are very small, and operate at low power. However, like many other sensors, they do not have the ability to form a mesh network, which is a key component of the project. [12]

Yet another option for wireless sensors is the CSRmesh development board. This board operates reliably within a wireless mesh network. However, it is expensive and does not include the sensors desired for the project (its main application is for lighting). [13]

Another option for sensors are TI SensorTags, which contain all of the desired sensors, operate reliably within a mesh network at low power, and are not too expensive. Because TI SensorTags satisfied all of

the design criteria, they were chosen as the best design alternative. A more detailed description about this design alternative may be found in the section below. [14]

9.1.3 Design Decisions

As seen in the previous section, each one of the design alternatives lacked one of the three main characteristics desired for the sensors with the exception of the TI SensorTags. As can be seen in **Figure 8** and **Figure 9**, the TI SensorTags possess the desired sensors in addition to many other sensors. These include a 9-axis motion sensor, a temperature sensor, a humidity sensor, an altimeter/pressure sensor, and an ambient light sensor. In addition, the TI SensorTags have the ability to communicate in a mesh network using Bluetooth, ZigBee, or 6LoWPAN. The battery lifetime for these sensors is about one year, and the data collection rate from the sensors can be modified as desired. In addition, the price per SensorTag is $29, which is much lower than some of the other design alternatives. Because the TI SensorTags satisfied the criteria of having the desired sensors, operating reliably in a mesh network at low power, and having a lower cost, they are chosen to be implemented in the design. Each of the other design alternatives lacks one of the desired design criteria, as demonstrated in the decision matrix in **Table 1**. [14]

Table 1: Sensor Systems Decision Matrix

Design Factors	Weight	MEMSIC	Linear Tech	Arduino	SensorBugs	CSR Mesh	SensorTag
Movement Sensor	9	8	0	8	9	0	10
Small Size	8	5	7	6	10	7	9
Long Battery Life	8	10	10	5	9	8	8
Mesh Networking	7	10	10	1	0	9	8
Low Price	6	6	7	8	7	4	7
Total		298	248	215	275	207	324

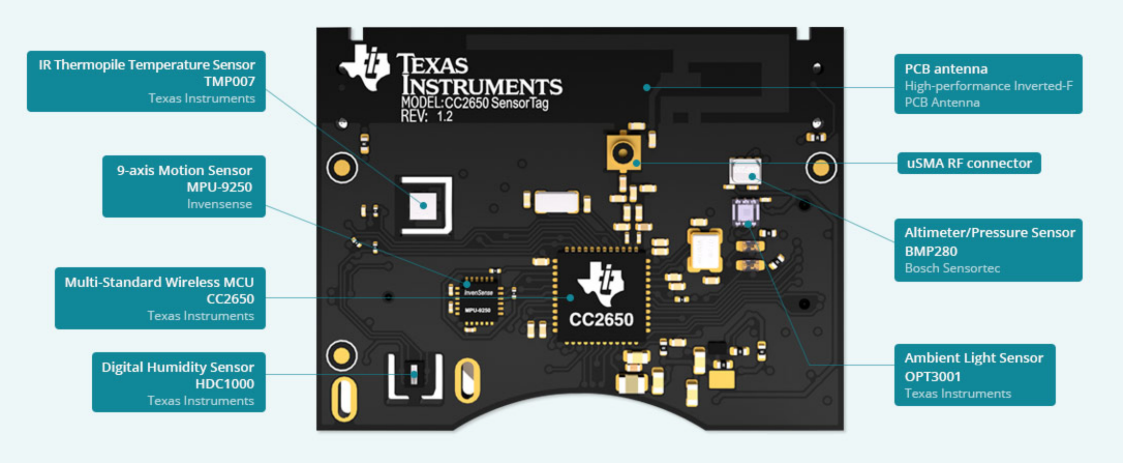

Figure 8: TI SensorTag Sensors and Microcontroller [15]

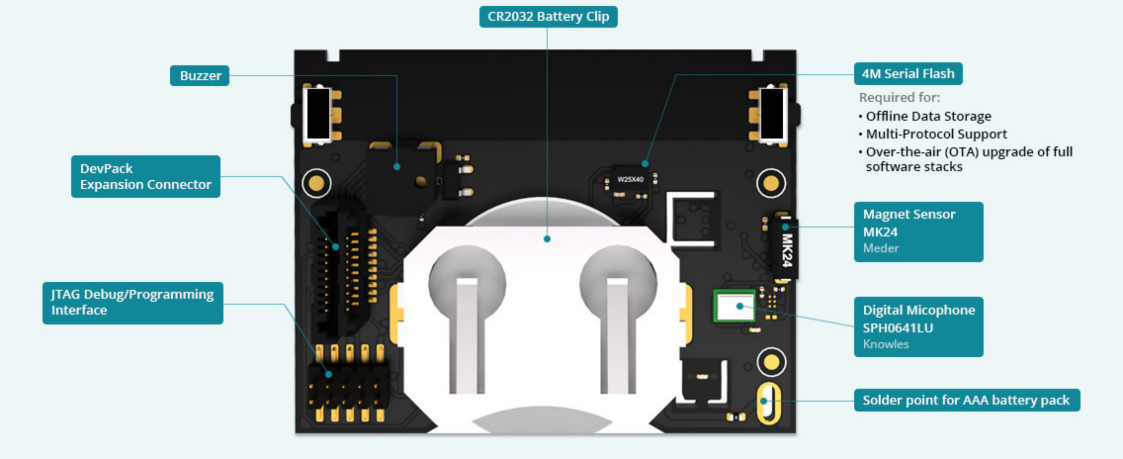

Figure 9: TI SensorTag Sensors and Connections [15]

9.2 Display Interfaces

The display interface allows the user to see current reservation information for a machine.

