Senior Project Final Report
(System for Workout Information Management)

Design Team: 04

Mark Archual (Team Archivist)

Tanner Daniels (Hardware Lead)

Ethan Schweinsberg (Software Lead)

Jack L. Wolfe III (Team Lead)

Faculty Advisor: Dr. Kye-Shin Lee

5/1/17

Contents

Abstract

Power racks are a weight machine used by swimmers to provide resistance while they swim away from a wall toward the center of the pool. The objective of this project is to build a modular data system that can be added to these machines to record and log quantitative information during their use. This information will be stored on a web server, and made available to the user for analysis and visualization through a web application. Workout data can also be downloaded and interpreted at a later time, independent of the web application. The data system should be water resistant, inexpensive, scalable for a various number of users, and require limited input to configure and use. The system will be design for the University of Akron Women's Swim team.

[MJA]

Section 1.01 Problem Statement

(a) Need

There are two different types of power racks used by The University of Akron Women's Swimming Team. One is operated by inserting a pin into a weighted plate, while the other is operated by filling a bucket with water. The weighted plate and water-filled bucket provide resistance to the swimmer as they push off of the wall and swim towards the center of the pool. These power racks have a number of shortcomings. For example, the machines do not provide feedback to the user in real time or record information that can be used to analyze the workout. Real time feedback can help a swimmer make adjustments during her workout. Additionally, recording information allows a user or coach to track progress over time. Tracking progress helps an athlete set and reach certain fitness goals.

[MJA, EES]

Section 1.01 Objective

The objective of this project is to design an independent data system that tracks different aspects of a swimming exercise performed with the power racks. The system will consist of three main components: a sensor board for capturing data, a display for providing real time feedback, and a server for hosting a web application. For each repetition, the sensor board will capture data for the amount of weight used, the amount of force applied, the distance traveled, and the amount of time to complete the repetition. The display will provide real time feedback as well as an interface to interact with the system. Information during each session will be recorded and then transferred wirelessly to the server, which will host a web application. The application will analyze the information and provide analysis on the individual's performance over time. The application will provide an interface that is accessible from any web browser. The system will be designed for Brian Peresie, the Head Coach of the Women's Swimming and Diving team.

[MJA, EES]

Section 1.02 Background

(a) Patent Search

A fitness machine containing a CPU, memory, and a network interface is described by *US Patent no. 2003/02003/0158014 A1*. This patent document contains the general concept of an exercise apparatus and computer communicating with each other through a data exchange port. According to the description in the patent, this technology would allow a computer to record data regarding a user’s workout session and download that data through the port to a network server or appropriate device. This patent is applicable to the proposed idea, since the proposed idea would also have the ability to record workout session data and download that data to a server or other networked device. However, this proposal is not for a fitness machine that records workout data, but an independent device that has the ability to be integrated into an existing fitness machine. In addition, this proposal discusses wireless networking technology, such as RFID and Bluetooth, which this patent fails to describe. These key differences make the proposed system unique from the system described in this patent.

A system for monitoring a user and providing a user a manner in which to interact with various components of a gym environment is described in *US Patent no. 2015/0133748A11*. In this patent, a wearable device is described that contains various networking technology (RFID, Bluetooth, Wi-Fi, etc.). According to the patented description, this device would enable a user to communicate with networked exercise components to monitor and assist with physical activity and provide feedback and motivation. This patent is applicable to the proposed idea, because there could be some type of wearable device capable of interacting with the system described in this proposal. A wearable device, like the one mentioned in this patent, could help transmit important data and information between the user and the proposed system.

An electromechanical device that applies a torque to a tethered swimmer in order to enhance training is described in *US Patent no. 7935029B2*. In this patent, a device is proposed that would have the ability to track and record distance, time, speed, force, and power of a tethered swimmer. In addition, this patent describes a torque control system which consists of an electronics control unit configured by software. This patent is very detailed in explaining the mechanical structure of the device and fairly vague on the electronics implementation. This patent is applicable to the proposed idea, because it describes a system that tracks and records the distance, time, speed, force, and power of an athlete that is tethered to a fitness machine, which is similar to how the proposed system would work.

[EES]

(b) Article Search

The article titled “*Strength and Power Training for the Elite Swimmer: Can Weights Positively Impact Elite Swim Performance when "Elite Performance" Requires 15 - 25 Hours/Week of Practice?*” was analyzed to understand and comprehend the importance of weight training in a swimming sport. The article is from “Olympic Coach Spring 2012, Vol 23 Issue 2”. The article states several ways in the benefits of weight training in any sport, including swimming. The article covers how there are many ways in which an athlete can develop a weight training program to help enhance their performance in the desired sport. This includes different workout plans revolving around either more weight and less repetitions versus lower weight and higher repetitions. Lastly, the article covers the necessity of weight training in an elite swimmers workout plan to achieve maximum performance potential.

“*A LTCC low-loss inductive proximity sensor for harsh environments”* from “Sensors & Actuators: A. Physical:” published by Elsevier B.V was used to develop ideas on potential sensor types for the position sensing aspect of the project. Proximity sensors use a magnet to signal when a piece of metal has come into contact with the sensor. Typically used on assembly lines to track pallets on the lines and to indicate when in position at a machine. This article talks about a design of an inductive proximity sensor that may be used in high temperature environments. An inductive sensor can be used to send a pulse to a module to do a certain task assigned to it by a computer such as turn on a timer or stop a timer when the inductive proximity sensor is triggered. This article explains in detail the capabilities of an inductive proximity sensor and how it generally operates.

The last article, titled *“Rugged Waterproof Push Button and Rocker Switches from Cherry Provide Price-competitive IP65 Protection”* was used to understand the possibilities of a waterproof button. The article covers a type of push button which is claimed to be very cost effective as well as having an IP65 Protection rating. This rating means the unit is dust tight as well as rated for waterproofing of a low pressure jet being directed at the unit. The article goes on to describe the different models that are currently offered.

[TMD]

(c) Other Source Search

Several cardio equipment machines exist on the market that can log information, such as heart rate, during a workout session. Typically, this data is transferred through a cable to a user’s phone or tablet, or the data is uploaded to an external website. Some fitness companies, such as Cybex, offer these capabilities in their elliptical or treadmill machines, but no company offers a modular solution for tracking data from weight training machines.

[MJA]

Section 1.03 Marketing Requirements

1) Record force, position, weight, and time data for each individual athlete
2) Coaches and athletes can view data on their own personal devices
3) Limited user input required to operate
4) Components on the pool deck are at least IP67 rating (hardware is dust proof, and waterproof up to 1[m])
5) Physically independent from the weight machine
6) Display time per repetition during use
7) Support up to ten users simultaneously

[MJA & EES]

Section 1.04 Objective Tree

The following objective tree shows the marketing requirements and how they relate to one another.

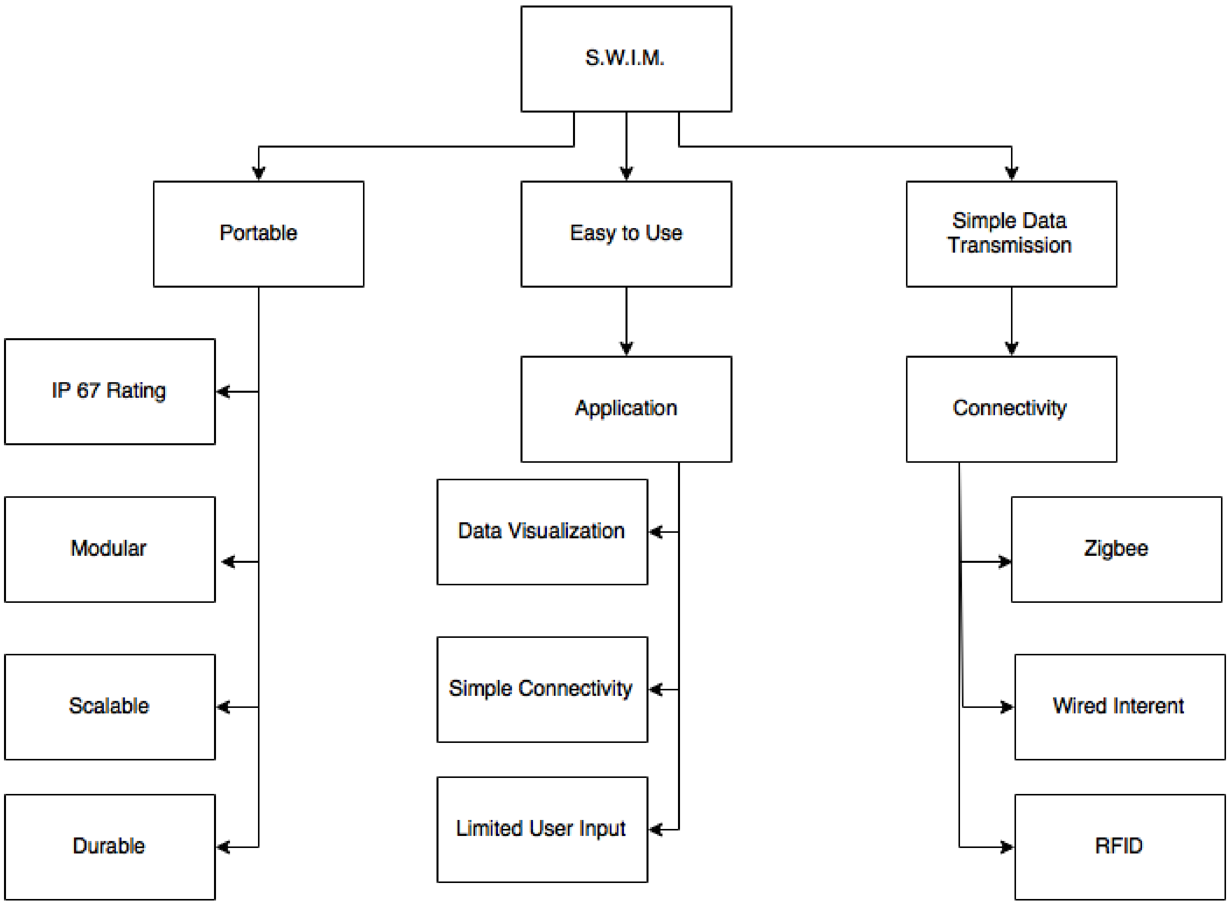


Figure 1: Objective Tree

[JLWIII]

Section 1.05 Demonstration of Ideas

The project will be tested by The University of Akron's Swimming and Diving Team. The design team could record and present a video of this system being used. Another option would be to have design team members actively demonstrating the system during the Senior Design presentations. The weight racks with the buckets are relatively lightweight (when empty) and have wheels on the bottom. They also do not have to be used in a pool, and could just as easily be demonstrated with someone walking to pull the weight.

[MJA]

Design Requirements Specifications

Section 1.01 Web Server and Application

The following table shows the design requirements for the server and web application. Each design requirement maps back to one or more marketing requirements, which are shown below.

Marketing Requirements	Engineering Requirements	Justification
3	Must be able to communicate at a distance of up to 50m	The server and sensor boards will be stored in separate areas. The sensor boards will be on the power racks on the pool deck, and the server in an office nearby.
2	The server's database will be fully updated within 5s after each workout repetition	The coaching staff may want to view new information between repetitions without having to wait for a long period of time.
2	The web application must be able to display data in graphs and tables that are easy to interpret	Looking at raw data values is not intuitive, and thus, the web application should provide some insight into data trends.
1, 2	The web application must be able to export data files in csv format	The coaching staff requested the ability to interpret the raw data using other software, such as Microsoft Excel.
2	Web application must be accessible by multiple devices simultaneously	While the web application will be a single page, each user should be able to control what information they are viewing independently.
7	The server must be able to support 10 active connections	Several swimmers are going to be using the power racks at the same time, and the server should accommodate updates from each user.
1	The following information will be stored and accessible: swimmer's name, date and	The coaching staff requested this information be made available.

	time or recording, weight pulled, force applied over time, distance over time.	

Marketing Requirements

1. Record force, position, weight, and time data for each individual athlete
2. Coaches and athletes can view data on their own personal devices
3. Limited user input required to operate
4. Components on the pool deck are at least IP67 rating (hardware is dust proof, and waterproof up to 1[m])
5. Physically independent from the weight machine
6. Display time per repetition during use
7. Support up to ten simultaneous data recordings

Table 1: Web Server and Application Design Requirement Specifications

Section 1.02 Sensor and Communication Board

The following table shows the design requirements for the server and web application. Each design requirement maps back to one or more marketing requirements, which are shown below.

Marketing Requirements	Engineering Requirements	Justification
4	Must be able to withstand ingress of water up to 1m of submersion for a time duration of 30 minutes	Sensor boards will be attached to weight rack on pool deck.
3	Must be able to communicate at a distance of up to 50m	The calculated max distance for the furthest workout area is 49.5m.
1	Must be able to measure weight (up to 100kg or 220lbs), as well as, distance swam, and time per repetition.	Weight and time will be displayed for the swimmers during use. Weight is set at 100kg as the coaching staff said this should be a high enough limit. The coaching staff requested distance be measured.
1, 6	The device must be able to display weight, repetition time, and distance swam to an accuracy of 1 pound, 100 ms,	The swimmers will need to see the weight used before starting each repetition. Also the swimmers would like to

	and 1 meter respectively.	see individual repetition time, in order to determine if more weight is needed or if time is consistent per multiple repetitions.
5,	The device must have a portable power source.	The device needs to be portable and freestanding.
3	The sensor board must autonomously update user information to server and web application.	The swimmers should focus on their workout, and the coaching staff on evaluating the swimmers, not operating the data system.

Marketing Requirements

1. Record force, position, weight, and time data for each individual athlete
2. Coaches and athletes can view data on their own personal devices
3. Limited user input required to operate
4. Components on the pool deck are at least IP67 rating (hardware is dust proof, and waterproof up to 1[m])
5. Physically independent from the weight machine
6. Display time per repetition during use
7. Support up to ten simultaneous data recordings

Table 2: Sensor and Communication Board Design Requirement Specifications

Accepted Technical Design

Section 1.01 Hardware Design

(a) Theory of Operation

The project's hardware will be composed of 3 main parts; a sensor board, sensors, and the web server/database. The sensor board will contain most of the hardware except for the sensors. The sensors will be attached to the main board via water tight connectors. The sensor board will then communicate the sensor data to the web server. The web server will process this information and store it locally in the database. This information can then be accessed via a web application.

[JLW]

(b) Block Diagrams

1. **Level 0**

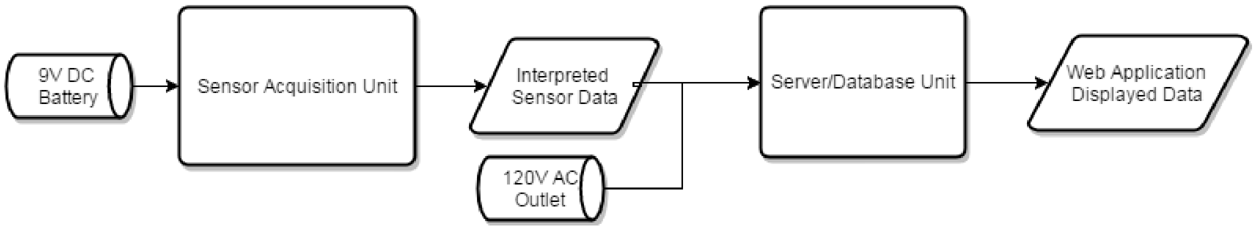


Figure 2: Hardware Level 0 Block Diagram

Module	Sensor Acquisition Unit
Designer	Tanner Daniels/Jack Wolfe
Inputs	• 9V DC Battery Pack
Outputs	• Interpreted Sensor Data
Description	The Sensor Acquisition Unit is the unit that will interpret the swimmer data and metrics and report it to the server as well as display information on a 7-segment display to be viewed by the user.

Table 3: Sensor Acquisition Level 0 Module

Module	Server/Database Unit
Designer	Tanner Daniels/Jack Wolfe
Inputs	• Interpreted Sensor Data • 120V AC Outlet
Outputs	• Web Application Displayed Data
Description	The web server will hold all user information, as well as, display it in an easy to use graphical interface. The webserver will have a 1.2GHz processor for fast computations and 2GBs of ram for fast data output and refresh rate.

Table 4: Server/Database Level 0 Module

2. Level 1

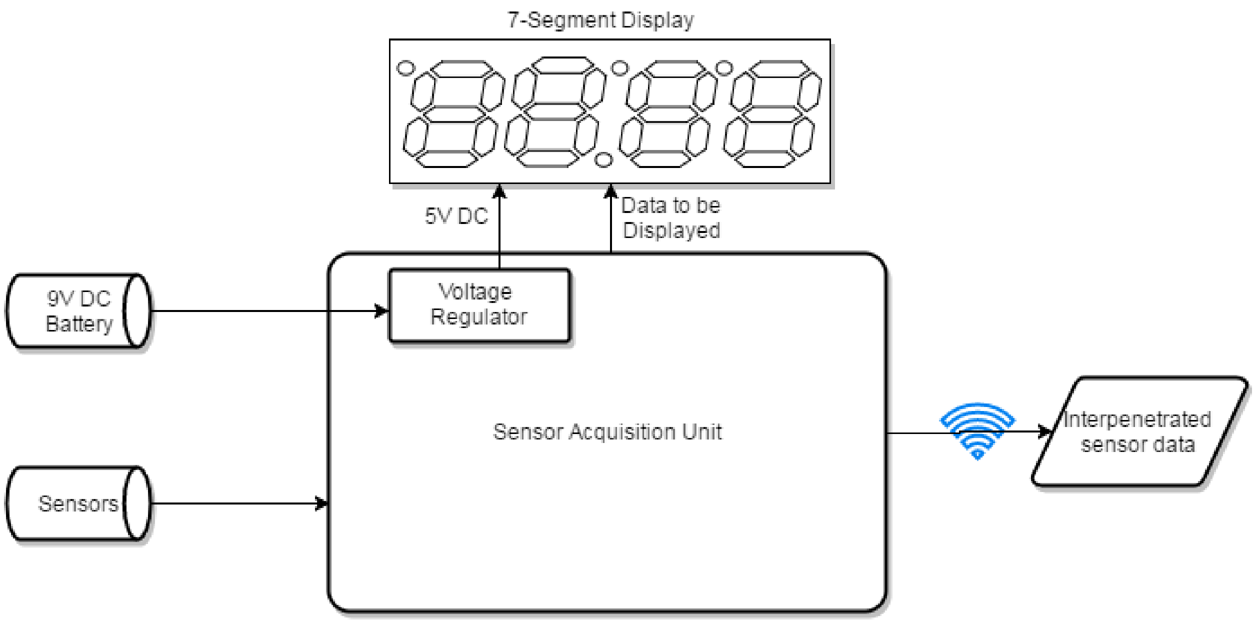


Figure 3: Sensor Acquisition Unit Level 1 Block Diagram

Module	Voltage Regulator
Designer	Tanner Daniels/Jack Wolfe
Inputs	• 9V DC Battery Pack
Outputs	• ±5V DC • +3.3 DC
Description	The voltage regulator will take in 9V from the battery pack and convert it into ±5V and 3.3V. The +5V will be used by the 7-segment display, the load cells, and the positive rail of the operational amplifier circuit. The -5V will be used for the negative voltage on the operational amplifier circuit. The 3.3V will be used to power the microprocessor and the sensors.