9.2.1 Design Criteria

Multiple design criteria have been used to select the optimal display interface. Battery life is an important criterion as it reduces the time spent by gym managers replacing or recharging batteries.

The range of a display interface and its ability to connect in a mesh network are also design criteria as they affect how large of a gym in which the system can be installed.

Simplicity is a third criterion as smaller or simpler boards are less obtrusive, promoting delightful harmony. In order to make the transition from a prototype to a real product, boards with available and open source hardware design files are preferred.

Lastly, the display interface constitutes a large portion of the cost of the system. Any variation in the cost in a single unit is multiplied since many display interfaces are purchased for one system. Thus cost is an important design criterion.

9.2.2 Design Alternatives

As a result of these criteria, four devices have been chosen as alternatives for small low-cost boards that could fulfill the design requirements: a Raspberry Pi, an Arduino, a CC3200 Launchpad, and a TI SensorTag.

For the display component of the display interface, the applicable criteria are battery life and price. Due to the fact that the board needs to run on batteries, the display cannot require a significant amount of power. This negates most typical Liquid Crystal Display (LCD) options. The two low power options that were found are E-Ink (also known as E-Paper) and Memory LCD. E-Ink has the advantage of requiring no power when the image is not changing, which would be most of time in this project. Memory LCD, on the other hand, does require some power, but typically under 200 µW, which means a single AA battery would last over 200 days. The original target display size was 2” x 3”; however, the closest available size for either type

of display is 2.7" (35mm x 59mm) with similar prices. Each of the four board alternatives can support at least one of these display options.

The Raspberry Pi (shown in **Figure 10**) is a credit card sized computer that runs a full Operating System (OS), such as Linux or Windows 10, with Universal Serial Bus (USB) ports and a High-Definition Multimedia Interface (HDMI) output. Most importantly, it has 40 General Purpose Input/Output (GPIO) pins that can be used to connect the display or other features like buttons and a wireless communication device.

The advantage of this board is that it has the most community developed libraries, including already built libraries for mesh networking and for controlling E-Ink displays, which would make the display interface very easy to build. Breakout boards for E-Ink displays are available for the Raspberry Pi and are well priced, which would make adding the display simpler. This board is also inexpensive; a $20 model would easily fit the requirements.

However, the Raspberry Pi has much more functionality than needed. Combined with fact that it is running an OS, it uses much more power than any of the other options, making it virtually impossible to run off battery. There are some batteries that might work for a few days, but they are expensive and rather large for this board. Along with the fact that it is not an open source board, the complexity of the Raspberry Pi would make it hard to use as an actual product. The price of a display interface with this board would likely be about $90.

Figure 10: The Raspberry PI, a Credit Card Sized Single Board Computer

The Arduino is an open-source prototyping board that runs a single program. It also has GPIO pins for controlling the display and connecting a wireless communication device.

Some basic Arduino boards are quite low priced, and there is also a big community behind Arduino boards. The fact that an Arduino runs only a single program can make it very power efficient, thus making operating it by battery over several months plausible, even using AA batteries. There is at least one reasonably priced E-Paper breakout board that eases the process of adding a display.

The only mesh network library found was for a radio frequency (RF) radio, though that may not be a disadvantage as it may be more power efficient than some other solution. Also, a lot of libraries use code that does not take power efficiency into account, which limits either the battery life or the libraries available. The price of a display interface with this board would likely be about $60.

The CC3200 chip is a microcontroller with built-in Wi-Fi. The CC3200 Launchpad is a board designed to provide features for this chip such as GPIO pins, USB interface, and power.

The primary selling point of this chip is the built-in Wi-Fi which translates to good power efficiency, allowing the board to run very well on batteries. An E-Ink breakout board is available and has available design files.

Along with the fact that the design files for the Launchpad are easily accessible, this board would be the easiest to transition from a prototype device to a commercial one. An additional advantage is that the built-in Wi-Fi is FCC approved, unlike many add-on Wi-Fi dongles, which removed a roadblock to commercialization.

However, no mesh network library could be found for the CC3200. The E-Ink breakout board for the CC3200 Launchpad is also quite expensive and large. The price of a display interface with this board would likely be about $80.

The TI SensorTag is a small board made for prototyping IoT devices and is shown in **Figure 8** and **Figure 9**. This board has a built-in communication device that supports ZigBee, a protocol designed for mesh networking, and is made to last extended periods of time on a coin cell battery. It also has a devpack for a "smart watch," which acts as a breakout board for memory LCD displays. The board has open hardware and software files which would facilitate the transition to a commercial product. Since the SensorTag was chosen as the sensor for the system, selecting it would result in only having to use one type of board for both the sensor and the display interface, thus streamlining the system. This might also eliminate the need for a separate sensor, which would reduce cost significantly.