Table 5: Voltage Regulator Level 1 Module

Module	7-Segment Display
Designer	Tanner Daniels/Jack Wolfe
Inputs	• 5V DC • Data to be displayed
Outputs	• Displayed Information

Description	The 7-Segment display will display the weight used during the workout, as well as, the time per repetition. This allows the user to update the weight easily during the workout. The weight will be displayed with a 1[pound] resolution and the time will be displayed with a 10[millisecond] resolution.

Table 6: 7-Segment Display Level 1 Module

3. Level 2

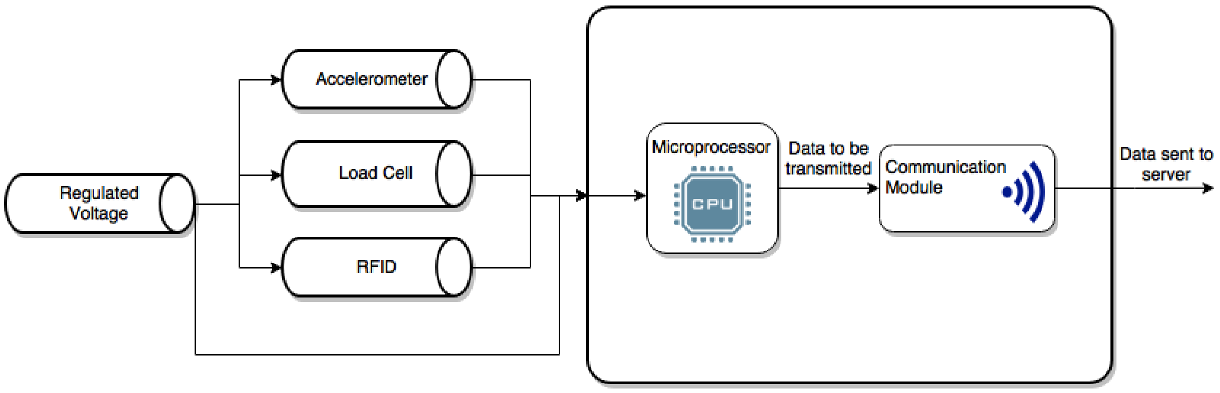


Figure 4: Sensor Acquisition Unit Level 2 Block Diagram

Module	Accelerometer
Designer	Tanner Daniels/Jack Wolfe
Inputs	• Regulated Voltage (+5V)
Outputs	• X, Y, and Z position data
Description	The accelerometer notices changes in its X, Y, and Z axes. The accelerometer then sends back this data as bits to the processor.

Table 7: Accelerometer Level 2 Module

Module	Load Cell
Designer	Tanner Daniels/Jack Wolfe
Inputs	• Regulated Voltage (±5V)
Outputs	• Sensor voltage
Description	The load cell creates a voltage change as an output depending on the weight applied to it. This weight voltage will then be amplified with a three stage operational amplifier circuit. Once amplified, it is then read by an ADC on the microprocessor.

Table 8: Load Cell Level 2 Module

Module	RFID
Designer	Tanner Daniels/Jack Wolfe
Inputs	• Regulated Voltage (+5V)h
Outputs	• RFID tag data
Description	This unit is used to offer a unique way of tracking data to each individual swimmer. A unique RFID tag is assigned to each user and when the tag is read by the RFID reader, the data for that workout session is logged in the web application under that particular user.

Table 9: RFID Level 2 Module

Module	Microprocessor
Designer	Tanner Daniels/Jack Wolfe
Inputs	• Regulated Voltage (+3.3V) • Accelerometer • Load Cell • RFID
Outputs	• Data to wireless communication module
Description	This unit contains the accelerometer, XBEE module, and load cells. The microprocessor receives data from all the sensors and sends it to the XBEE module to be sent to the web server via the ZigBee protocol.

Table 10: Microprocessor Level 2 Module

Module	Communication Module
Designer	Tanner Daniels/Jack Wolfe
Inputs	• Data to be transmitted • Regulated Voltage (+3.3V)
Outputs	• Data sent to server
Description	The main unit in the communication module is an XBEE module. The XBEE module communicates via the ZigBee protocol. This module will transmit the recorded data sent to it from the microprocessor. It will communicate with the Server/Database Unit.

Table 11: Communication Level 2 Module

(c) Sensor Board

The sensor board will hold a dsPIC33EP256MC506, this microprocessor was chosen for several reasons. This processor is rated for 70[MIPS] (million instructions per second), this allows for very fast processing as the system is to update within 5 seconds of a workout. The processor is set to program at 60[MIPS]. This processor contains 2 UART, 2 I2C, and 2 SPI communication ports. Because we are communicating with four devices, these ports allow for communication specific to each module. This processor and all connected components are shown below in Figure 5.

The sensor board contains the microprocessor, RFID module, XBEE module, and the seven segment display. The sensor board will be housed in a water tight container, which also connects to the accelerometer, and load cell via watertight wired Ethernet connectors. The microprocessor communicates to all of the sensors and receives raw data back to interpret. It communicates to the seven segment display via I2C, accelerometer via SPI, XBEE via UART, and RFID via UART. It then transmits information through the XBEE module to the web server which is being hosted by our PINE 64+. This information is then analyzed by the web server and displayed on the web application for a user to view.

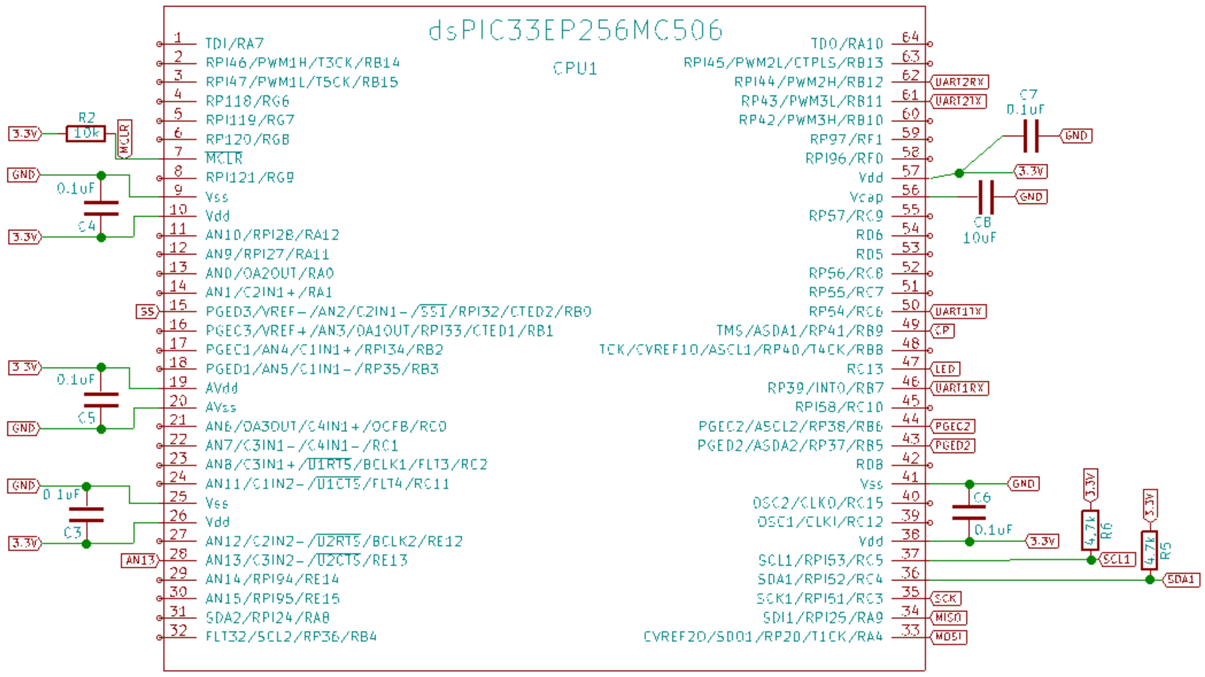


Figure 5: KiCad dsPIC33EP256MC506 circuit Schematic

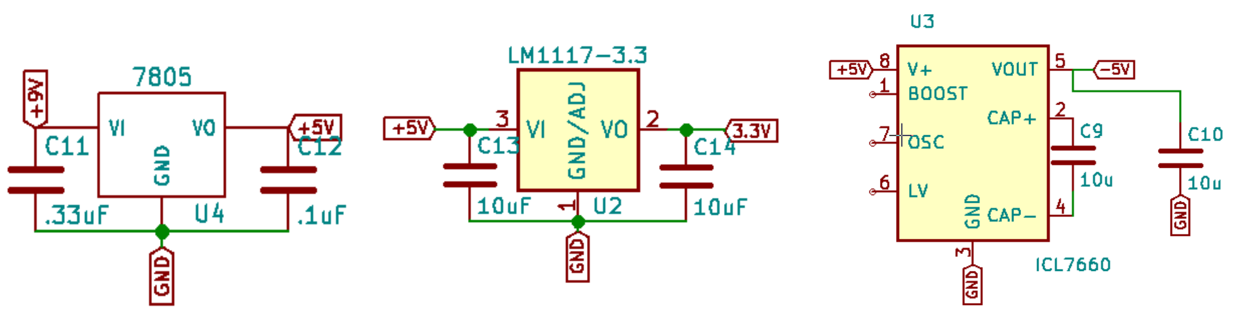


Figure 6: KiCad Power Rail Circuit Schematic

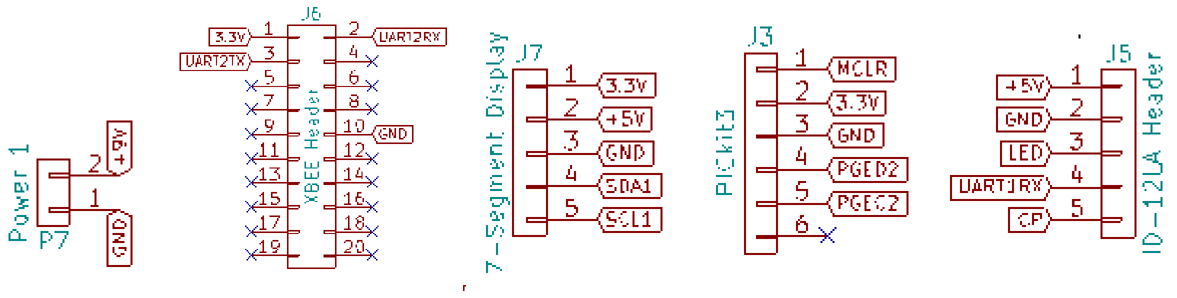


Figure 7: KiCad Connector Schematic

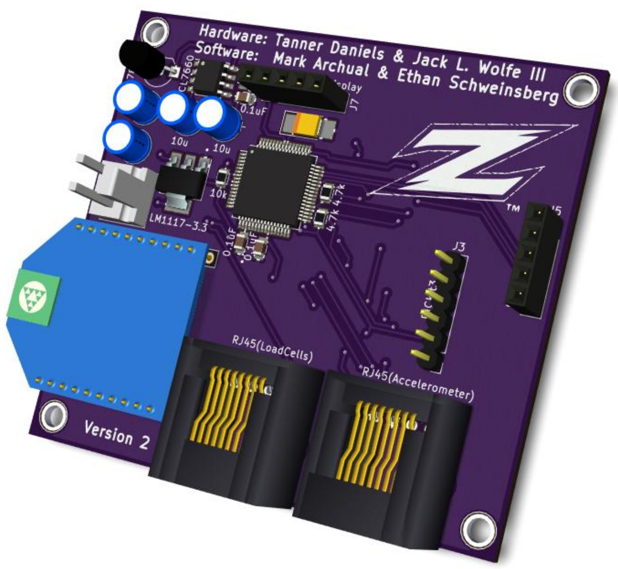

Hardware: Tanner Daniels & Jack L. Wolfe III
Software: Mark Archual & Ethan Schweinsberg
ICL7660
0.1uF
10u
10u
LM1117-3.3
4.7k
J3
J5
RJ45(LoadCells)
RJ45(Accelerometer)
Version 2

Figure 8: Sensor Board 3D Model

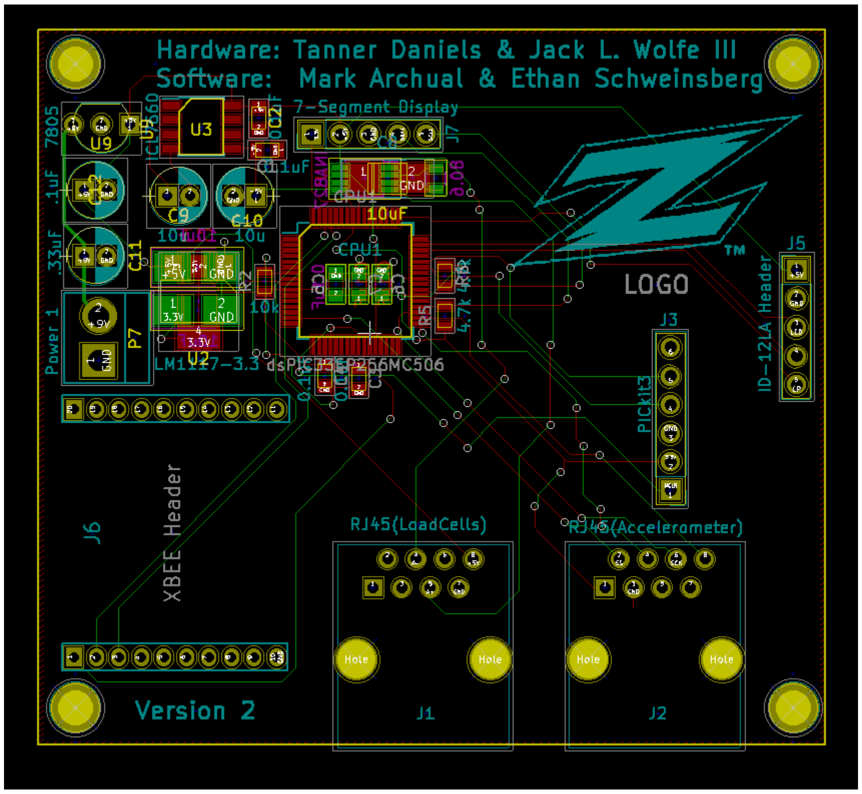


Figure 9: KiCad Footprint

[TMD and JLW]

(d) Accelerometer

The accelerometer chosen is a MC3635A 3-axis accelerometer. This particular accelerometer was chosen as the sampling frequency was from 14-1300 samples per second. For conclusive data, we determined 100 samples per second was enough for accurate data to be obtained. Testing on the sampling frequency of the accelerometer can be seen below in the software design section of the report. This particular accelerometer also is low powered in the most intensive mode, 1300[samples/s], at 36[μA]. The last major advantage of this accelerometer is it interfaces via the I2C protocol as well as SPI. This makes connecting to the microprocessor much more intuitive.

The accelerometer has many functions in this design. It will be used to determine the initial start of the swimmer when they push off the wall. As the swimmer pushes off the wall, the accelerometer will start sending non-zero values to the processor. The microprocessor will notice this during an interrupt service routine. If the interrupt source determines a certain threshold value is met, this indicates the push off the wall by the swimmer. The accelerometer will also be used to determine the position of the swimmer and how far the swimmer traveled in the pool for each repetition. This is done via a mathematical process performed with discrete integration. The raw accelerometer data is sent to the server to allow for this complex computation to occur.

The accelerometer will also be used to help assist in the weight measurement abilities of this design. Using the data from the accelerometer the tilt of the bucket can be determined. This is useful because this helps to know if the swimmers are dumping out water from the bucket, thus reducing the weight. This will initiate a “re-weigh” process to the microcontroller so the logged data sent to the web application is as accurate as possible. More details on the software aspects of the accelerometer are covered in detail in the software design section of the report.

[TMD and JLW]

(e) Weight Measurement

Measuring the weight of the swimming buckets and weight racks is performed by a set of load cells. Load cells are essentially resistors that when a load is applied, either by compressing or expanding the unit, a change in resistance value is achieved. A Wheatstone bridge circuit is constructed with the 4 load cells branches of the bridge, shown in Figure 10. This configuration is used to allow a zero voltage difference measured across the branch points when no load is applied. As the load is increased on one branch, therefore altering its resistance value, the difference in the branches becomes higher and the weight can be determined. The weight applied and voltage change to the Wheatstone bridge create a linear relationship. An example of the linear graph is shown below in Figure 10.

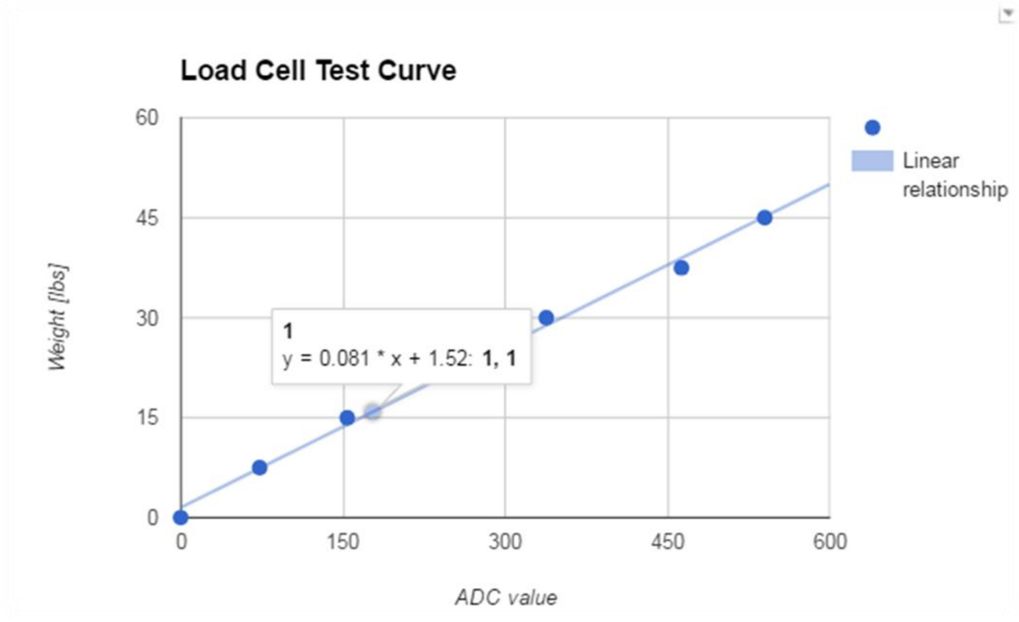


Figure 10: ADC voltage vs. applied load

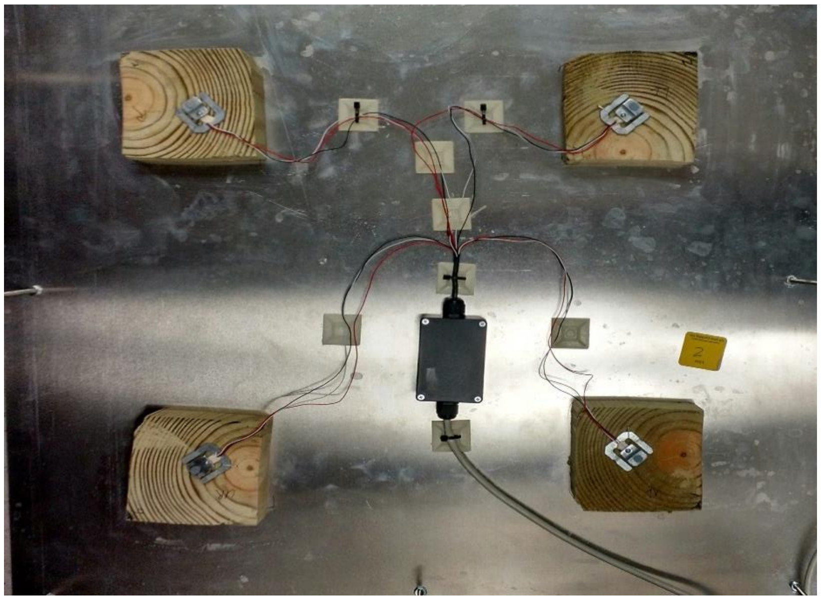


Figure 11: Load cell configuration for scale

The full Wheatstone bridge circuit can be seen below as

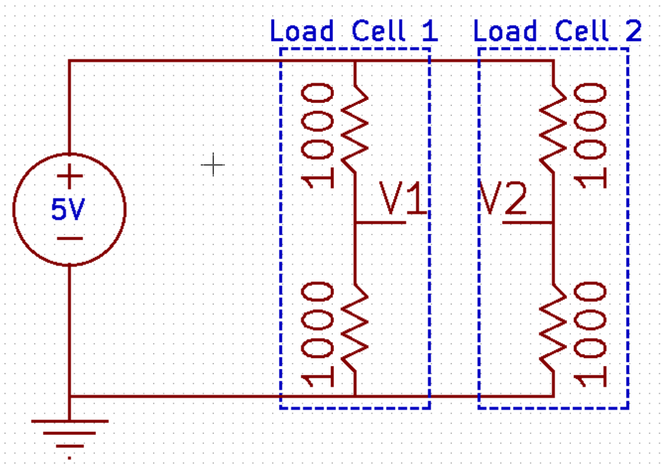


Figure 12: Load Cell Full Wheatstone Bridge Configuration

To interpret the data from the Wheatstone bridge, an amplification circuit needs to be constructed, as the voltage change is in the order of microvolts. The amplification circuit and gain calculations are shown below in Figure 13Figure 13: Amplification circuit. We chose to have a gain of about 538.3$[\frac{V}{V}]$, as this coupled with our microprocessors 12 bit ADC gives us accuracy down to 1.5μV.

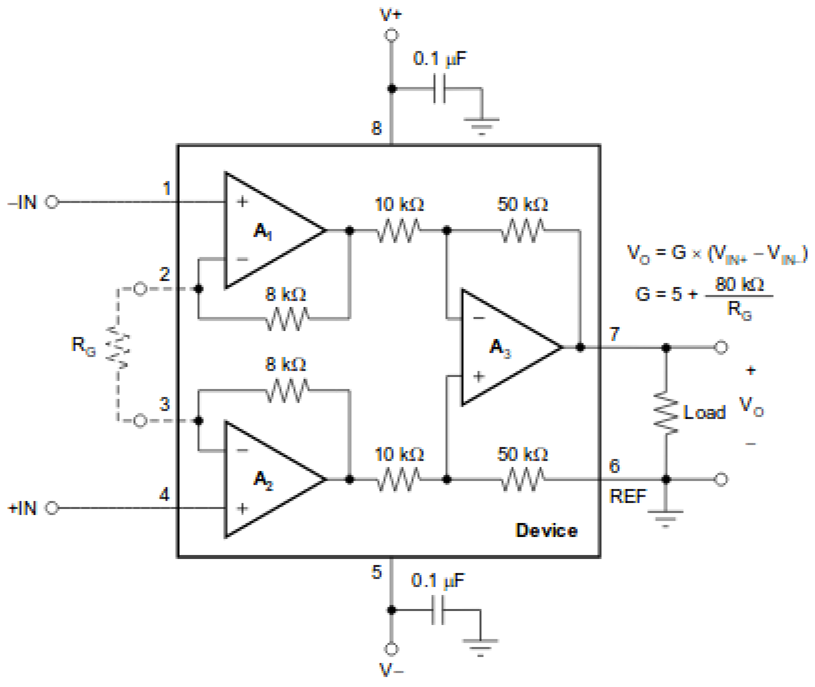


Figure 13: Amplification circuit

An instrumentation amplifier was determined to be the best to eliminate noise first and then amplify the load cell voltage readings. The first stage is a Voltage Buffer, this stage has 0 gain but is used to aid in eliminating DC noise. The next stage is a Differential Amplifier, the gain is determined by the equation: $Gain_{DA} \;=\; 5+\frac{80k\Omega}{R_G}$

The R_G chosen was a 150Ω resistor as this resulted in a gain of approximately 538.33$[\frac{V}{V}]$.

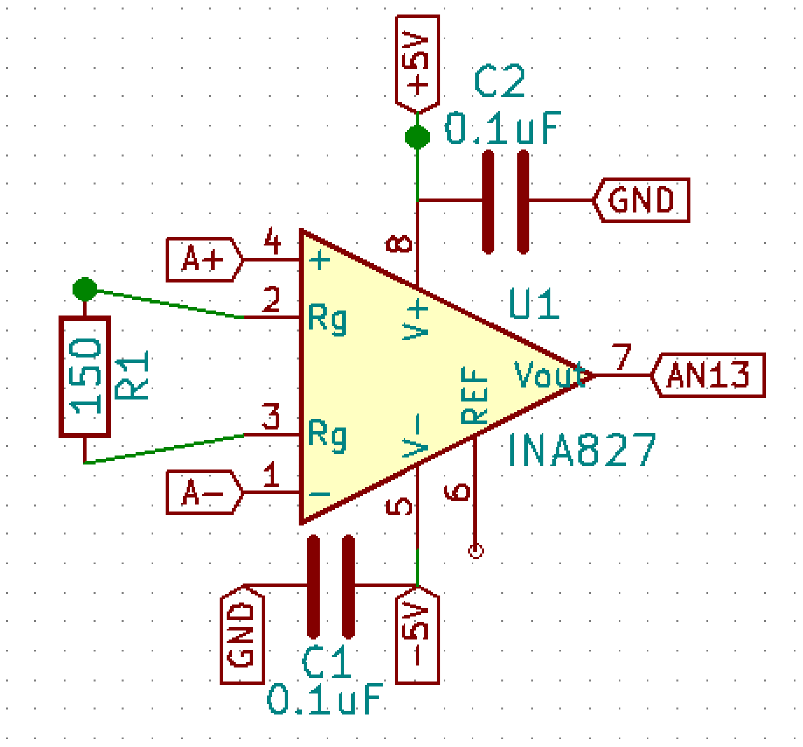


Figure 14: KiCad Amplification Circuit Schematic

In order to properly weigh the buckets weight, a platform was designed and constructed. The platform consists of a bottom plate, the middle area containing 4 load cells, and a top plate. The design creates a heavy duty scale to weigh the buckets. The material chosen was 6061 grade aluminum as it is very resistant to corrosion and light weight while remaining sturdy enough to support the weight. The scale can also be fixed to the bucket bracket if desired to allow the swimmers to not have to install it before every swim.

[TMD]

(f) RFID System

A Radio Frequency Identification system is implemented to allow for individual swimmers to have the ability of scanning into a certain machine using a unique RFID tag. Each tag will be assigned to an individual swimmer. This will then log the data recorded at that machine to a certain swimmers profile, inside the database, so the user and coach may be able to track their progress using the web application.

The RFID reader chosen to be used was an ID-12LA. This module has an integrated antenna, range of about 2 inches, and communicates via UART. The main reason this sensor was chosen was because it has a small compact design making it simpler to implement, as well as, making it water resistant.

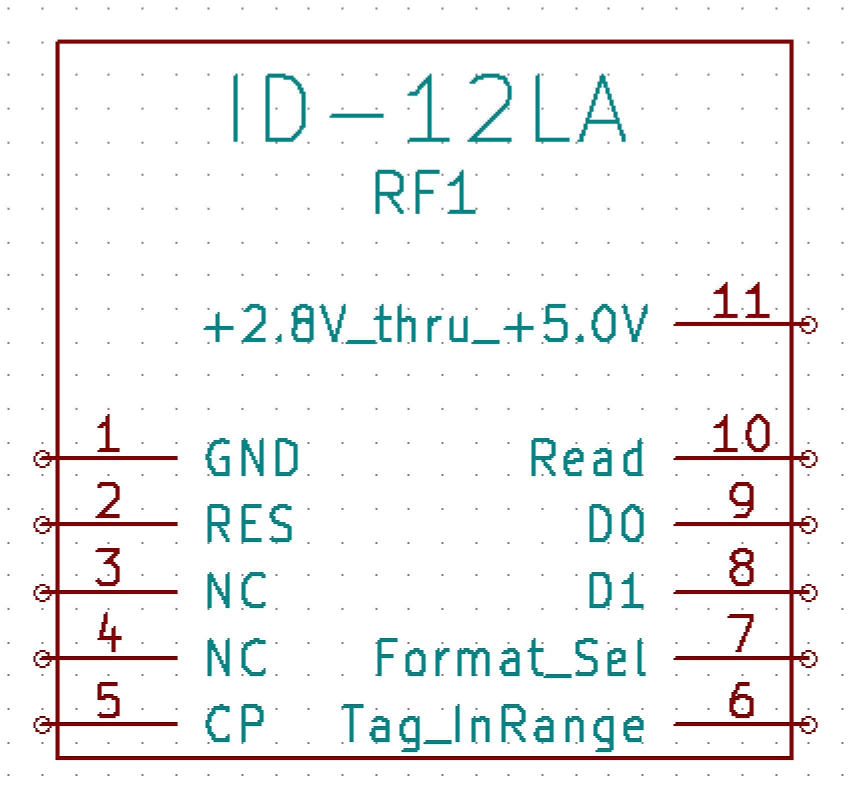


Figure 15: KiCad RFID Circuit Schematic

[TMD and JLW]

(g) Communication Module

The communication module is in charge of wirelessly transmitting data to the web server to be displayed on the web application. This module will be sent information from the microcontroller such as the swimmers identification, weight of the bucket or rack, start and stop times, and/or distance data. Testing was performed with Bluetooth connectivity but proved to have undesirable results. The connection power at the distance the machines would be placed in relation to the server was seen as undesirable and unreliable. A figure of the test is shown below as Figure 16. Another drawback of Bluetooth is that it can only support up to 7 active connections.

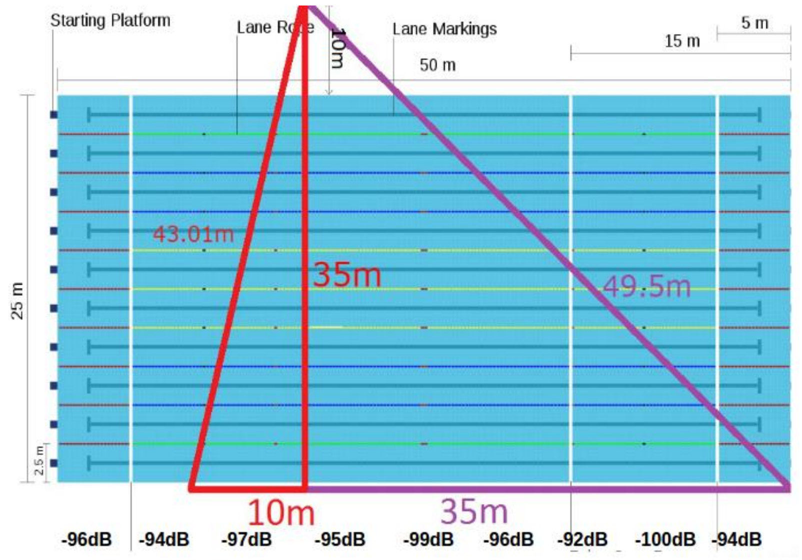


Figure 16: Bluetooth Connectivity Testing at Pool Deck

This figure shows the estimated distances of the pool deck in relation to estimated areas for the server and clients. The max wireless communication distance is about 50m total. This is shown in the calculations below. The formula for a right angle triangle is given:

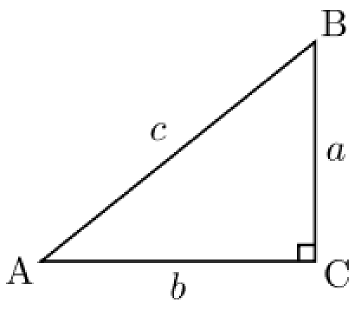


Figure 17: Right Triangle Diagram

$$a^2 + b^2 = c^2$$

Based on our distances of 25m, 35m, and 10m the following triangle hypotenuses are found.

$35^2 + 10^2 = c^2 \rightarrow c = \sqrt{35^2 + 10^2} \approx 43.01m$

$35^2 + 35^2 = c^2 \rightarrow c = \sqrt{35^2 + 35^2} \approx 49.5m$

The communication protocol decided to be used was ZigBee. ZigBee allows up to 255 active connections. This allows the system to be expanded in the future, as well as, connect to the

26 desired machines at the current time. ZigBee also allows for long range communication in the magnitude of miles, thus creating a reliable connection between the server and power racks.

The XBee S2 module was used to utilize the ZigBee communication protocol. This XBee module is a through-hole device, low power (max 40mA), and communicates at minimum of 300ft.

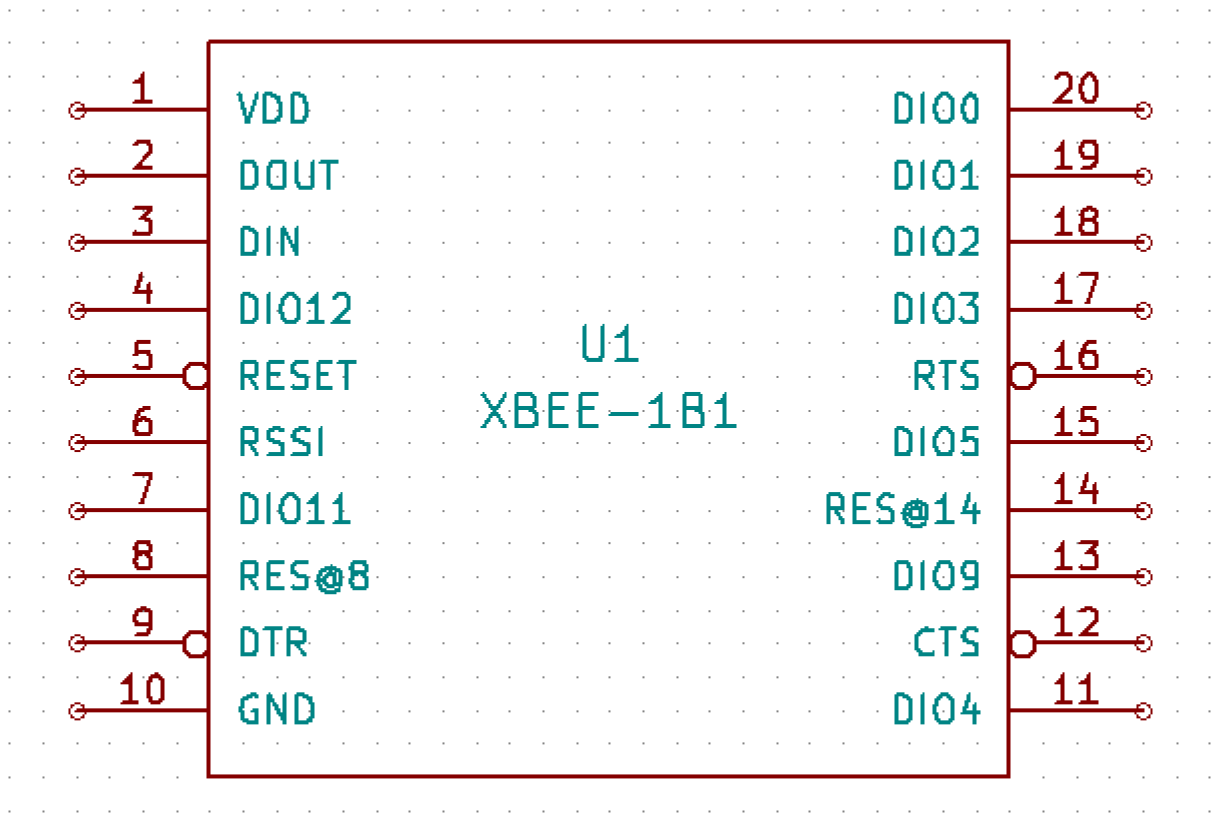


Figure 18: KiCad XBEE Circuit Schematic

[JLW]

(h) Display Module

A seven segment display was chosen to display the information to the swimmers in real time. The seven segment display chosen was the Adafruit 1.2” 4-Digit 7-Segment display with an I2C backboard. This display was chosen as it has a very high visibility and communicates via I2C. The display draws approximately 50[mA] when set to full brightness. The brightness was minimized to save power and reduced to approximately 8-10[mA]. The information being displayed will be the weight, when in weight reading mode, as well as the overall time the swimmer was swimming, during a repetition. When in weight reading mode the display will show the weight of the bucket or weight rack to the nearest 1 pound interval. While displaying the swimming time the display will show the swimmer's time in seconds and milliseconds.