However, the devpack comes with various accessories including a 1.3" memory LCD, while the only desired component is the connector for the memory LCD. The larger memory LCD display would need to be bought separately. The price of a display interface with this board would likely be about $70. This cost could be further reduced if the breakout board for the memory LCD display could be replicated so that the smartwatch devpack would not need to be bought for each display interface.

9.2.3 Design Decisions

While overall the Raspberry Pi is a good board, the difficulty and price of running it by battery and its complexity mean that it has not been chosen for the system. This would be a good solution if the project allowed for a power supply instead of batteries. The Arduino is a good option since it is the least costly and fits all the requirements fairly well, making it the second best choice. The CC3200 Launchpad would be the easiest choice in transitioning to an actual product, but it would be the most difficult to use in getting a prototype working. For this reason, along with the cost, this solution was not chosen for the project. While the price of the SensorTag may not be as low as the Arduino, the battery life and the built-in mesh networking are huge advantages. The fact that this board can also be used for sensors and is open source make this the top candidate for the project, as shown by the decision matrix in **Table 2**.

Table 2: Decision Matrix for the Display Interface

Design Factors	Weight	Raspberry Pi	Arduino	CC3200 Launchpad	SensorTag
Battery Life	10	2	7	7	10
Network Capabilities	5	8	7	3	10
Simplicity	2	8	10	1	10
Open Hardware	3	8	8	5	9
Cost	10	11	17	12	14
Total		201	321	245	338

9.3 Hub

The Hub is the component through which the data from the display and sensor networks is passed to the server and from the server out to the displays. It is essentially the gateway for the system between the gym

and the Server.

9.3.1 Design Criteria

The most important aspect of the Hub is its ability to communicate. Since it can be hidden and plugged in, form factor and battery power are not necessary considerations. This opens up options to using full small computers as long as they have the ability to communicate with the sensors and displays, along with the computing power necessary to serve as the client to the server. In order to minimize the cost of the system, the board needs to be cheap but still have the necessary computing power. For the communication to be as reliable as possible an Ethernet port is necessary on the Hub.

The design norms that can be applied to the Hub component are transparency and trust. Transparency is shown by being open and forward with the gym administrators as to what information the Hub is passing onto the server and how it uses the gym's existing Internet for that process. Trust is applied through the security of the software and the encryption of the data that is passed back and forth between the Hub and the Server. Ensuring that the user's data is secure is important in order for users to feel safe in continuing to use the system.

9.3.2 Design Alternatives

The primary options for the hub were different models of the Raspberry Pi, the Intel Edison board, or possibly some other single board computer. There is an increasing number of single board computers available on the market, and they are getting cheaper and faster. In considering these various boards, one large difference is the size of the community using the boards and the number of devices available that can easily interface with them. Most of the boards would have worked well for this application, as the necessary network communication can be connected via a USB port and the software needed to collect the ZigBee data can run on Linux. Due to the fact that this component can be plugged into permanent power, this opens up the possibility of any computer device running Linux and having a USB port as being a viable option. The reasons for narrowing this field down to simply the Pis and the Edison board is due to their large support, stability, and their low price point.

The Edison board has the advantage of having lower power consumption and the support of a large company. However, it does not have an Ethernet port and is approximately double the price of a Pi, as can be seen in **Table 3**. The Edison board is also intended as more of a sensor board than a small computer, though it could work for this project with the right breakout boards. [16]

There are multiple models of the Raspberry Pi boards, each one with improved specs over the previous model, generally in the form of more Random Access Memory (RAM), a faster Central Processing Unit (CPU) and Graphics Processing Unit (GPU), and additional ports. The main difference with the most recent Pi, as compared to previous ones, is its ability to run Windows 10. This is not currently a requirement of this project, though. However, if the sensor's software is made for only Windows, it could be a potentially useful aspect of choosing the newest Pi, the Raspberry Pi 2 Model B. The older versions of the Raspberry Pi should be adequate for this project, though. As long as there is no need for Windows 10, then the computational power differences should not make an impact on this project. For the amount of computing power that Pis provide they are the cheapest single board computer currently available on the market. This is due to their success and large scale production. An additional advantage of this large scale is that Pis have a stable Linux Operating System (OS) developed specifically for them, which should help minimize any bugs in the system and increase the stability. [17]

The other aspect of the hub to be considered besides the single board computer is how it will communicate

with the sensor and display network. The only connectivity built into the Raspberry Pi is an Ethernet port. This means that whatever wireless technology is used will need to be added through either the GPIO pins or a USB dongle. There are options for Bluetooth dongles [18], Wi-Fi dongles [17], and ZigBee modules [19] that a Pi can use. All of these options enable the Pis to connect to whichever network ends up being used. This is one area where the Edison board has an advantage, as it has Wi-Fi and Bluetooth connectivity built into the main chip and is primarily designed as an IoT board [16]. However, since reliability is an important factor, the inclusion of an Ethernet port on the Pis is more important than the Edison Board's capability with Wi-Fi and Bluetooth. This is especially true since the network technology may be something besides Wi-Fi or Bluetooth, which would require an additional dongle to any chosen board, such as ZigBee or 6LoWPAN.