[TMD & JLW]

(i) Overall Power Consumption

Calculating the overall battery life of the system was done by using the “worst case” current draw of each sensor. This gives a minimum range of battery life values when 1 lithium-manganese Dioxide 9V battery. The battery suggested to use is the Energizer L522 as it has 650[mAh] of storage. The first equation is for the “worst case scenario” meaning that every sensor is in high power mode and on at the same time. This mode will never happen in standard operation but is shown for reference. The second equation is of the standard operation current consumptions. The values for that equation were achieved by dimming the display as it was too bright, setting the accelerometer in a low power mode, and generally not demanding too much from the other sensors. This rating could have been improved if the XBEE module was configured to enter a sleep mode until a signal is received as it draws a lot of current all the time in its current state. Another improvement would have been to acquire a less demanding processor as the dsPIC33EP256MC506 draws a lot of current in its standard setup.

$$BatteryLife_{Min} = \frac{650[mAh]}{\sum I_{Sensors}}$$

$$\Rightarrow \frac{650[mAh]}{I_{Microprocessor} + I_{RFID} + I_{Accelerometer} + I_{XBEE} + I_{LoadCells} + I_{Display}}$$

$$\frac{650[mAh]}{60[mA] + 35[mA] + 36[\mu A] + 55[mA] + 1.25[mA] + 500[mA]} \Rightarrow \frac{650[mAh]}{651.286[mA]}$$

$$\Rightarrow 0.998025445[hours]$$

$$BatteryLife_{Standard} = \frac{650[mAh]}{\sum I_{Sensors}}$$

$$\Rightarrow \frac{650[mAh]}{I_{Microprocessor} + I_{RFID} + I_{Accelerometer} + I_{XBEE} + I_{LoadCells} + I_{Display}}$$

$$\frac{650[mAh]}{55[mA] + 18[mA] + 36[\mu A] + 40[mA] + 1[mA] + 8[mA]} \Rightarrow \frac{650[mAh]}{122.036[mA]}$$

$$\Rightarrow 5.326297[hours]$$

[TMD]

(j) Bucket Weight

Using all 4 load cells in opposite corners of the scale, shown in Figure 11 above, the weight is evenly distributed. We chose 4 load cells rated at 50[kg] each. When 4 load cells are used, the maximum weight is 200[kg], or 440.925[lbs.]. The derivation of calculating whether the chosen load cells were adequate or not is shown below. It can be seen from the calculation that the max weight of a full bucket of water that the swimmers use is well within the range of these 4 load cells.

25 gallon bucket filled full of water:

Weight of water per gallon ≈ 8.36 [pounds/gallon]

25*8.36 = 209[lbs.]

$$25[gal] * 8.36[\frac{lbs}{gal}] = 209[lbs]$$

[TMD]

Software Design

Section 1.01 Theory of Operation

(b) Firmware

The firmware consists of embedded C code on the sensor board for the dsPIC33EP256MC506 microcontroller. Its purpose is to interface with the accelerometer and RFID module, perform signal processing, handle wireless communication with the server, control a seven-segment display, and perform analog to digital conversion on the load cell signal. In addition, the firmware will implement algorithms to sense the start time of the swimmer, the stop time of the swimmer, and weight changes of the power rack.

The firmware is implemented as a state machine. This was chosen because it works well with how the user interacts with the system. Also, state machines are good ways to implement efficient code. The software has many different states, but they can be boiled down to two basic ones: when the athlete is swimming and when the athlete is waiting to swim. Also, this code implements algorithms to determine if the swimmer has pushed off the wall to begin swimming and to detect when the swimmer is adding or removing weight from the power rack. In the "waiting" state the software continuously looks for changes in the weight and the RFID signal. Figure 19 shows the functionality of this state in detail.

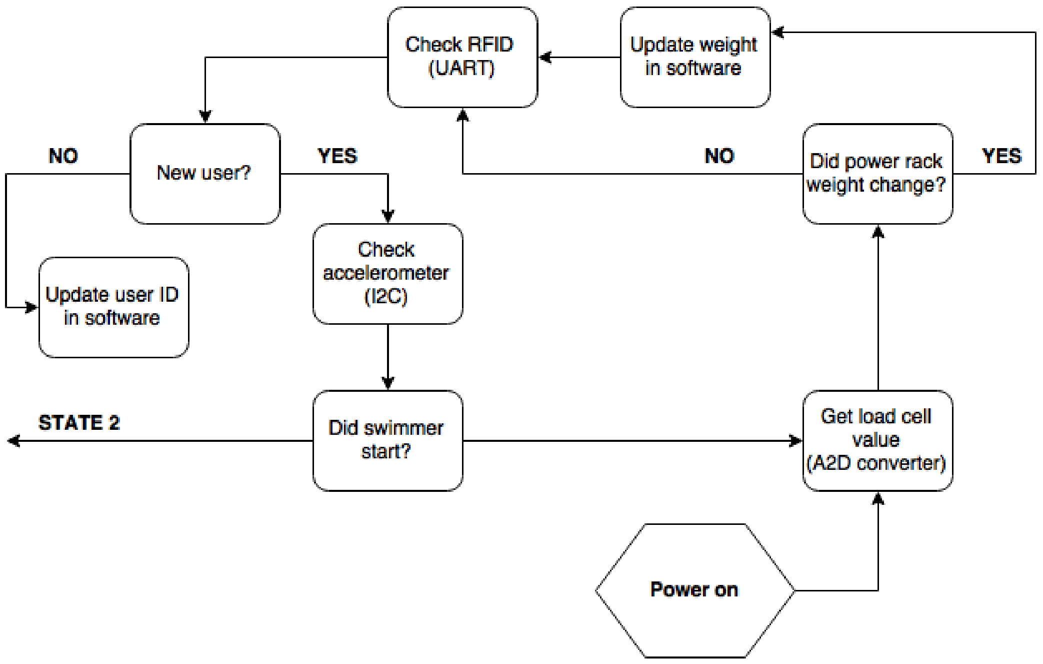

Check RFID (UART)
Update weight in software
NO
New user?
YES
NO
Did power rack weight change?
YES
Update user ID in software
Check accelerometer (I2C)
STATE 2
Did swimmer start?
Get load cell value (A2D converter)
Power on

Figure 19: State 1 Firmware Flowchart

During the ‘swimming’ state, the firmware will continuously read acceleration values and transmit them to the server until it senses that the swimmer has stopped moving. Figure 20 shows this functionality in detail.

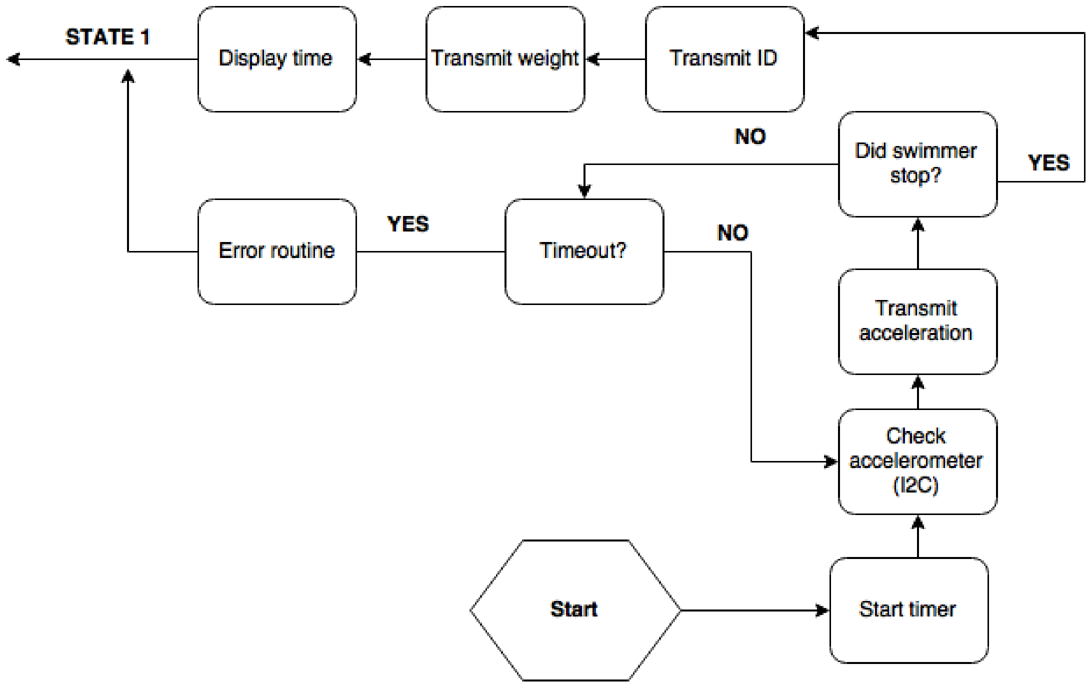


Figure 20: State 2 Firmware Flowchart

Although the flowcharts show the processing happening in a linear manner, the firmware will be interrupt driven. This ensures that no single task occupies the processor's execution time for too long. The dsPIC33EP256MC506 has 5 timers which can be used as interrupt sources. One main 10ms interrupt will drive the state machine shown above. Another interrupt will be attached to the USART bus to capture RFID data whenever a swimmer scans their card. Below is the code that shows the possible states that the system can be in.

```
void system_tasks( void )
{
    switch ( sys_state )
    {
        case RESET:
            sys_state = systemStateReset();
            break;

        case SCAN:
            sys_state = systemStateScan();
            break;

        case WEIGH:
            sys_state = systemStateWeigh();
            break;

        case BLINK:
            sys_state = systemStateBlink();
            break;

        case WAIT:
            sys_state = systemStateWait();
            break;

        case ACCEL:
            sys_state = systemStateSendAccel();
            break;

        case WEIGHT:
            sys_state = systemStateSendWeight();
            break;

        case SWIM:
            sys_state = systemStateSwim();
            break;

        case END:
            sys_state = systemStateEnd();
            break;

        case RETURN:
            sys_state = systemStateReturn();
            break;

        case ERROR:
            break;
    };
}
```

One major design consideration is the accuracy at which the firmware will be able to detect when the swimmer begins and ends each exercise. An accelerometer, placed on the moving portion of the power rack, will be used to accomplish this task. It is expected that there will be a major jump in acceleration at the beginning of the swim and a dampening as the swimmer reaches their maximum distance. Sample acceleration data was collected to verify that this is the case. The section below, *Using Acceleration Data to Determine Athlete Position,* describes the data acquisition process in more detail. Figure 21 shows data taken from the beginning of a swim.

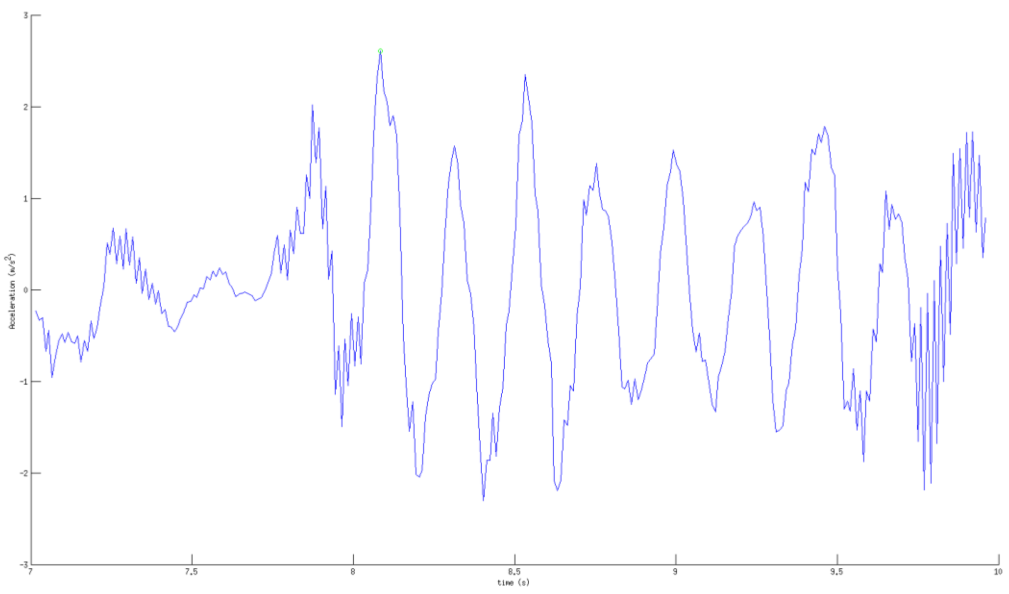


Figure 21: Acceleration Values at the Beginning of the Swim

There is a clear spike in acceleration around the 10s mark, which is the time that the swimmer pushed off of the wall to begin exercise. The peak is a value of $2.6039 m/s^2$. Similarly, it can be shown that the acceleration clearly dampens at the end of the workout. See Figure 22 for a plot of the acceleration at the end of a swim.

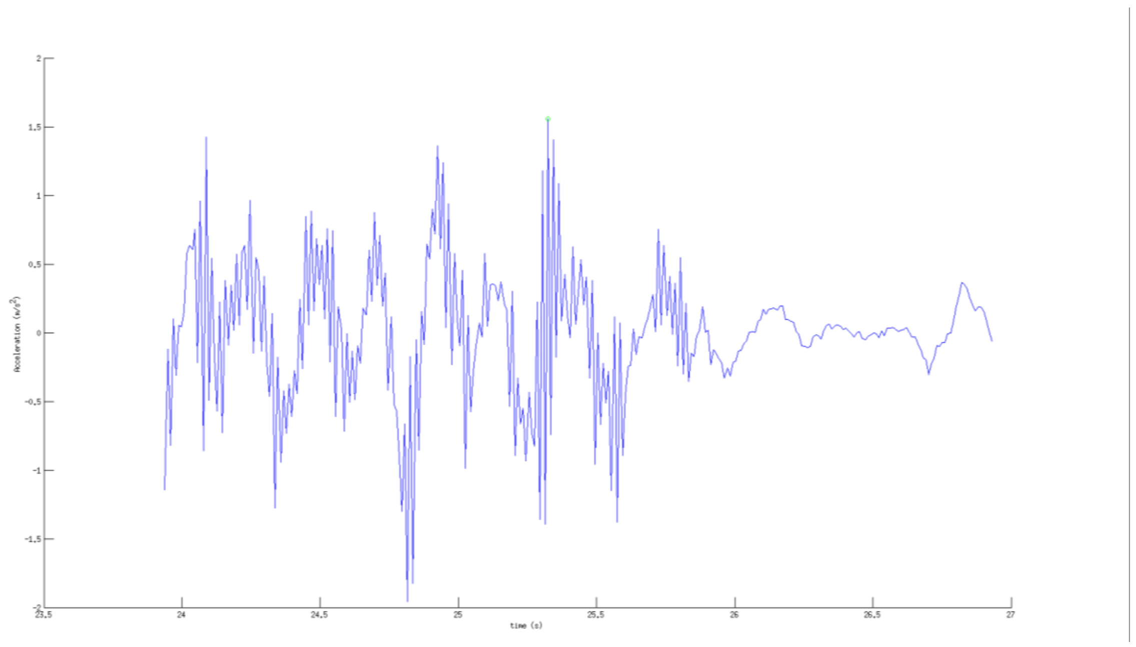


Figure 22: Acceleration Values at the End of the Swim

This dampening happens at about 30s, which is when the swimmer starts to return to the wall. A detection algorithm can be implemented in real time on the microcontroller to detect these scenarios. This algorithm will calculate a running variance of the data. A high variance corresponds to a high level of movement, whereas a low variance corresponds to a low level of movement. A ratio of different types of variances can be used to determine if the bucket is at steady state or not. The algorithm is shown below.

```
float ALG_Calculate_R(short a)
{
    /* calculate the R value */
       xf = L1*a + (1-L1)*xfm1; //running average
       vi = L2 * (a - xfm1) * (a-xfm1)  + (1-L2)*vim1; // running variance (1)
       si = L3 * (a - am1) * (a-am1) + (1-L3)*sim1; // running variance (2)
       R = ((2-L1)*vi) / si;

    /* update previous values */
    xfm1 = xf;
    sim1 = si;
    vim1 = vi;
    am1 = a;
    Rm1 = R;

    return R;
}
```

(k) Using Acceleration Data to Determine Athlete Position

As per marketing requirements, the system needs to determine the distance the swimmer swam per repetition. This can be accomplished by using acceleration data from an accelerometer placed on the moving portion of the power rack. The moving portion of the power rack will be a bucket or weights depending on the type of machine. This section will focus on the bucket, although the same theory can be applied to the weights. The position of the bucket can be determined by the following equations:

$$v(t) = \int a(t)dt$$
$$x(t) = \int v(t)dt$$

where *a(t)* is continuous time acceleration and *x(t)* is continuous time position. However, since a digital accelerometer will be used, the above integrals will have to be approximated with numerical integration techniques. The data returned from the accelerometer will be an array of data points, and not a continuous function. Thus, integration will have to be repeated several times depending on the technique that is applied.

To test this technique, data sets were obtained by using a smartphone to record the acceleration of a bucket during five separate workouts. Matlab provides services for data acquisition using the Matlab Mobile App. The Matlab Mobile App can be downloaded to an Apple or Android smartphone and then used to access the phone's sensors suite. Notably, the data acquisition can be controlled remotely through a connection between an instance of Matlab Mobile and Matlab running on a laptop or desktop computer.

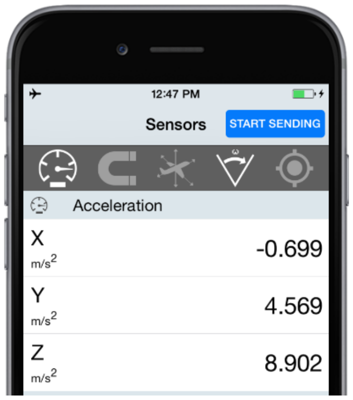


Figure 23: Matlab Mobile App for Data Acquisition

For the purposes of this project, the Matlab Mobile suite was utilized to record accelerometer data from an iPhone 6S plus at 100 Hz. The phone was oriented such that the z-axis would correspond to the vertical acceleration of the weight being pulled. The data acquisition process would be started and stopped remotely from a laptop so that the phone would not be physically interfered with during testing. At the end of each test, the mobile app would automatically save the workspace variables from the test to a .mat file on the user's Matlab Drive account. This data could then be easily reloaded into Matlab and the workspace variables directly manipulated for the different integration techniques. A backup recording was also taken using the Android App 'Physics Toolbox' on a Motorola Moto X. Figure 245 shows how the data acquisition was controlled in Matlab:

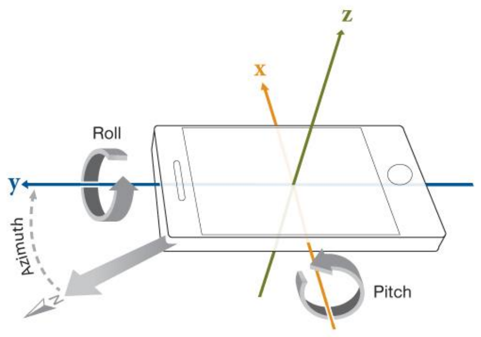


Figure 24: Phone Orientation for Data Acquisition

Start Script	**Stop Script**

`m = mobiledev; %Creates a mobile dev object`
`if(m.Connected)`
`    m.SampleRate = 100;`
`    m.Logging = 1; %Starts the recording`
`    disp(m); %Outputs the recording session information`
`end`
	`time_stamp = m.InitialTimestamp; %Get the timestamp of when the recording began`
	`m.Logging = 0; %Stop recording`
	`[a,t] = accellog(m); %a is acceleration data, t is time increments`
	`filename = strcat('test_', time_stamp(1:20)); %creates a unique filename`

`m.SampleRate`	`save(filename); %Save all the workspace variables to a file`
	`clear m; %clears the mobiledev object from memory so you can start a new session`

Figure 25: Matlab Data Acquisition Scripts

A total of 6 repetitions were recorded. For the first four workouts, the swimmer swam out to a predetermined distance of 7.5692 meters. For the last two workouts, the swimmer swam out to a predetermined distance of 15.113 meters. Videos were recorded of each workout to use as reference. The known distances were used to verify the integration was working properly. Five different methods of numerical integration were applied to the each of the five data sets, using two different filters for each combination via Matlab. These methods include: Midpoint, Trapezoid, Simpson’s, Simpson’s ⅜, and Boole’s rule. The final, or peak, position calculated from each integration technique was compared to the measured position of the swimmer to determine accuracy. Table 12 and Table 13 shows the relative errors of each integration technique.