9.3.3 Design Decisions

The final board that was chosen was the Raspberry Pi 2 Model B. This board was chosen because for its price it had the most options. Primarily, it was chosen to accommodate for potential future design changes, such as running Windows 10 on it. This board is available for $35, has more than the needed requirements, and can use the ZigBee module via the GPIO pins as opposed to the USB through a breakout board, saving some money. However, initially the board that was being used to prototype was a Raspberry Pi Model B+. That is because Calvin already had several older Pi models available for use and the small advantage that the Pi 2 would offer is not necessary, until it was determined that the Pi B+ was too slow for our purposes, at which time we purchased the Pi 2. This change from the initial design decision saved some money initially, however it became necessary to purchase a newer pi before the end of the project as the time that its increase in speed saved the team was worth the cost.

Table 3: Decision Matrix for Hub Hardware

Price		$40	$95	$35
Design Factors	Weight	Pi 2 Model B	Edison Board	Pi 1 Model B+
Network Capabilities	10	5	5	5
Ethernet	8	10	1	10
Range	8	5	5	5
Costs	5	8	1	9
Reliability	10	10	10	9
Total		310	203	305

9.4 Server

The server will handle all connections requesting information about the current status of the machines, along with making and viewing reservations for the equipment. It will also handle sending historical data for administrators and personal historical data for individual users. It takes all the data from the server and stores it, creating these services for the client.

9.4.1 Design Criteria

The server's primary requirement was that it must be reliable. Both the gym administrators and the gym's clients must be able to trust that when attempting to ascertain information about the machines, whether historical or current information, that the information can be sent to them. The server must be capable of modern networking protocols and functions, including HTTP/HTTPS requests and creating and maintaining

TCP network connections. The HTTP requests requirement is to enable the server to send information about the state of the machines over the internet to clients of the system. The TCP network connections are to ensure a reliable connection between the hub and the server for consistent and reliable information sharing. The server must also be powerful enough to encrypt data and send data over encrypted connections to make sure that all client information is protected.

9.4.2 Design Alternatives

There are in essence two main players in the server world: Windows and Linux. Both of these options have the capabilities to meet the requirements stated above in the Design Criteria section, however Linux has several advantages. The first is that it is free, which will reduce the cost of our system. The second is that it is often more reliable.

9.4.3 Design Decisions

It was chosen to use Ubuntu 14.04 as the server. It provides the reliability required to provide the users with a trustworthy experience. It also should be low maintenance once installed, as security and software updates can be handled automatically using built-in Linux functionality. Ubuntu 14.04 is the Long Term Supported (LTS) version of Linux currently and will be supported for several more years. It also has the modern capabilities to use Python and other libraries to create a secure and consistent connection with the Hub. Also because the hub is running Linux, it makes sense to continue to use the same operating system for ease of upkeep. It also aligns with the team's goal of transparency because Linux is an Open Source solution.

9.5 SensorTag Network Type

In order to pass information between SensorTags, a type of network protocol must be used. Each network protocol has various strengths and weaknesses, and some network types are better suited for the requirements and application of this project.

9.5.1 Design Criteria

As listed in the requirements section, each sensor must have network capabilities and have the ability to form a mesh network where the sensors can communicate with and pass data to one another. This communication must be reliable and timely in order to give the user real-time data about machine usage. In addition, the network type used must be capable of communicating over long distances so that information from a SensorTag at the edge of the gym is able to reach the hub, either directly or through a series of other sensors.

9.5.2 Design Alternatives

The design alternatives for the type of network used with the SensorTags was dependent on the kinds of networks that were compatible with the SensorTags and could be flashed onto them. One alternative was Bluetooth, which comes installed on the SensorTags and is easy to use. However, the Bluetooth option could only be used with a provided mobile application, and Bluetooth does not support mesh networking. Another alternative was ZigBee, which does support mesh networking and can communicate over longer distances. However, very little to no documentation about using ZigBee with SensorTags is provided, and it was not

obvious as to how to get ZigBee working properly in this context. The last alternative was 6LoWPAN, which also supports mesh networking and can communicate over longer distances than ZigBee. 6LoWPAN is relatively new, but enough documentation and open source code is provided for it to be implemented on the SensorTags. This open source code is Contiki, an operating system written for small sensor nodes such as a SensorTag.

9.5.3 Design Decisions

Contiki is an open source real time operating system that can run on the SensorTags and utilizes 6LoWPAN wireless networking. Its website describes Contiki as "The Open Source OS for the Internet of Things". The team decided to use Contiki primarily because it was the only way found to use 6LoWPAN networking with the SensorTags and it provides several examples and tutorials on how to use it and get started with it. Contiki is publicly available through a git repository and there are pre-built .bin files that can be flashed to the SensorTags using the debugger for several different examples.

Contiki also offers an easy way to get a development environment up and running with something they call instant Contiki. This is a download from their site which is an Ubuntu image made to run in VMWare and already has all the tools and libraries needed to work with Contiki. There are many more tools for Contiki than what we used, some of which are meant to simulate a larger scale implementation of a sensor network and others made to just help in tweaking the code. The only thing that seemed to be missing from the Contiki tool kit is a good debugger, but this is likely because they'd have to make a different debugger for each board Contiki is available for, which is quite a few. This only impacted the project in that in order to test changes to the Contiki codes, it needed to be flashed to a SensorTag and tested directly.

9.6 Sensor Data Acquisition and Communication

In order to reliably acquire data from various sensors in real time, a few methods existed within the Contiki operating system. Each option had its benefits and downsides, and many different methods and approaches were taken before the final method was selected.

9.6.1 Design Criteria

As outlined in the requirements section, the sensors need to output data in a way that preserves battery life on the order of months or years. In addition, sensor data must be reliable, accurate, and have a clear way to be passed from one sensor to another.

9.6.2 Design Alternatives

Each design alternative was based off of an example for the CC2650 SensorTag provided by the Contiki operating system. The simplest design alternative, which was implemented first, was outputting sensor data over UART (Universal Asynchronous Receiver/Transmitter). The example was modified to output a stream of data where each set of data contained the machine ID number and gyroscope and accelerometer data for each of the coordinate axes. With this example, each SensorTag had to be connected to a Raspberry Pi, but the success of this design alternative showed proof of concept in that a working system could be designed from the bottom to the top (sensors to the user interface).

Many other design alternatives were tried with varying degrees of success. These included modifying the various examples in the Contiki operating system and flashing them onto the SensorTags for testing. Some

methods implemented UDP or TCP over 6LoWPAN, others implemented other versions of IPv6, and still others consisted of modifying examples written for other devices, but each example used 6LoWPAN mesh networking to pass data between the SensorTags and the ultimate destination, the Raspberry Pi. Multiple methods were attempted in gathering data from the SensorTags, including a USB dongle that could be configured in multiple ways, such as a packet sniffer or a slip radio.

The last design alternative that was tried was developing and modifying the code for the CC26xx web demo, provided by the Contiki operating system. This web demo supported multiple ways of transferring sensor data between devices and an ultimate destination using the 6LoWPAN mesh network, including network UART, a 6lbr client, a CoAP server, HTTPD, and an IBM Quickstart/MQTT Client. A SensorTag was used as an antenna to receive 6LoWPAN packets for the Raspberry Pi, and this example provided basic functionality in terms of gathering sensor data from each SensorTag.

9.6.3 Design Decisions

The web demo example provided by the Contiki operating system was the final design chosen to be used for the project. A few configuration changes were made to the example to better suit the application of the demo in this specific project, such as turning off certain sensors and modifying the network address prefix. The demo passes reliable sensor data and device ID information from various SensorTags to the Raspberry Pi in real time using mesh networking, which fulfills all of the requirements and design criteria for this part of the project. Once collected, a short Python script implements some signal processing to determine whether or not each machine is in use, and this information is then passed on to the server.

9.7 Display Software and Communication

The SensorTag displays can be utilized by a few different RTOS's or networking methods. The watch devpack display is used with the CC2650 SensorTags from TI.

9.7.1 Design Criteria

The display software and communication needs to be reliable, secure, and quick to update. It needs to also display the information in a way that is easily seen and understood by the user.

9.7.2 Design Alternatives

The display software and communication has two different options. It could work with Contiki using 6LoWPAN, which would be preferable as it would then share a network with the other SensorTags that aren't acting as displays. However, with this path there are no display drivers currently available with Contiki so they would have to be completely re written from TI-RTOS to work on Contiki.

The second option is to use the existing Bluetooth phone app available from TI. With this option the SensorTags would have the default TI-RTOS running on them, and just need an over the air (OTA) update from the app in order for the display to work with them. Once updated there is a textbox in the app, and whatever is entered there is then displayed on the SensorTag. This wouldn't use a mesh network, but would require much less work to get functioning with our database to automatically read the reservation status of the machines.

9.7.3 Design Decisions

Several weeks were spent trying to re write the device drivers to work on Contiki. However, after that time it was decided that the team would switch and add to the Bluetooth app for the demo, as some of the files that the driver's interface with for TI-RTOS weren't available or found for Contiki, primarily one to interface with the device pins (where the display attaches) and one providing power to peripheral devices (needed to power the display). The functionality that needs to be added to the android app is to automatically check the database or website for the reservation status of a specific machine. Using the Bluetooth app for this is primarily for the senior design night demo, for a production scale system we would have a custom built and designed board and display, utilizing a mesh network technology.

9.8 Hub to Server Communication

The server must communicate with the hub to determine the status of the machines and communicate reservations to the displays

9.8.1 Design Criteria

The communication between the hub and the server must be reliable, and fast enough to transfer all the required data.

9.8.2 Design Alternatives

One alternative is for the hub and the server to communicate via TCP sockets. The hub and the server would both setup sockets listening for updates. This has the advantage of being fast, reliable, and relatively easy to implement.

A second alternative is for the hub to use http requests to send updates through the server's http server that delivers the website, and also use those requests to get updates from the server. This has the advantage of being simple, and streamlining the database access methods on the server. This option is slower than the TCP socket method.