Method	**Error 298_1**	**Error 298_2**	**Error 298_3**	**Error 298_4**	**Error 595_1**	**Error 595_2**	**Avg Error**
Boole's	0.040	0.094	0.207	0.033	0.540	0.089	0.167
Midpoint	0.052	0.098	0.208	0.046	0.531	0.074	0.168
Trapezoid	0.040	0.106	0.172	0.003	0.532	0.163	0.169
Simpson's	0.186	0.407	0.514	0.379	0.027	0.366	0.313
Simpson's 3/8	0.028	0.091	0.189	0.039	0.569	0.136	0.175

Table 12: Relative error results for second order Butterworth Filter

Error Analysis with a fourth order Butterworth Filter

Method	**Error 298_1**	**Error 298_2**	**Error 298_3**	**Error 298_4**	**Error 595_1**	**Error 595_2**	**Avg Error**
Boole's	0.110	0.145	0.287	0.128	0.435	0.011	0.186
Midpoint	0.091	0.126	0.255	0.101	0.470	0.042	0.181
Trapezoid	0.081	0.136	0.219	0.066	0.475	0.119	0.183
Simpson's	0.126	0.340	0.478	0.272	0.235	0.422	0.312
Simpson's 3/8	0.107	0.146	0.268	0.134	0.446	0.042	0.191

Table 13: Relative error results for fourth order Butterworth Filter

From these results, it is determined that Midpoint is the best overall integration technique for this application. Additionally, it is the simplest integration technique to implement. Midpoint numerical integration is accomplished by taking the middle point between two consecutive values of a function and multiplying it by the difference in index. This creates a "box" whose area is the estimated area under the curve of the function.

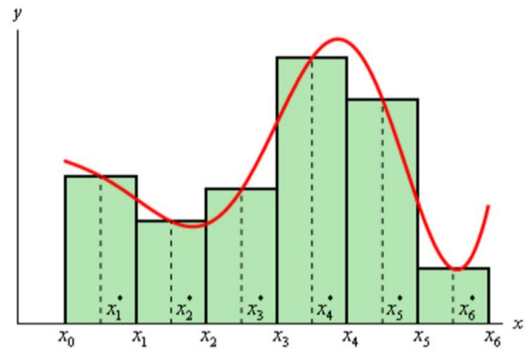


Figure 26: Midpoint Integration Example

The area of each box becomes a data point in the integrated function. In this application, the y-axis is acceleration or velocity and the x-axis is time.

Once the data for the distance the bucket traveled is determined, the displacement of the swimmer can be found using a simple linear relationship of the form:

$$y = m * x + B$$

The coefficients of this polynomial will be determined using the linear least squares approximation (see code below). The linear least squares approximation assures that the coefficients of the polynomial have been determined, such that minimal error exists for the residuals over the points that the polynomial will be evaluated. The data points used for this model will be taken from measurements taken in the pool that describe the bucket's displacement relative to the swimmer's position in the pool. These data points for the swimmer's distance can then be modeled using another polynomial approximation with the linear least squares method. Rather than storing all of the individual acceleration values for the bucket, integrating twice, and then applying the linear relationship described above, the swimmer's distance can be found be simply evaluating a polynomial over the appropriate time interval. This second approximation will help greatly simplify the database in both complexity and size.

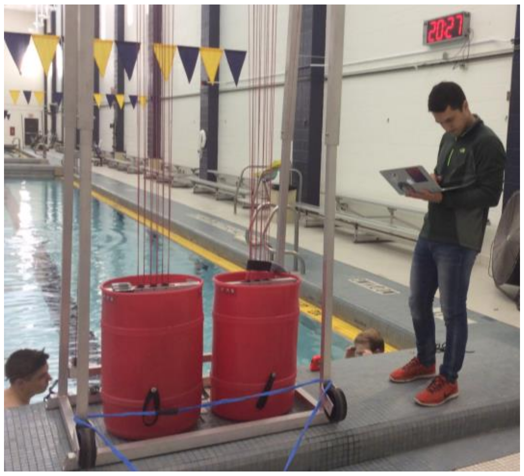


Figure 27: Team Doing Data Acquisition at Pool

```
function C = lspoly(x,y,M)

n = length(x);
F = zeros(n,M+1);
for k = 1:M+1
   F(:,k) = x'.^(k-1);
end
A = F'*F;
B = F'*y';
C = A\B;
```

Figure 28: Matlab Code for Polynomial Approximation

Figure 29 shows the relationship between the swimmer's displacement in the pool and the bucket's height using the linear least squares approximation method.

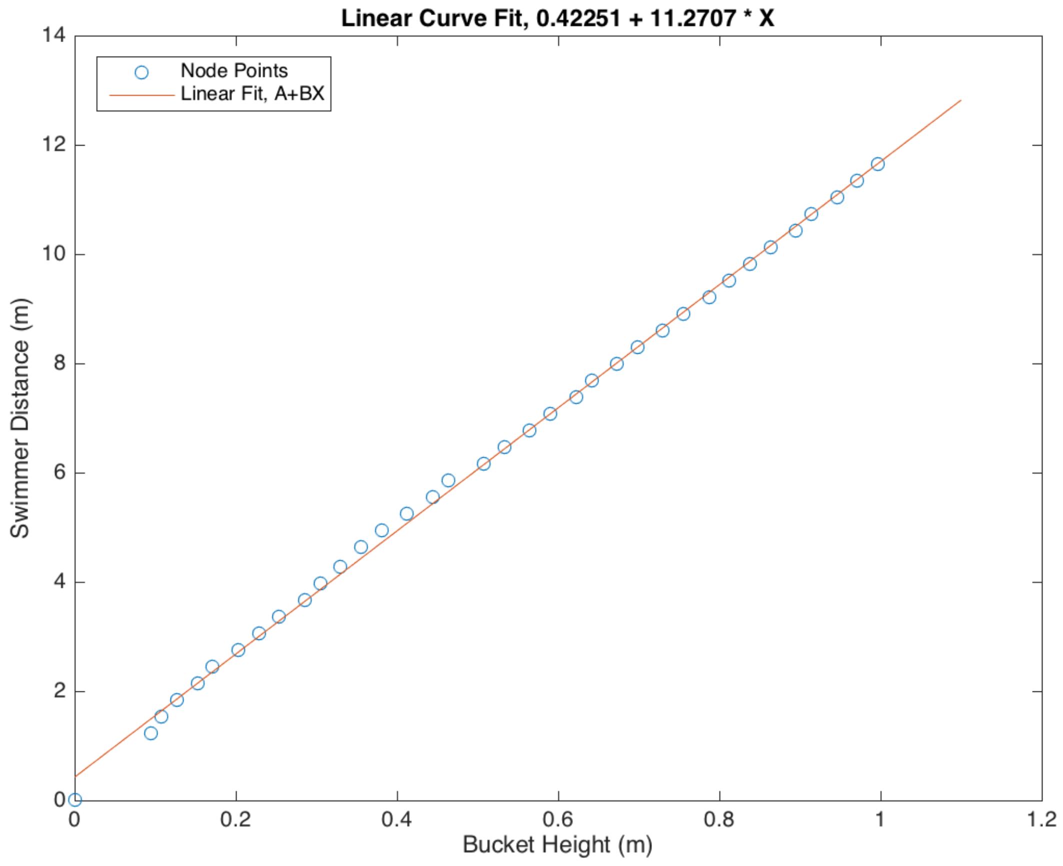


Figure 29: Linear curve fit mapping bucket height to swimmer distance

This fit yielded an R^2 value of 0.9986. This relationship, $0.42251 + 11.2707 * x$ was applied to a data set of the bucket's height over time. The swimmer swam to a predetermined distance of 7.511 meters. The bucket height was found using the Midpoint integration technique described above from a recording of the bucket's acceleration. See Figure 30 for results.

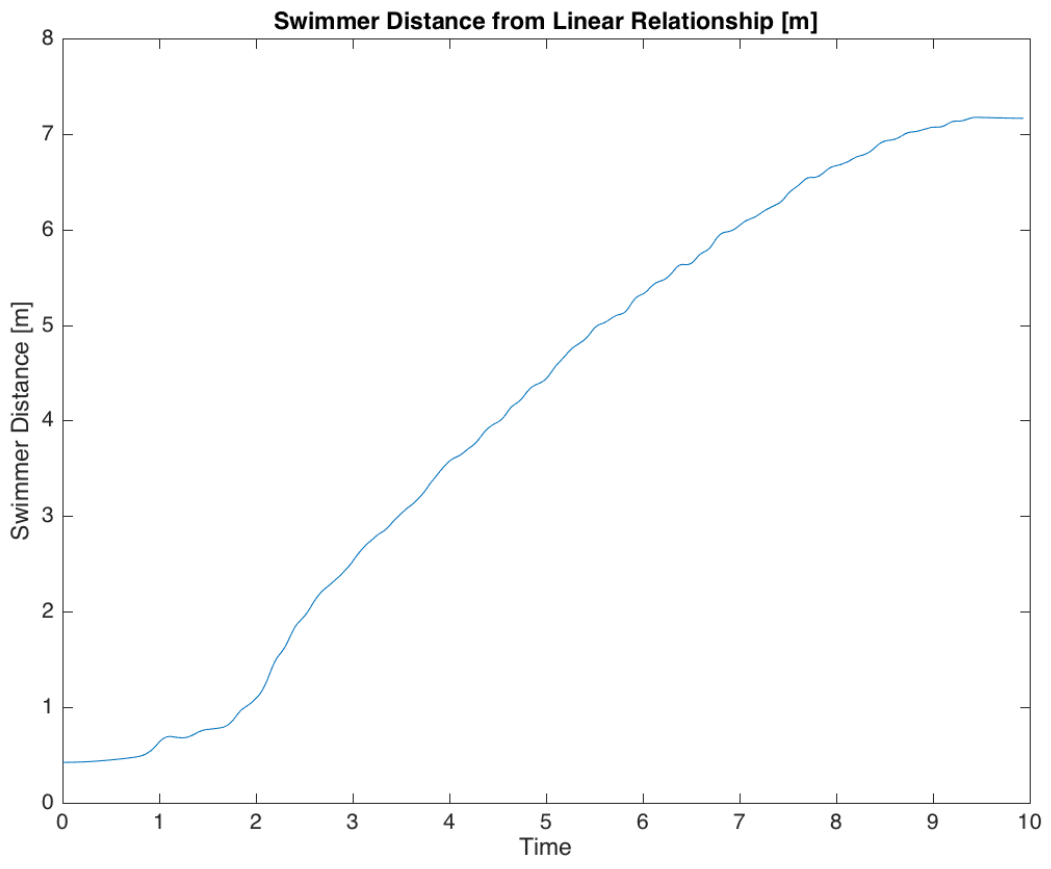


Figure 30: Swimmer distance curve after apply linear relationship

This distance curve can then be approximated by again applying the linear least squares method to find a 10th order polynomial approximation to the data.

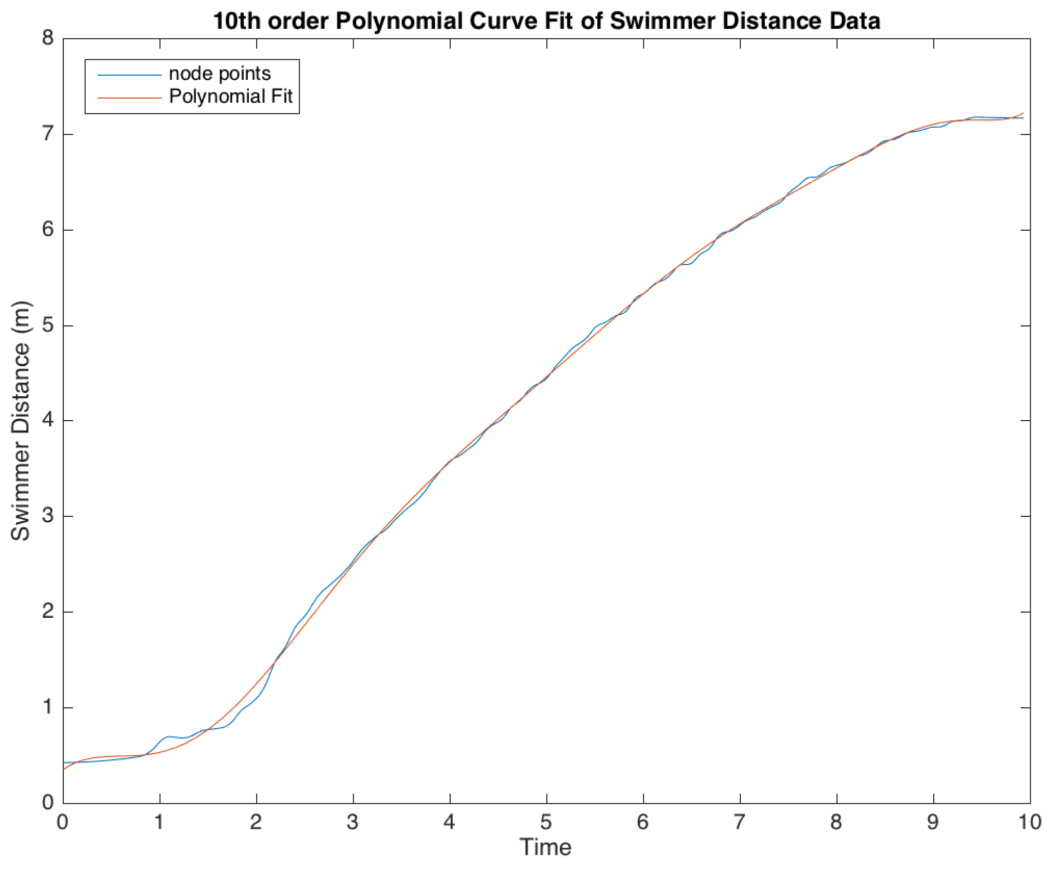


Figure 31: 10th order polynomial approximation of swimmer distance curve

The regression analysis for this fit showed an R^2 value of 0.9996. Thus, the swimmer's position can be determined accurately and this position can be modeled as a 10th degree polynomial. For calculations on how much this polynomial will reduce the size of the database, please refer to the section entitled, 'Database'.

(l) Filter design

As stated above, better integration results were accomplished when applying a digital low-pass filter to the acceleration data. This removes high-frequency noise that is not associated with the movement of the swimmer, but the behavior of the power rack system. The noise can be seen in Figure 32, a plot of acceleration data that was collected during a trial.

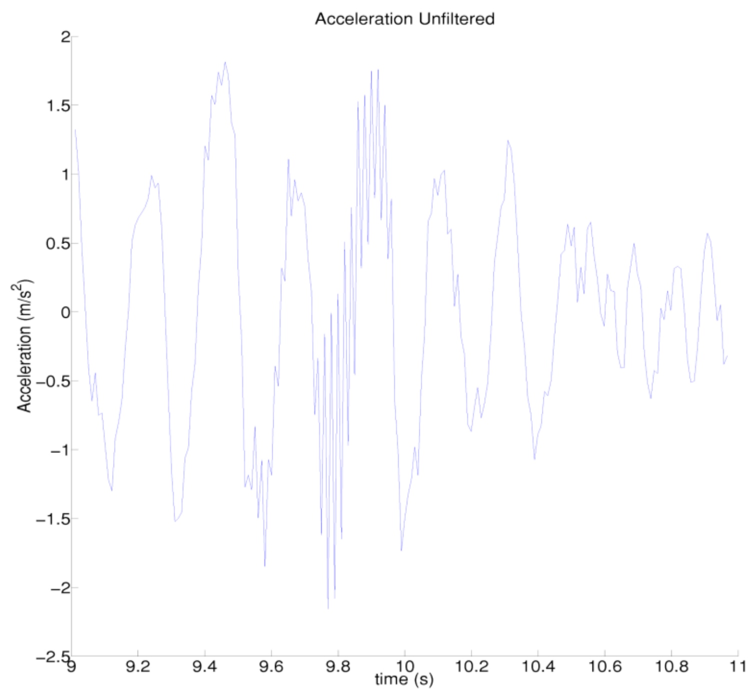


Figure 32: Unfiltered accelerometer data

A frequency spectrum of the entire data set was obtained by taking the discrete Fourier transform of the signal. This shows a cluster of very low frequencies and a cluster of very high frequencies. It is assumed that the low frequencies correspond to the movement of the swimmer and the high frequencies correspond to noise.

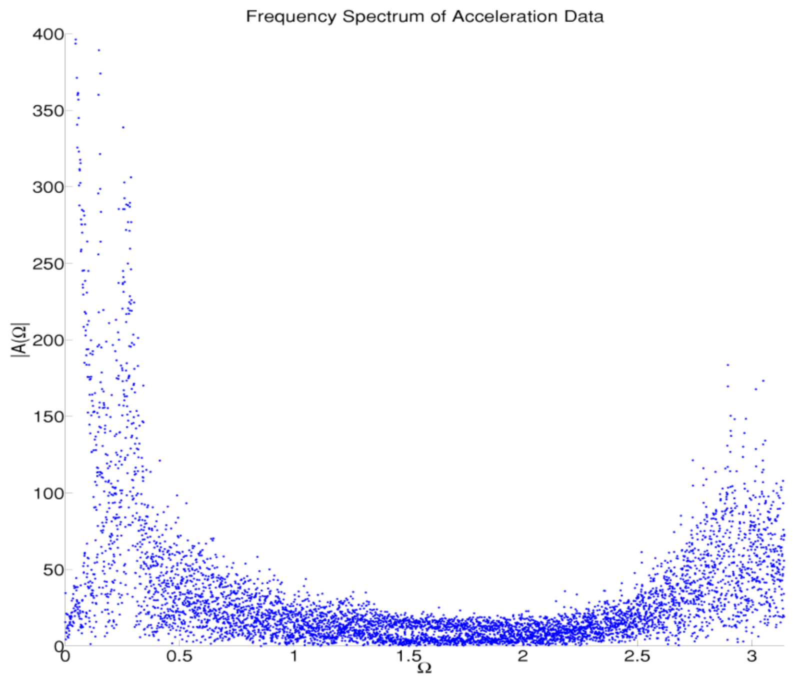


Figure 33: Frequency Spectrum of Unfiltered Accelerometer Data

A Butterworth filter can be used to remove the high-frequency components of the data. A Butterworth filter was chosen because it is maximally flat in the passband and it can be implemented with a low-order. 2nd and 4th-order systems were considered. Using Matlab, transfer functions were designed with a cut-off frequency of $\frac{\pi}{2}$ and zero gain. They are shown below.

$$H(z) \;=\; \frac{0.2929z^2 \;+\; 0.5858z \;+\; 0.2929}{z^2 + 0.1716}$$

$$H(z) \;=\; \frac{0.094z^4 \;+\; 0.3759z^3 \;+\; 0.5639z^2 \;+\; 0.3759z \;+\; 0.0940}{z^4 \;+\; 0.4860z^2 \;+\; 0.0177}$$

The frequency response of both filters were also obtained using Matlab. The flat passband is ideal so that the final position curve will not be amplified.