9.8.3 Design Decisions

The http request method was chosen because the database was reporting errors when updated from multiple TCP socket transmissions, which could impede reliability. Because the http requests go through the http server, thus reducing the threads accessing the database to one, the http request method does not present this problem. Any issues that may arise due to the slower speed of the HTTP method can be negated if the data transferred is aggregated before being sent, thus reducing the number of packets sent.

9.9 Database

The database must be able to scale with both large amounts of traffic and large records. The large amount of traffic scalability is required so that when the website has large amounts of visitors, they do not experience long wait times when attempting to view information. In terms of the large records scalability, the database needs to be able to handle queries on large amounts of data quickly so that when managers want to look at

historical usage of machines over extended periods of time, they are able to without having to wait inordinate amounts of time between requests.

9.9.1 Design Criteria

The database must be well structured, easy to add features to, and well abstracted.

9.9.2 Design Alternatives

An alternative to the database would have been in memory storage or file storage for historical data. In memory storage, while fast, is not reliable and if the system were to go down or restart, all the data would be lost. While file storage is easy to set up, it is very slow to read through large files and parse them. This would not allow the database to be scalable as required.

9.9.3 Design Decisions

The team decided to use the SQLAlchemy framework in Python for a variety of reasons. The first reason was that our model was already well defined in Python, so to interact with the database in native Python is a large advantage. Also, SQLAlchemy provides an abstraction layer above the database, which allows us to easily switch between different database implementations. This allows us to test many different scenarios, as well as have a production version of the database that can be changed later if necessary.

9.10 Website

The website serves as an interfaces with the end users, including gym users wanting to check what machines are available or reserve machines, and gym administrators wanting to edit the system or view machine statistics. The website contains a front end, which is what the users interact with, and a back end that interacts with the database, provides the dynamic data that needs to be displayed and generates the web pages.

9.10.1 Design Criteria

From a design standpoint, we wanted the website back end to easily be able to interact with both our database and our model. These are both written in Python, so it would follow to use a website that could take advantage of Python.

The website front end needs to look good for the end user so that it will not detract from their experience using our system. We are also assuming that most gym users will be using the website on their phones, so mobile friendliness is also essential.

9.10.2 Design Alternatives

A first alternative for our back end is Flask, which is a micro webdevelopment framework for Python. A major alternative to Flask is Django. While Django is appealing, it has more setup and configuration options that the team was not familiar with how to set up.

There are many front end packages intended to make beautiful and mobile friendly websites. The first alternative was using the same one used for our team website, called future imperfect. The second alternative considered is the bootstrap framework, a popular front-end framework which advertises itself as "the world's most popular HTML, CSS and JS framework for making responsive, mobile first projects on the web."

9.10.3 Design Decisions

Flask was chosen as our back end framework because it was simple to start, easy to use, and provided very good documentation. Additionally, it interacted very well with our Python database, written in SQLAlchemy, and when interacting with the native Python objects that represent machines, reservations, etc.

The bootstrap framework was chosen for our front end because it was more general and allowed more features, whereas the future imperfect package is designed for making website centered around articles and posts.

10 Integration, Test, and Debug

Testing is a crucial part of the design process, as it determines whether or not the system meets the design requirements.

10.1 Tests passed

The sensors have been tested to make sure they are sensitive enough to detect the vibrations of functioning equipment. To do this, single sensors were installed on Calvin College gym equipment. The output from the sensors when the machine was in use as opposed to the noise were different enough to be distinguishable, so the sensors are sensitive enough. This test also determined that the physical connection of the sensor to the equipment can sufficient to endure normal use.

The hub and server have been tested to make sure that information is properly propagated between the sensors, the display interfaces, and the server. It took take less than one minute for information to be transmitted from any device to any other device, including the server. In fact it usually only took a few seconds. This was tested by stimulating the sensors, recording the response time of the server detecting the change, and recording the response time of the displays to changes made on the server.

The website was tested by verifying that current usage and reservation information were received without any communication errors and were displayed to the user correctly. Reservations made using the website were also be transmitted to the server without any errors.

Unit tests were written for the software to verify that the code functions correctly at all times, and these tests pass.

10.2 Tests failed

The display interfaces were be tested to determine whether all the required information can be displayed while maintaining good visibility from a distance of about 7'. This test was failed, but that is made up for by the decision to include a large screen display which is easily visible at much more than 7'.

Both the sensors and the displays were tested for battery life. During our development process, the batteries in the SensorTags had to be changed multiple times, which indicates that the battery life is not long enough.

10.3 Future tests

The network will be tested to verify that all devices communicate properly and that the mesh network is working as expected. Multiple devices will be activated one at a time; each should connect to the network rapidly without disrupting the other devices' connection. Devices will be added outside the range of the central hub with intermediate devices in between to verify that the devices are properly forming a mesh network instead of all directly connecting to the hub. No data transmitted over this network should be corrupted.

All the equipment will be left running for multiple days with some occasional artificial activity (such as manually moving a sensor to simulate a running machine) during typical gym hours to verify that the system will work without crashing for at least 24 hours.

Once all the single feature tests are completed, integration testing will be performed in order to determine that the system works properly in a real gym environment. This involves setting the system up in a gym and letting it run during a regular day. This will confirm or invalidate the results of the single feature tests as the integration test is a more realistic testing scenario. The same devices being used across different equipment will determine the generality of those devices. The status of the gym according to the sensors will be monitored and regularly cross-checked with actual gym usage to ensure that the gym usage information is correct.

11 Business Plan

The business plan for this gym reservation system is divided into two parts, which include a marketing study and a cost estimate. The marketing study describes the need for a gym reservation system, and the cost estimate gives a detailed breakdown of the costs involved in implementing such a system.

11.1 Marketing Study

The marketing study for the gym reservation system includes a description of similar existing systems that might be in direct competition with this product. It also includes a survey of prospective customers and managers to gain a sense of what characteristics they desire to see implemented in such a product.

11.1.1 Competition

While there are many reservation systems in the world that exist for gyms, most systems are only capable of reserving gym facilities or sports equipment. Few gyms have the ability to reserve specific machines at specific times. A few gyms do, though, including some gyms on the campuses of Syracuse University [1] and Harvard University [2]. Gyms at both of these locations use machines built by Precor, a company that manufactures fitness equipment [3]. Precor in turn uses the fitness software provided by Preva [4] to have a reservation system capability that is built into each one of their machines.

Each of these systems has the reservation system capability built in to the machine itself. There are no known companies that sell gym reservation systems that are not built in to the machine itself. The system

proposed in this project is unique in that it can be applied to any existing gym without having to replace all of the machines in the gym.

In addition, the cost necessary to implement this system in an existing gym is less than the difference in price between machines that do not have a reservation system and machines that have a built in reservation system. As a result, it is smarter economically to implement a reservation system that is separate from each machine. In addition, a reservation system that is separate from each machine is able to be generalized for any gym and is modular in nature.

11.1.2 Market Survey

A survey of prospective customers asking about the need for a gym reservation system was conducted. Both gym users and gym managers were questioned as to what features they would like to see in a reservation system and what price they would view as reasonable for the system. Out of a total of 70 respondents who went to 12 different gyms, 60.4% indicated that it would be very useful to view the availability of gym machines on a mobile application or the web. In addition, 47.6% of respondents indicated that it would be very useful to remotely reserve cardio machines in the gym. Cardio machines seem to be used more frequently than weight machines, as 55.5% of respondents indicated that they use cardio machines most of the time, whereas only 41.3% of respondents said that they used weight machines most of the time. The majority of responses pertaining to a gym reservation system were either positive or neutral, with very few respondents indicating that some sort of gym reservation system would not be useful. The results of this survey demonstrate the market need for such a system.

11.2 Cost estimate

The cost estimate includes both the summary of the development costs during this project and an estimate of the production cost of a system.

11.2.1 Development

The vast majority of the budget for this project is used for purchasing sensors and equipment for the display interfaces. The items purchased or planned to be purchased are shown in **Table 4**.

Table 4: Class Development Budget Usage

Quantity	Part	Price per Unit	Total
2	TI Debug DevPack	$15	$30
1	TI Watch DevPack	$19	$19
4	Memory LCD Connector Board	$2.5	$10
10	TI SensorTag	$29	$290
5	2.7” Sharp LCD Display	$30	$150
			$499

The Raspberry Pi used for the hub is a model B+ owned by the Calvin engineering department that has been borrowed for the duration of the project. A ZigBee wire antenna for the Raspberry Pi was also borrowed from the Calvin engineering department. These items and their values are shown in **Table 5**.

Table 5: Items Loaned by Calvin Engineering Department

Part	Value
Raspberry Pi Model B+	$30
ZigBee Mesh Wire Antenna	$27
	$57

11.2.2 Production

The cost of producing a single system includes both fixed and variable costs. Both of these affect the price at which the system will be sold for commercially.

11.2.2.1 Fixed Costs

In order to transition from the prototype developed in this project to a final product, it is estimated that about 1000 hours of additional design time would be required. At $100 per hour, this would cost $100000. The total development budget to get an initial commercial product would be about $125000. This price should be recovered over the first five year of production, and the annual development recovery cost is $25000.

To account for other fixed costs such as accounting, marketing, facilities, and further R&D, an additional overhead of 40% is added to total cost of the system.

11.2.2.2 Variable Costs

It is estimated that the system could be sold to about 100 gyms annually. A typical gym might have about 100 pieces of gym equipment that could be equipped with sensors and displays, which would lead to a total sale of 10000 display interfaces (with sensors built in), 100 hubs, and 100 servers. For the prototype, a TI SensorTag is used for the sensor and display interface. It has many more sensors than needed or desired and also comes with multiple wireless communication methods. It is estimated that a board without the extraneous features and with a small case would cost about $20.

For the hub, the final product would likely still use a Raspberry Pi, as creating a custom board would be complex, and this would likely result in a more expensive yet inferior product. Equipment such as an ethernet cable will be needed. These part costs are shown in **Table 6**.

Table 6: Parts Cost for One System

Quantity	Part	Price per Unit	Total
100	Display Interface	$20	$2000
100	2.7” Memory LCD	$18.63 [21]	$1863
1	Raspberry Pi Model B	$35 [22]	$35
1	XBee	$19 [24]	$19
1	50’ Ethernet cable	$5.25 [23]	$5.25
	Total		$3922

The cost of these parts does not include shipping and parts loss, so an extra 10% is added to the parts price as overhead. Over a total of 100 systems, the annual initial R&D cost recovery is $50 per system. Cloud Server services such as Amazon server will be used for hosting the server. This cost will be a monthly

cost, and will scale with how much usage it gets. For typical usage which might be similar to a moderate WordPress installation, a 3-year instance on a small server could cost $17.95 per month, or $646.2 for three years [25]

These costs along with an estimated sales price and profit are shown in **Table 7**. The corporate income tax rate applied is 35%.