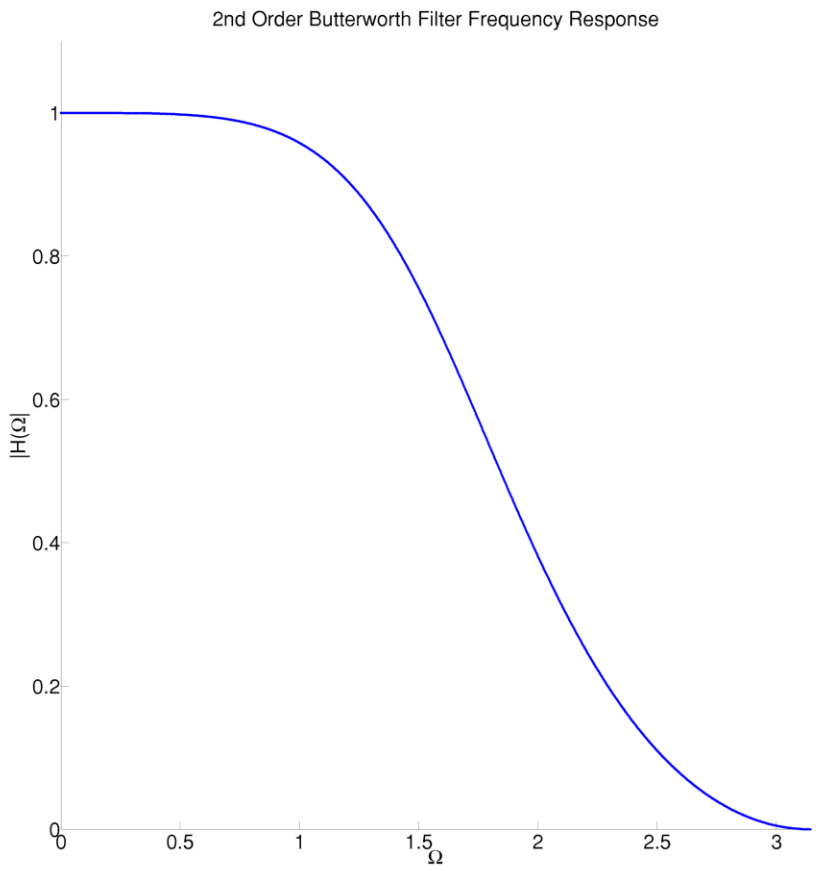


Figure 34: Frequency response of 2nd order Butterworth filter

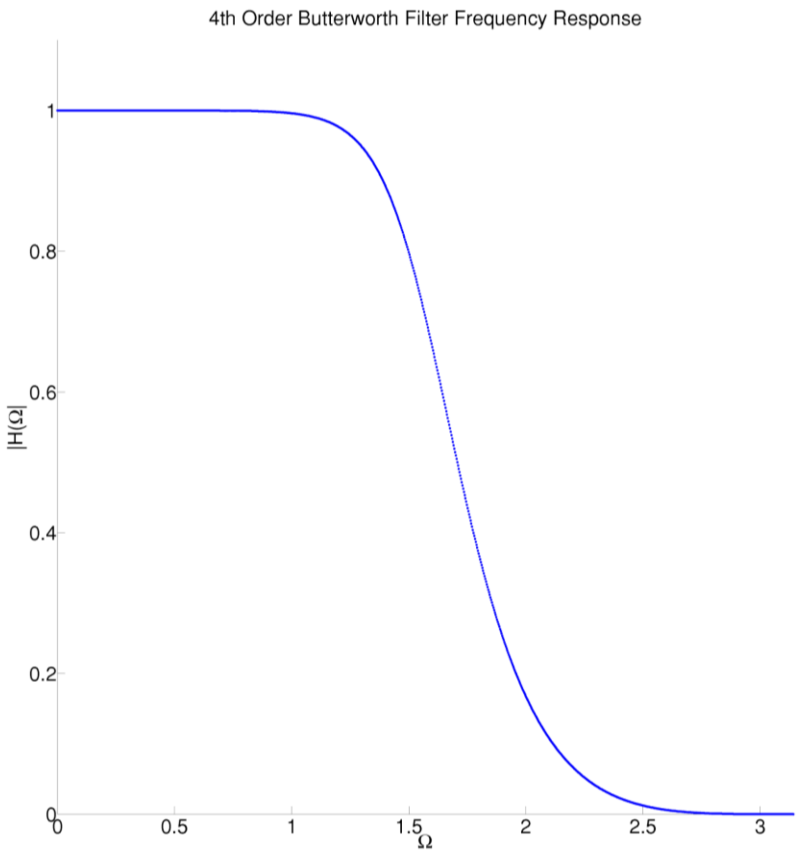


Figure 35: Frequency response of 2nd order Butterworth filter

These filters can be applied to the acceleration data to remove the high-frequency components. This is done by implementing the difference equations of each filter. The resulting frequency spectrum of the acceleration data is shown below. Notice that the high-frequency components are now gone.

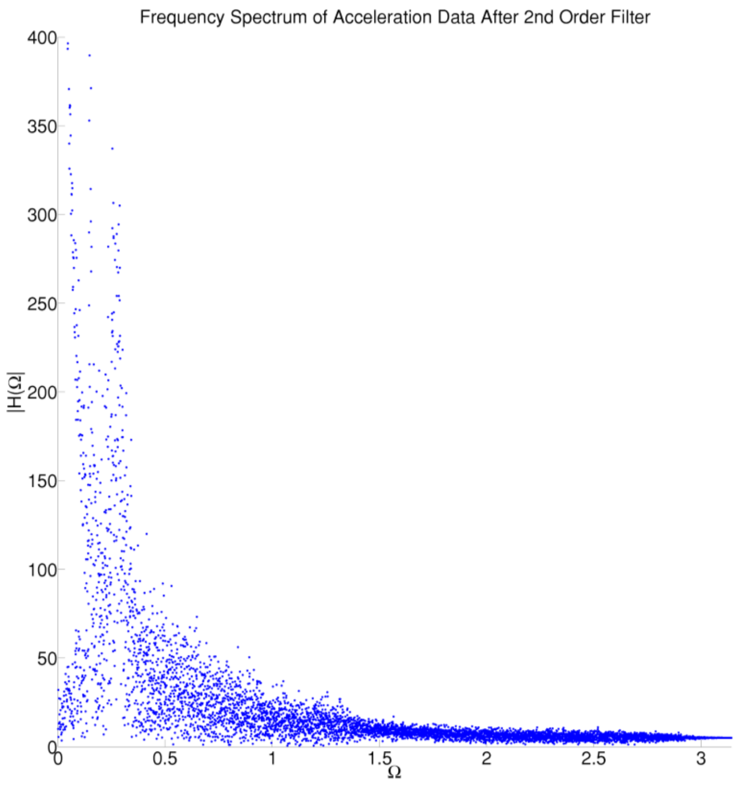


Figure 36: Frequency spectrum of acceleration data after 2nd order Butterworth filter

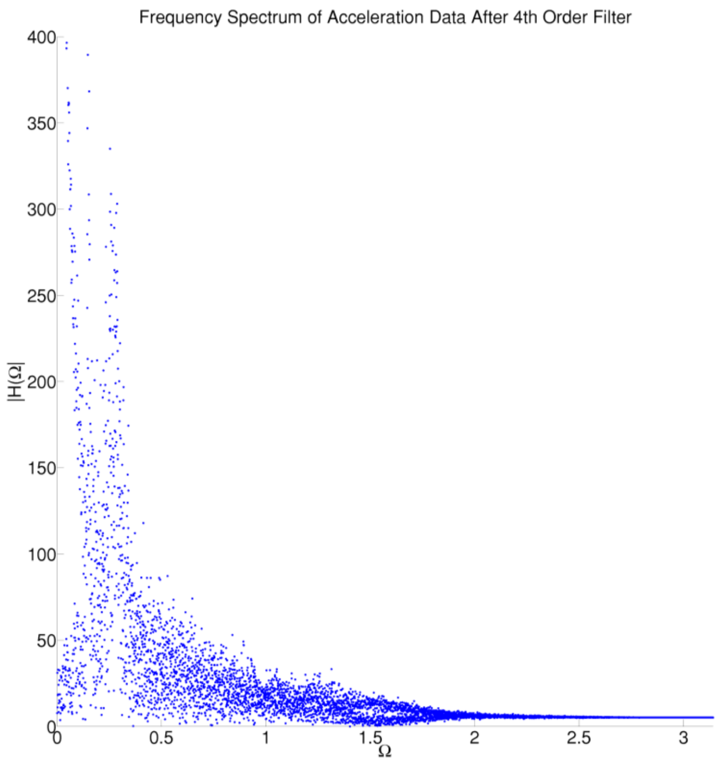


Figure 37: Frequency spectrum of acceleration data after 4th order Butterworth filter

Now, the resulting acceleration data can be plotted in the time domain. The result is a much smoother signal.

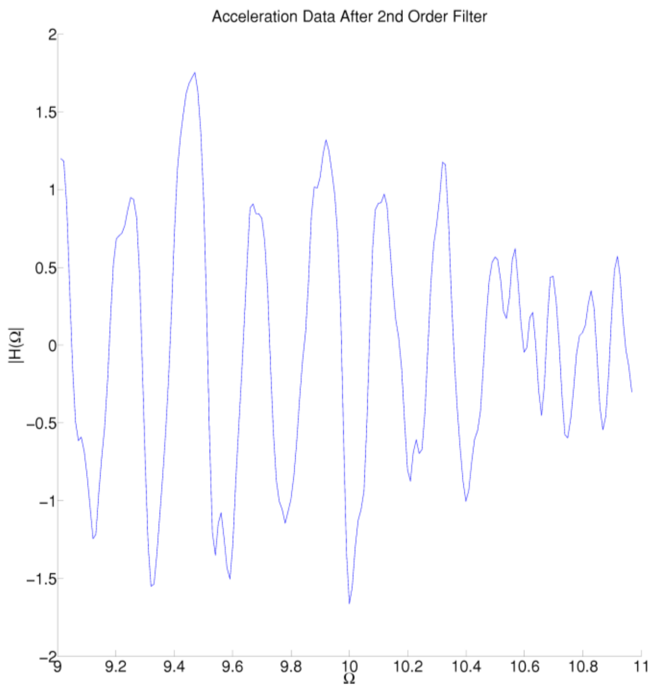


Figure 38: Acceleration data after 2nd order Butterworth filter

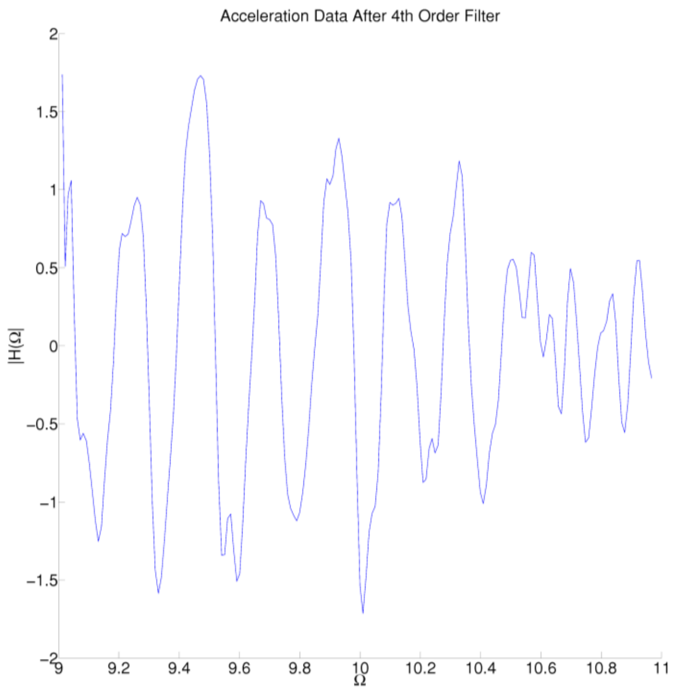


Figure 39: Acceleration data after 4th order Butterworth filter

[MJA,EES]

(m) Web Application

The web application is designed to provide the coaching staff and swim team members a simple tool that can provide analysis and visualization of workout data. A web application was decided upon because it can be used on a variety of platforms (e.g. a phone, or PC). Also, user interface design using HTML 5 offers a number of dynamic ways to present information without having to work with the constraints of any one specific platform. The core of the web application is a NodeJS server. This server will request data from a MongoDB database, which houses workout information. Finally, a front-end will be designed with Backbone and Bootstrap, to provide a dynamic interface for the end-user.

```
{
    "_id" : ObjectId("58fcdefe20487515bd9d30a3"),
    "rfid_tag" : 9569576,
    "date" : ISODate("2017-04-23T17:06:06.459Z"),
    "dateString" : "04/23/2017 13:06:06",
    "weight" : 48,
    "tot_time" : 9.75,
    "distance" : 4.35,
    "max_force" : 717.32,
    "coefficients" : [
        3.04259172225147e-07,
        -5.89781285073587e-06,
        -8.24047274119444e-05,
        0.00353902887813007,
        -0.0441638481256348,
        0.278540479626372,
        -0.963701526468412,
        1.85412286937981,
        -2.20679905227553,
        2.34838808866971,
        0.302731466019281
    ],
    "swimmer" : "Jack Wolfe",
    "acc_time" : 9.75
}
```

Figure 40 Example JSON for workout entry in database

1. Database

A database will be hosted on the server to store workout information for each athlete that uses the weight machines. During workouts, the database will be updated upon completion of a repetition. The web application will pull information off of the database.

There are several database packages that implement well with NodeJS; one of which is MongoDB. MonogoDB is a non-relational database that organizes data into 'collections'. The team opted for this non-relational structure because it allows for more flexibility in storing data. A relational database (think SQL) organizes data into structured tables with specific key-value pairs used to couple information. The SQL structure is not required for this application, and the added functionality of a non-relational structure lends itself well to this design.

```
{ "_id" : { "$oid" : "58b584c0cc9e0944528faf97" }, "photo"
: "images/mark.jpg", "rfid_tag" : 9570440, "name" : "Mark
Archual", "workouts" : [] }
```

Figure 41 Example JSON for a swimmer entry in the database

The database houses three main collections. A collection stores data following a document model approach (like json). One collection, 'workouts', will be used to store information from all repetitions of the weight machine. When the user exports data, this collection is searched. A second collection,

```
var mongojs = require('mongojs'); //Needed to connect to the database

var dbURL = 'test'; //URL to database, current assumed locally
//names of collection to pull from within database
var collections = ['swimmers', 'data', 'workouts', 'recent_workouts'];

var db = mongojs(dbURL, collections); //setup database connection

module.exports = db;
```

Figure 42 JavaScript used to collect to the MongoDB database through NodeJS

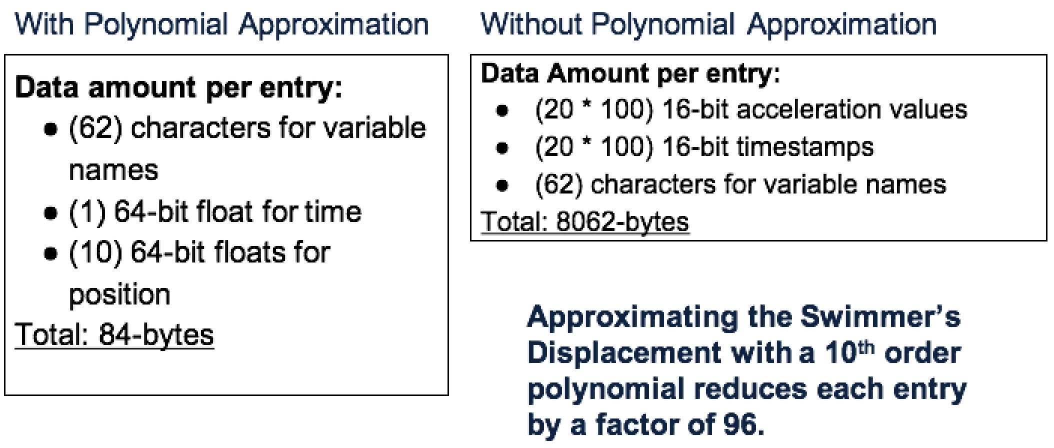
‘recent_workouts’, will be used to store the 10 most recent repetitions completed. Each document in the ‘recent_workouts’ collection follows the same structure as the ‘workouts’ collection, however it is a capped collection. A capped collection is a collection that limits the numbers of entries it keeps. For this design, a capped collection with a max size of 10 documents was chosen. These 10 most recent workouts are the only ones that are displayed on the website because it would be impractical to display all of the entries in the ‘workout’ collection in a manageable way. A final collection, ‘swimmers’ will also be required to map a swimmer’s name to a unique RFID tag. This collection also contains the path to an image on the server to display when the ‘View Swimmers’ page is selected. See Figures 40 and 41 for examples of each collection’s structure.

```
var express = require('express');
var router = express.Router();
var db = require('../lib/database.js');
var db_model = require('../database/workout_dbmodel');

router.get('/', function(req,res) {
    db.recent_workouts.find({}, null, {sort: {date: -1}}, function(err,workouts) {
        if (!err) {
            //console.log(workouts);
            return res.send(workouts);
        } else {
            return console.log(err);
        }
    });
});
```

Figure 43 An example of retrieving data from the 'recent_workouts' collection.

One specific aspect to note is that the ‘Position’ entry for the workout data does not correspond to the position of the swimmer, and instead corresponds to polynomial coefficients that will be used to replicate the data generated. Using this polynomial approximation reduces the size of each entry in our database by a factor of 96 as shown in Figure 44 below.



With Polynomial Approximation

Data amount per entry:
- (62) characters for variable names
- (1) 64-bit float for time
- (10) 64-bit floats for position

Total: 84-bytes

Without Polynomial Approximation

Data Amount per entry:
- (20 * 100) 16-bit acceleration values
- (20 * 100) 16-bit timestamps
- (62) characters for variable names

Total: 8062-bytes

Approximating the Swimmer’s Displacement with a 10th order polynomial reduces each entry by a factor of 96.

Figure 44: Data amounts with and without polynomial approximation

[MJA]

2. Web Server

The web server will be powered by NodeJS. Node is a server side event-driven JavaScript run-time environment. Node was chosen for two main reasons: its method of code execution and excellent library management tools. Figure 45 describes the overall execution of the web server, each component of which will be explained accordingly.

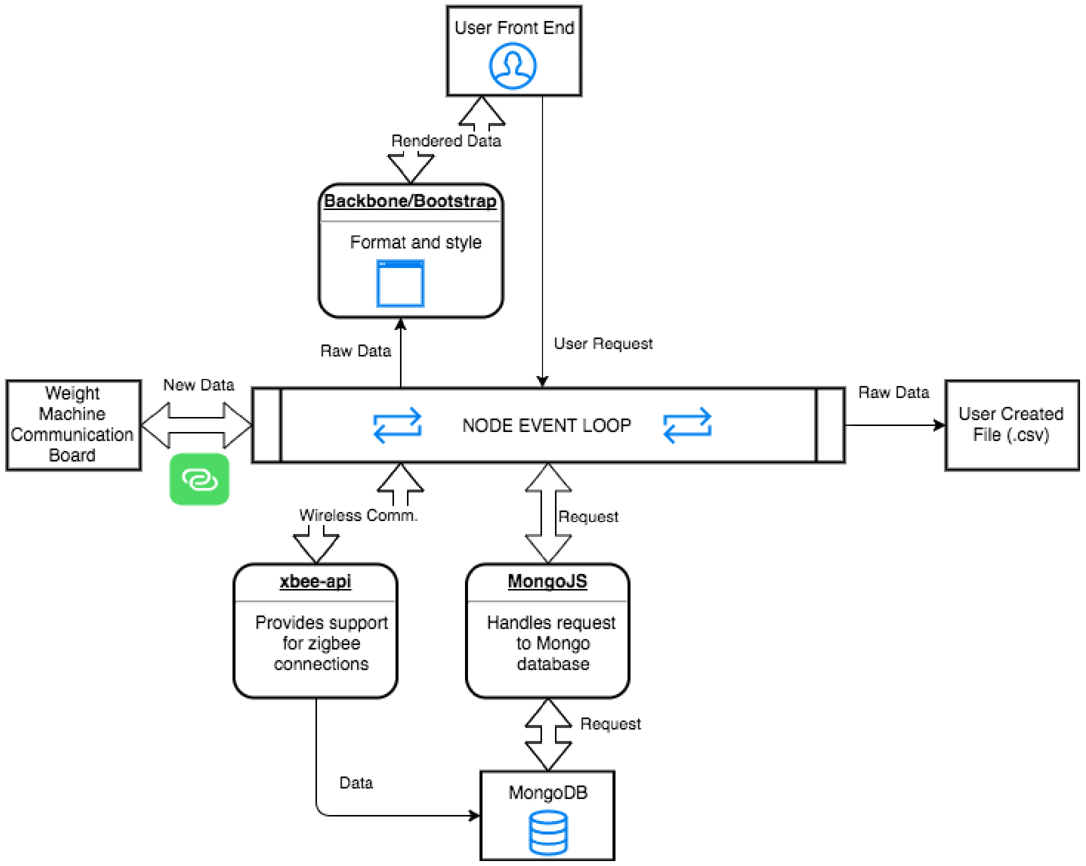


Figure 45: Web application flow diagram

NodeJS is driven primarily by I/O events. The beauty of NodeJS comes from how it handles responding to these events. Node uses an internal (i.e. hidden from the programmer) execution structure called the 'Event Loop'. The Event Loop processes and handles all I/O request asynchronously using callback functions. The key is that I/O requests are processed in parallel, and does not allow any one request to block the execution of another. The web application will be I/O driven with interaction from the coaching staff, and thus NodeJS was a natural fit.

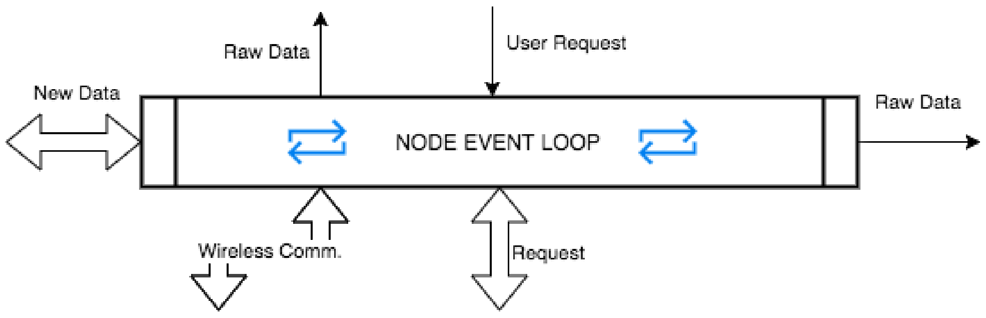


Figure 46: Node event loop

NodeJS also comes with a package installation tool called npm, or node package manager. Npm provides support for using open-source libraries in a project. The web application uses several of these libraries to extend functionality including:

RequireJS, ExpressJS, MongoJS, Grunt, Socket.IO, and xbee-api. Another great asset that npm provides is a *package.json* file that is generated for each application. This *package.json* file centralizes program dependencies on these external libraries to one file. Due to this development structure, the installation of our code on the Pine64 is relatively straightforward.

The ExpressJS library simplifies the implementation of a web server by abstracting away from the programmer the more cumbersome details of NodeJS. Express also lends a basic directory structure for organizing the files in a program. The web application design follows the structure shown to the right.

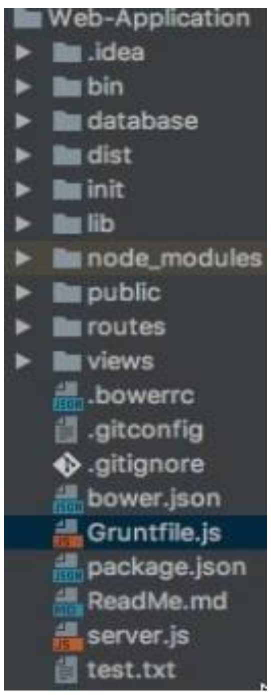
```
//delete the swimmer specified by name
server.post('/delete_swimmer', function(req,res) {
    //console.log(req.body.firstname);
    var queryName = sanitize(req.body.firstname) + " " +
sanitize(req.body.lastname);
    //console.log('deleting ' + queryName);
    db.swimmers.remove({'name': queryName}, function(err,result) {
        //console.log(result);
        if(result.n === 0) { //result.n specifies the number of
deletions, if this is 0 then no one was deleted
            res.sendFile(path.join(__dirname, '/public/',
'delete_fail.html'));
        }
        else { //successful delete
            res.status(200).sendFile(path.join(__dirname, '/public/',
'delete_success.html'));
        }
    });
});
```

Figure 48 Example of routing for deleting a swimmer from the database.

One of the most important tools used to manage compilation and testing of the server code is called Grunt. Grunt is a JavaScript task runner that can be used to automate tasks during development. Different tasks can be performed by installing Grunt plugins using the Grunt command line interface. These tasks are defined and managed in a file called Gruntfile.js. For the purposes of this project, Grunt was used to compile code for the front-end client, run a static code analyzer called JSHint, and automatically reboot our server when a file on the server was changed.

Socket.IO is used to open communication channels between the front-end and the server. This was key to the charting functionality of the website. When the user wants to chart the swimmer's displacement, they select a checkbox in a table. This triggers a request to the data base to retrieve the corresponding workout. This information is then returned to the client and plotted using ChartJS.

A key design component of the server is the idea of 'routing'. Routing essentially structures the program logic of an application into files that correspond to the address the user navigates to in a web browser. The routes will be defined in server.js, which will then call the corresponding file to execute in the 'routes' folder. See example in Figure 48. More sophisticated examples can be built from this basic structure.



```
Web-Application
  ▶ .idea
  ▶ bin
  ▶ database
  ▶ dist
  ▶ init
  ▶ lib
  ▶ node_modules
  ▶ public
  ▶ routes
  ▶ views
    .bowerrc
    .gitconfig
    .gitignore
    bower.json
    Gruntfile.js
    package.json
    ReadMe.md
    server.js
    test.txt
```

Figure 47: Web application file structure

The last core library to the application is called *xbee-api.* This library provides support for communicating with XBee radios. It is important to note that integrating the connection directly into the node application will be handled by the *serialport* API, the xbee-api simply provides a direct way to interface with the XBee protocol. The following example shows basic communication with an XBee radio.

```
var SerialPort = require('serialport')
var DP = require('./data-processor.js')
var db = require('./database.js')
var xbee_api = require('xbee-api')

var PORT_M = 'FTDI'

var Xbee = function () {};
var _data = [];

var xbeeAPI = new xbee_api.XBeeAPI({
  api_mode: 1
});


Xbee.prototype.start = function()
{
  SerialPort.list(function(err, ports) {
    ports.forEach(function(p) {
      if (p.manufacturer == PORT_M)
      {
        console.log("opening port " + p.comName);

        port = new SerialPort(p.comName , {
          baudrate: 115200,
          parser: xbeeAPI.rawParser()
        });

        port.on('open', function() {
          console.log('serial port open and ready to receive data');
        });

        port.on('error', function() {
          console.log('error opening serial port!');
        });

        xbeeAPI.on("frame_object", function(frame) {…
```

Figure 49 An example of XBEE data parsing on the server

3. Front-end

The front-end of the web application is just as complex and nuanced as the server. Bower is a front-end package manager, analogous to npm. Loading these packages can be a complex process and thus it is common to use a front-end module loader to manage these libraries at run time. RequireJS is an asynchronous module loader that is used in the design to manage this process. Some of the front-end libraries in this project include: Backbone, Underscore, Bootstrap, and ChartJS.

```
require.config({
    paths: {
        'jquery': 'vendor/jquery/dist/jquery',
        'underscore': 'vendor/underscore/underscore',
        'backbone': 'vendor/backbone/backbone',
        'backbone.radio' : 'vendor/backbone.radio/src/backbone.radio',
        'text' : 'vendor/text/text',
        'bootstrap' : 'vendor/bootstrap/dist/js/boostrap',
        'tether': 'vendor/tether/dist/tether',
        'holder': 'vendor/holderjs/holder',
    },
    shim: {
        underscore: {
            exports: '_'
        },
        jquery: {
            exports: '$'
        },
        backbone: {
            exports: 'Backbone',
            deps: ['jquery', 'underscore']
        },
        bootstrap: {
            deps: ['jquery', 'tether']
        }
    },
    deps: ['jquery', 'underscore']
});
```

Figure 50 RequireJS file used to load front-end dependencies.

Backbone.JS provides structure to a front-end through three main components: collections, views, and models. This structure helps separate the functionality of the front-end from the backing, or 'business', logic. The key components are the concept of a 'view' and 'model'. A view is how the data is presented the user. A model contains the program logic to execute based on the user's request. See the following diagram from the Backbone.JS website.

A 'collection' helps manage views with several models. For example, consider the management and presentation of one repetitions worth of data. This information can be placed into a 'workout' model which is presented to the user with the corresponding 'workout' view. Several 'workout' views are then managed in a 'workout' collection**.** See the following code diagrams for an example of how this can be structured. By structuring the web application this way, a framework is in place for fluid data updates through structured content management.

```
var app = app || {};
define(['/../common', 'backbone'],
function(common) {
    var Backbone =
require('backbone');
    //console.log('Swimmer Model');
    //Database model for swimmer
    app.Swimmer =
Backbone.Model.extend({
        idAttribute: '_id',
        defaults: {
            photo:
'images/peresie.jpg',
            rfid_tag: '0',
            name: 'Coach Peresie',
            workouts: '[]'
        }
    });
    return app.Swimmer;
});
```

Figure 51 Example of a Backbone Model for a swimmer.

```
require(['./common',
'./views/AnalyticsView'],
function(common, AnalyticsView)
{
    $(function() {
        var app = new
AnalyticsView();
    })
});
```

Figure 52 Example of instantiating a Backbone View with RequireJS

```
//A collection of swimmers
var app = app || {};
define(['/../common', '/../models/Swimmer'],
function(common) {
    var Backbone = require('backbone');
    //console.log('Swimmer Collection');
    app.SwimmerCollection =
Backbone.Collection.extend({
        model: app.Swimmer,
        url: '/swimmer' //server endpoint to
retrieve data from for model
    });
    return app.SwimmerCollection;
});
```

Figure 53 Example of a Backbone Collection used to hold several Swimmer Models.

Underscore is a templating engine that can be used to create dynamic HTML pages. This is important for this application because the content that needs to be displayed on the website is constantly changing as the swim team uses the machine and more information is added to the database. Underscore is versatile and a dependency of the Backbone library so its use this project was an obvious choice. Underscore templates can be found in the HTML files for the site. Also included in the HTML files are the Bootstrap CSS template that is used to style the website and a reference to the RequireJS file to pull in the extra required front-end libraries.

```
<tbody id = "workout">
<script id = "workoutTemplate" type="text/template">
<td class = "dateString"><%= dateString %></td>
<td class = "swimmer"><%= swimmer%></td>
<td><%= tot_time%></td>
<td><%= distance%></td>
<td><%= weight%></td>
<td><%= max_force%></td>
</script>
</tbody>
```

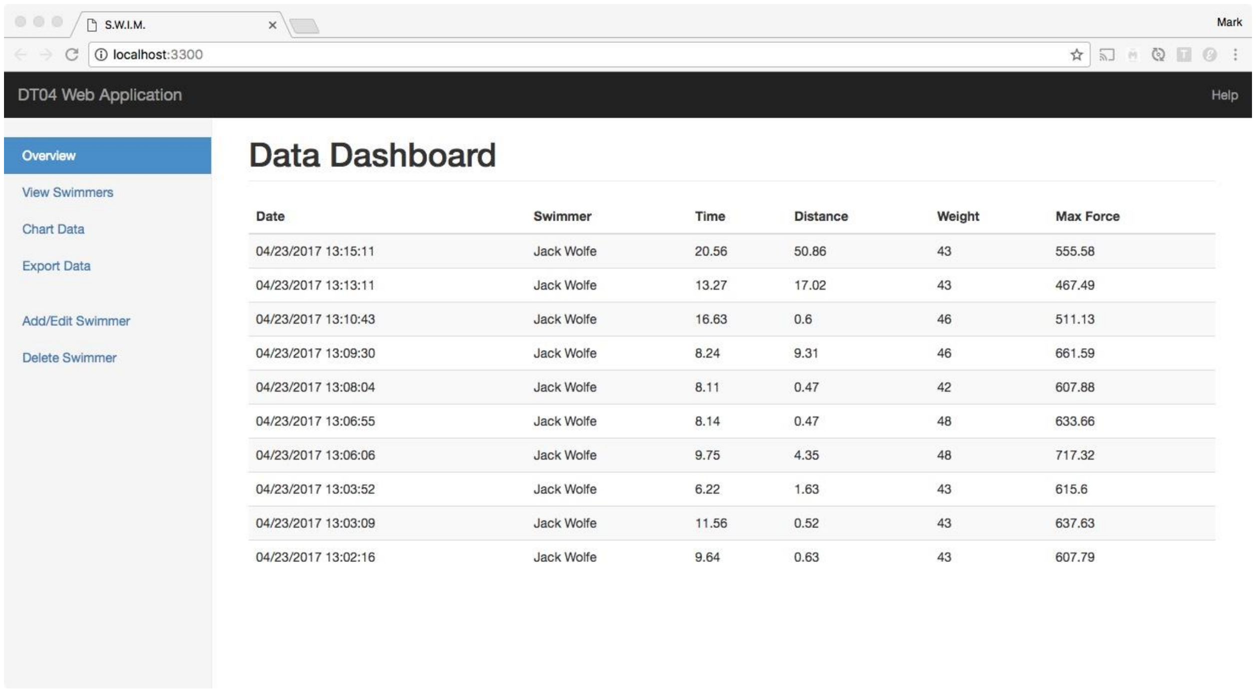
Figure 55 Example of an Underscore template used to generate dynamic HTML.

One of the design requirements for the application was to provide data interpretation through line charts. This was provided by the ChartJS library. The Charts are organized with a Backbone view, collection and model. The information is presented to the use through a table on the website with the option to plot one repetition at a time. When a workout is selected, a Socket.IO request is fired to the server and the corresponding workout information is retrieved. This information is then sent back to the client and rendered with the ChartJS library.

```
socket.on('position', function(data) {
//console.log(data.result.position);
myChart.data.datasets[0].data = data.result.position;
myChart.data.labels = data.result.time;
//console.log(myChart.data.datasets);
myChart.update(); //renders chart
});
```

Figure 54 Example of creating a chart through ChartJS and a SocketIO request

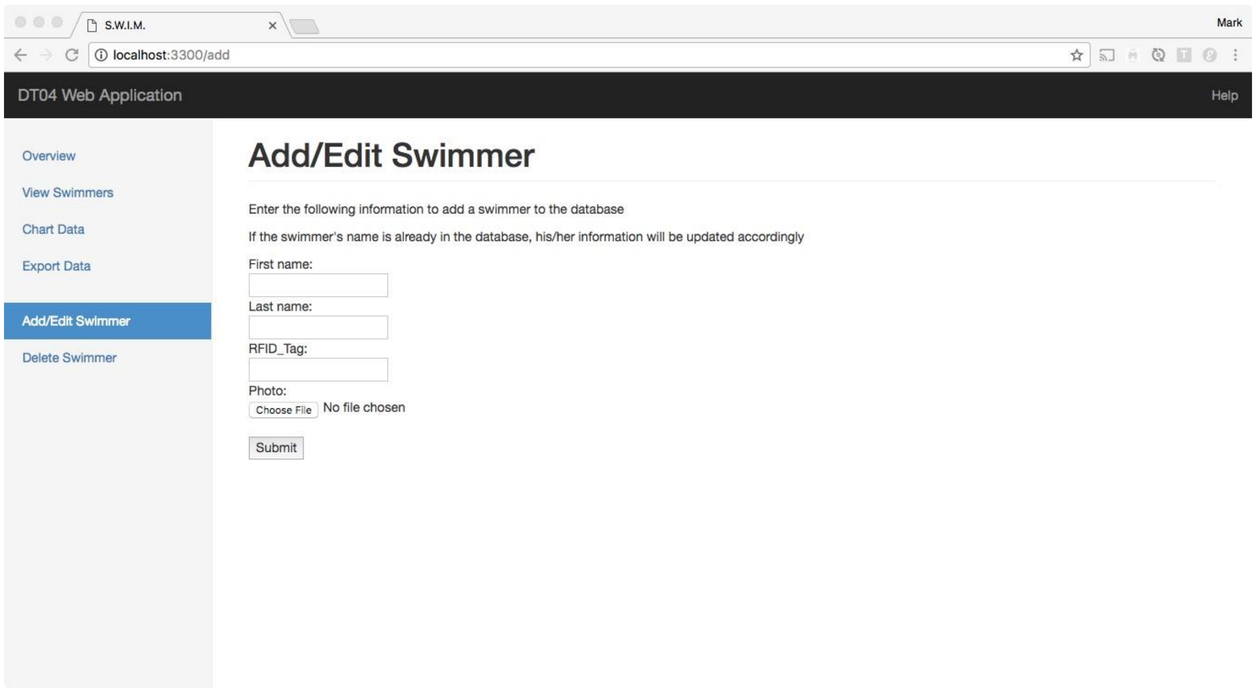
[MJA]