Table 7: Cost of a Single System

Parts	$3922
Parts Overhead	$392
3 Year of Server Usage	$646.2
Total	$4960
Total with Overhead	$6944

11.2.2.3 Summary Financials

Table 8: Sales Price and Profit of a Single System

Cost of one System with overhead	$6944
Amortized Startup Costs	250
Total Cost	7194
Sales Price	$10000
Profit before Tax	$2806
Profit after Tax	$1824

Using a sales price of $10000 per system, the resulting profit of $1824 (shown in **Table 8**) results in a profit margin of 18.2%, which is satisfactory for this project.

12 Conclusion

Due to the lack of a similar products in the market and the ability to produce this product with relatively cheap hardware, this project is feasible and will provide a useful and enhanced experience to gym users and administrators. It will be necessary for the network technology used by the sensors and displays to be a mesh network in order to have the necessary range. The server and hub do not have any immediate problems, as they will be implemented largely though software. This project contains large amounts of software, and that will likely be a point which has the potential to provide a great user experience and added functionality beyond the basic goals. Since the team has already ordered and received most of the hardware necessary for a basic implementation of a prototype, the largest part of the project ahead is writing all the software to make all the hardware work together in harmony.

One possible future expansion on this project would be to work on the design of custom hardware for this project's purpose. This aspect may be explored more if there is adequate time towards the end of the project once the necessary goals have been accomplished.

There could be other ways of implementing a system that accomplishes the goals of this project, such as the initial idea of using a camera to visually analyze and determine use. In order to accomplish this project by use of visual analysis, there would need to be adequate cameras to cover the entire area of the gym, whose

sizes and shapes can vary greatly. The network to connect these cameras would need to have a much higher bandwidth than what a low power network, such as the suggested implementation, would be capable of. It would not be feasible to run the cameras on batteries, so additional wired installation would be necessary. The amount of computational power needed to do the analysis on visual data would be much greater than what sensors require. There could be ways to also accomplish the goals by using the gym user's smart phone. In order to implement this system, it would first be necessary for each user to have a smart phone. Each user would then also have to download the application, enter information, and opt into the reservation and sensing system. This would make the system very alienating to those who are only visiting the gym or are not comfortable with smart phone technology. The reliability of such a system would also be very inconsistent unless the user is asked what specific machine they are using, which would be bothersome, as it would have difficulty in determining the exact machine in use. Sensors and the IoT is a rapidly growing industry, and the method of implementation chosen takes advantage of many technological devices and advances currently available in the market.

13 Acknowledgements

Team 14 would like to thank the professors of the Engineering 339/340 class and especially Professor Michmerhuizen as our advisor for guiding us in this class and for providing insight into our project. We also want to thank our industrial advisor, Eric Walstra, for his excellent insight and ideas, and Roy Zuidema, the director of Campus Wellness at Calvin College, for his insight.

We are thankful to Bob DeKraker and the Texas Instruments vendors for their helpfulness in working with us to acquire the components we need, and we are thankful for additional funding from the Eric DeGroot Engineering Fund.

We would like to thank the friends and family that have given us ideas and encouraged us, especially the wives of two of our members for putting up with the time commitment of their husbands to this project.

14 References

[1] http://syracuse.reserve.subitup.com/

[2] http://recreation.gocrimson.com/recreation/facilities/Hemenway

[3] http://www.precor.com/en-us

[4] http://www.preva.com/en-us/exercisers

[5] http://robotics.eecs.berkeley.edu/ pister/SmartDust/

[6] http://webs.cs.berkeley.edu/nest-index.html

[7] http://cens.ucla.edu/

[8] http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page

[9] http://www.memsic.com/wireless-sensor-networks/

[10] http://www.linear.com/products/Wireless_Sensor_Networks_-_Dust_Networks

[11] http://store-usa.arduino.cc/

[12] http://www.blueradios.com/hardware_sensors.htm

[13] http://www.digikey.com/product-detail/en/DB-CSR1010-10185-1A/DB-CSR1010-10185-1A-ND/4841745

[14] http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/index.html

[15] http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/tearDown.html

[16] http://www.intel.com/

[17] http://www.raspberrypi.org

[18] http://www.adafruit.com/products/1327

[19] http://www.digi.com/pdf/chart_xbee_rf_features.pdf

[20] https://drive.google.com/folderview?id=0ByRzevjyNjCMeE1sSElHNzd6aVU&usp=sharing

[21] mouser.com part 852-LS027B7DH01A

[22] alliedelec.com part 70465426

[23] http://www.amazon.com/0877083042452-Network-Ethernet-Cable-Blue/dp/B000QZ001I/

[24] mouser.com part XB24-AWI-001

[25] http://publishingwithwordpress.com/estimating-costs-wordpress-amazon-aws/