Data Dashboard

Date	Swimmer	Time	Distance	Weight	Max Force
04/23/2017 13:15:11	Jack Wolfe	20.56	50.86	43	555.58
04/23/2017 13:13:11	Jack Wolfe	13.27	17.02	43	467.49
04/23/2017 13:10:43	Jack Wolfe	16.63	0.6	46	511.13
04/23/2017 13:09:30	Jack Wolfe	8.24	9.31	46	661.59
04/23/2017 13:08:04	Jack Wolfe	8.11	0.47	42	607.88
04/23/2017 13:06:55	Jack Wolfe	8.14	0.47	48	633.66
04/23/2017 13:06:06	Jack Wolfe	9.75	4.35	48	717.32
04/23/2017 13:03:52	Jack Wolfe	6.22	1.63	43	615.6
04/23/2017 13:03:09	Jack Wolfe	11.56	0.52	43	637.63
04/23/2017 13:02:16	Jack Wolfe	9.64	0.63	43	607.79

Figure 56 Web Application Homepage



Add/Edit Swimmer

Enter the following information to add a swimmer to the database

If the swimmer's name is already in the database, his/her information will be updated accordingly

First name:

Last name:

RFID_Tag:

Photo:

Choose File No file chosen

Submit

Figure 57 Web Application form for editing/adding swimmers

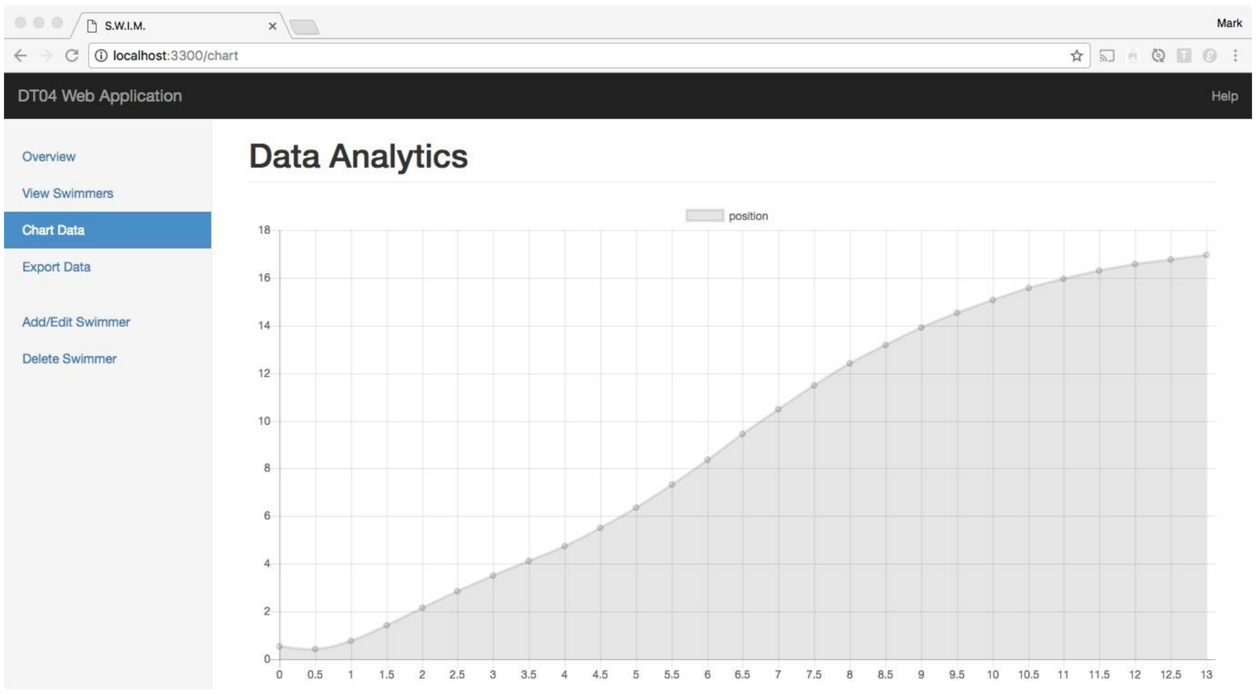


Figure 58 Web application chart example

(n) Block Diagrams

Level 0

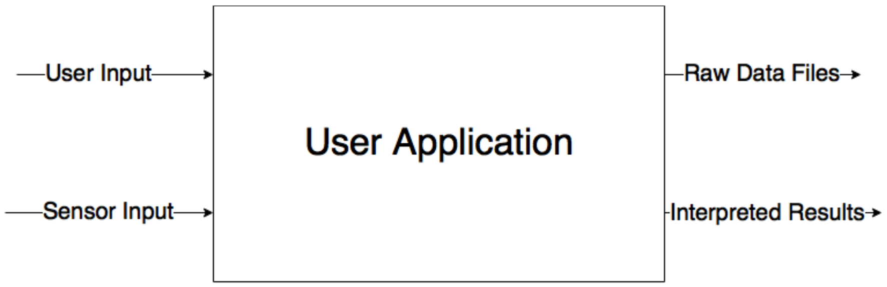


Figure 59: Level 0 Software Block Diagram

Module	User Application
Designer	Mark Archual
Inputs	User Input, Sensor Input
Outputs	Raw Data Files, Interpreted Results
Description	The User Application is a web application that the swim team will be able to use to interpret their workouts. It will display interpreted data values from the sensor board, as well as allow the users to create an exportable file of raw data values that can be used in another application.

Table 14: User Application Level 0 Module

Level 1

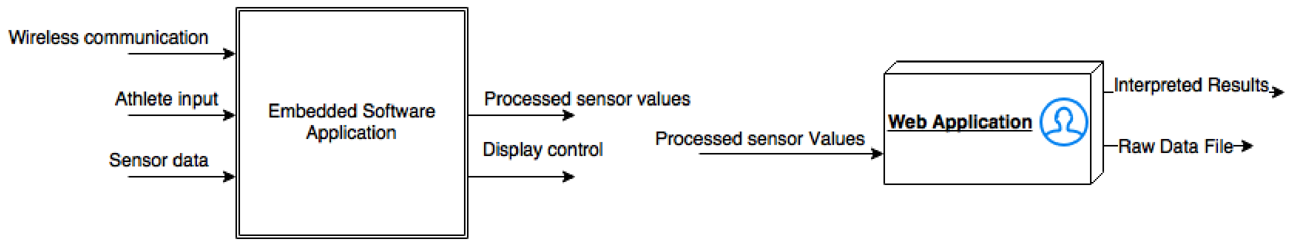


Figure 60: Level 1 Software Block Diagram

Module	Embedded Software Application
Designer	Ethan Schweinsberg
Inputs	Wireless communication, Athlete input, Sensor values
Outputs	Wireless communication, Processed sensor values, Display control
Description	The embedded software application handles gathering data from sensors, digital filtering, and wireless communication.

Table 15: Embedded Software Level 1 Module

Module	Web Application
Designer	Mark Archual
Inputs	Processed Sensor Values
Outputs	Raw Data Files, Interpreted Results
Description	The Web Application is the interface that the swim team will be able to use to view data from their workouts. It will display interpreted data values in tabular and graphical form. The application will also allow the users to export a subset of recorded raw data values. Any user configuration or diagnostics of the main sensor board for a power rack will be done through this application.

Table 16: Web Application Level 1 Module

Level 2: Embedded Software

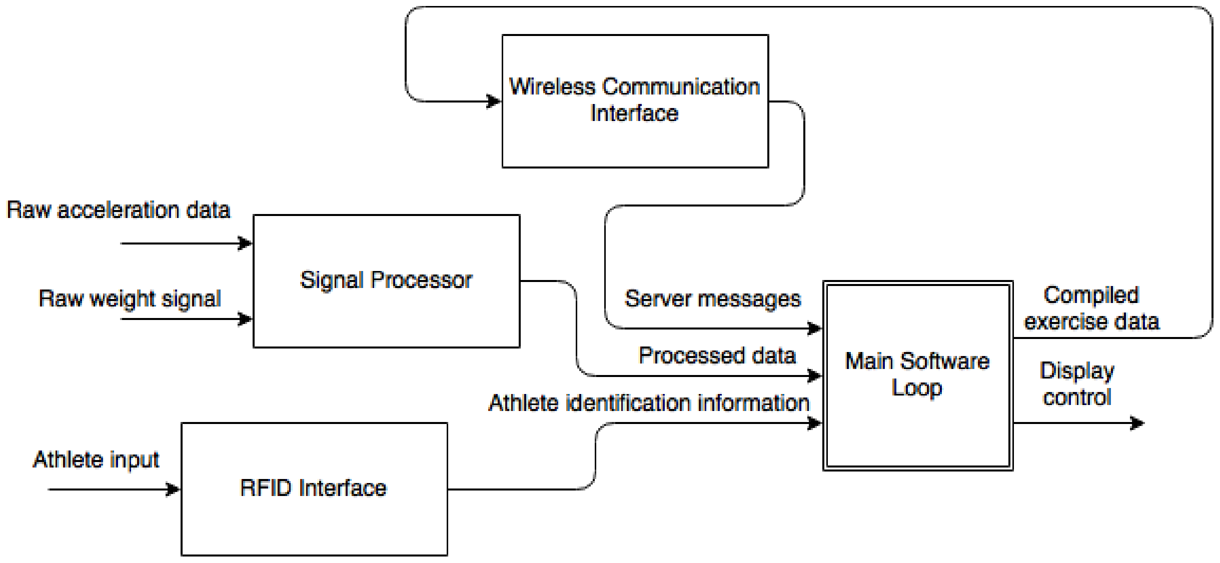


Figure 61: Level 2 Embedded Software Block Diagram

Module	Wireless Communication Interface
Designer	Ethan Schweinsberg
Inputs	Compiled exercise data
Outputs	Server messages
Description	The Wireless Communication Interface handles maintaining a wireless connection to the server. It will be capable of receiving and sending messages to and from the server.

Table 17: Wireless Communication Interface Level 2 Module

Module	Signal Processor
Designer	Ethan Schweinsberg
Inputs	Raw acceleration data, Strain gauge signal
Outputs	Processed data
Description	The signal processor is responsible for processing raw sensor outputs into meaningful data.

Table 18: Signal Processor Level 2 Module

Module	RFID Interface
Designer	Ethan Schweinsberg
Inputs	Athlete input
Outputs	Athlete data
Description	The RFID interface captures RFID tag data and translates it into information that can be used to uniquely identify an athlete.

Table 19: RFID Interface Level 2 Module

Module	Main Software Loop
Designer	Ethan Schweinsberg
Inputs	Server messages, processed data, Athlete identification information
Outputs	Compiled exercise data
Description	The Main Software Loop controls the various interfaces and compiles data to send to the server. In addition, the Main Software Loop controls a display.

Table 20: Main Software Loop Level 2 Module

Level 2: Web Application

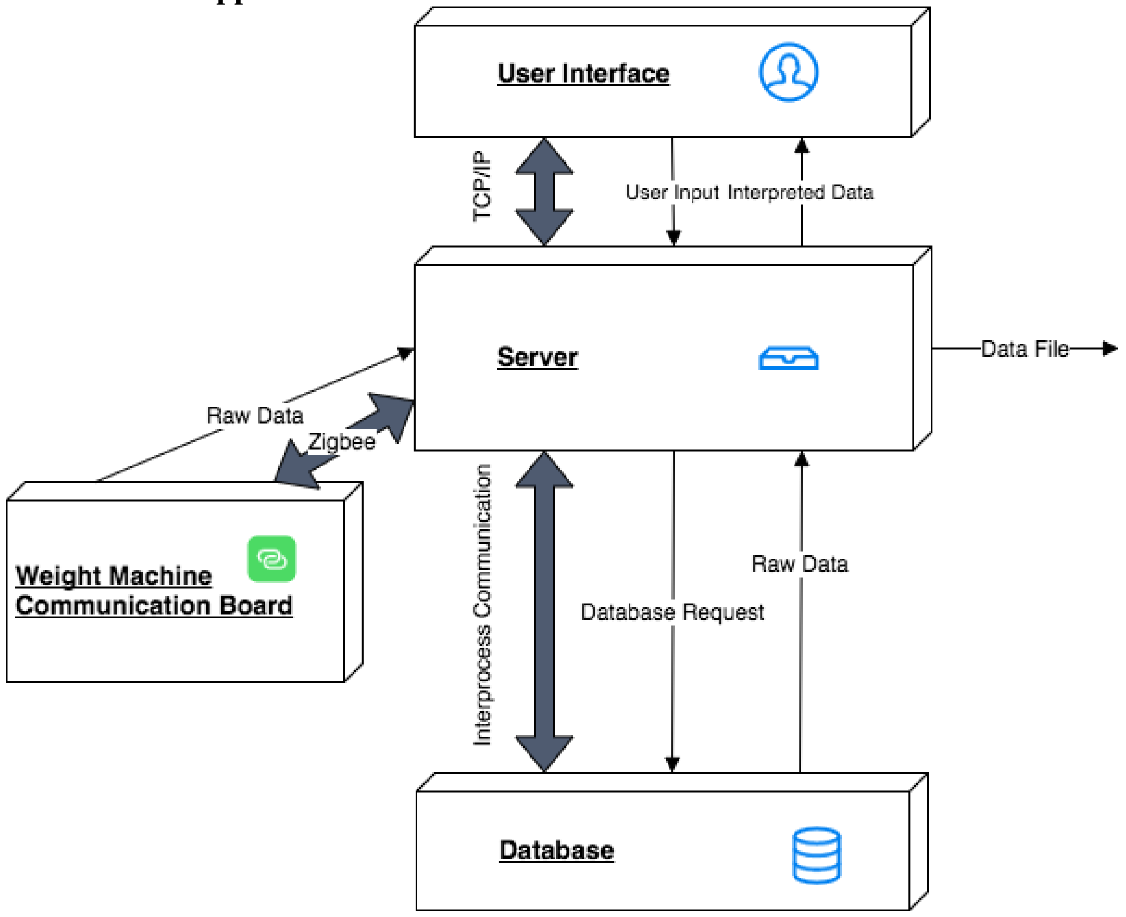


Figure 62: Level 2 Web Application Block Diagram

Module	User Interface
Designer	Mark Archual
Inputs	Interpreted Data
Outputs	User Input
Description	The user interface contains code that will help style the information that is passed to it by the server. It is the last step before the user views the information.

Table 21: User Interface Level 2 Module

Module	Server
Designer	Mark Archual
Inputs	Sensor Input, User Input, Data
Outputs	Interpreted Data, Database Requests
Description	The server handles data routing. Any user input will be mapped to an appropriate callback function that will request data from the database. The server will also be open to accept incoming data from the communication board on the weight machines.

Table 22: Server Level 2 Module

Module	Database
Designer	Ethan Schweinsberg
Inputs	Database Requests
Outputs	Data
Description	The database will house all data recorded by the sensor board on the weight machines.

Table 23: Database Level 2 Module

Level 3: Embedded Software

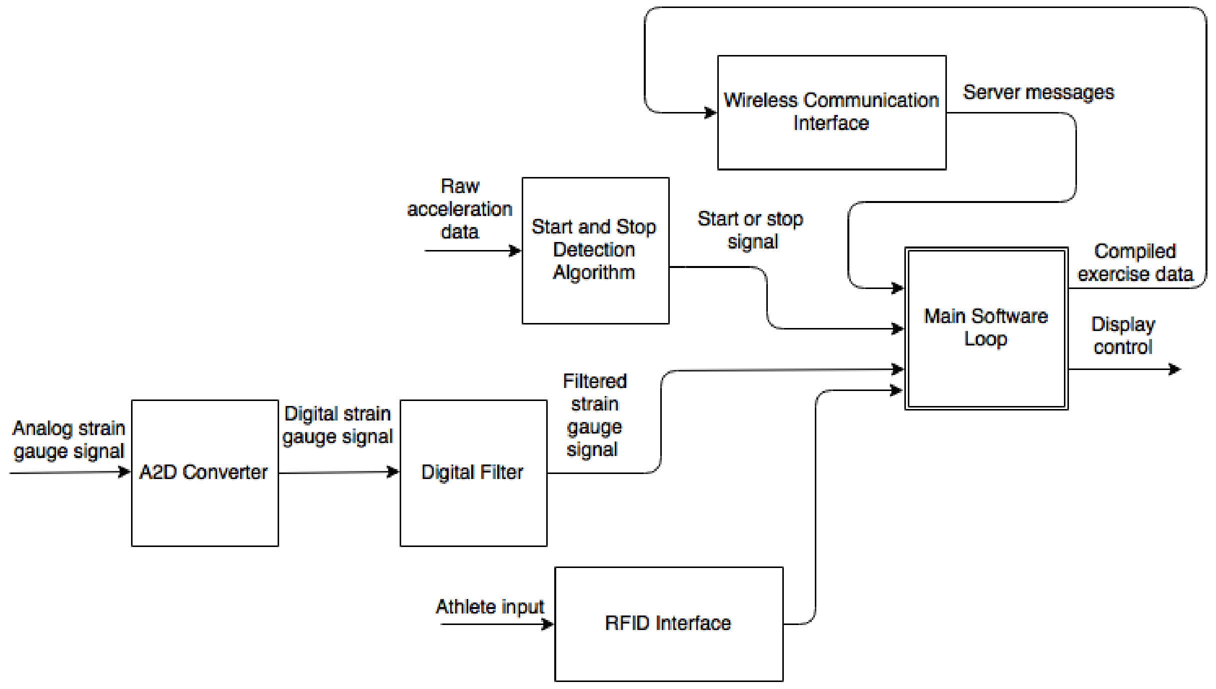


Figure 63: Level 3 Embedded Software Block Diagram

Module	A/D Converter
Designer	Ethan Schweinsberg
Inputs	Analog strain gauge signal
Outputs	Digital strain gauge signal
Description	The A2D Converter converts the analog signal from the strain gauge into a digital value that accurately represents the amount of strain.

Table 24: A/D Converter Level 3 Module

Module	Digital Filter
Designer	Ethan Schweinsberg
Inputs	Digital strain gauge signal
Outputs	Filtered strain gauge signal
Description	Digital Filter removes any noise from the strain gauge signal that may interfere with getting an accurate reading of the strain the machine that is caused by a variation of weight used by the athlete.

Table 25: Digital Filter Level 3 Module

Module	Start and Stop Detection Algorithm
Designer	Ethan Schweinsberg
Inputs	Raw acceleration data
Outputs	Filtered acceleration data
Description	Start and Stop Detection Algorithm utilizes the accelerometer data to determine when a swimmer starts or stops a swim.

Table 26: Start and Stop Detection Algorithm

Web Application

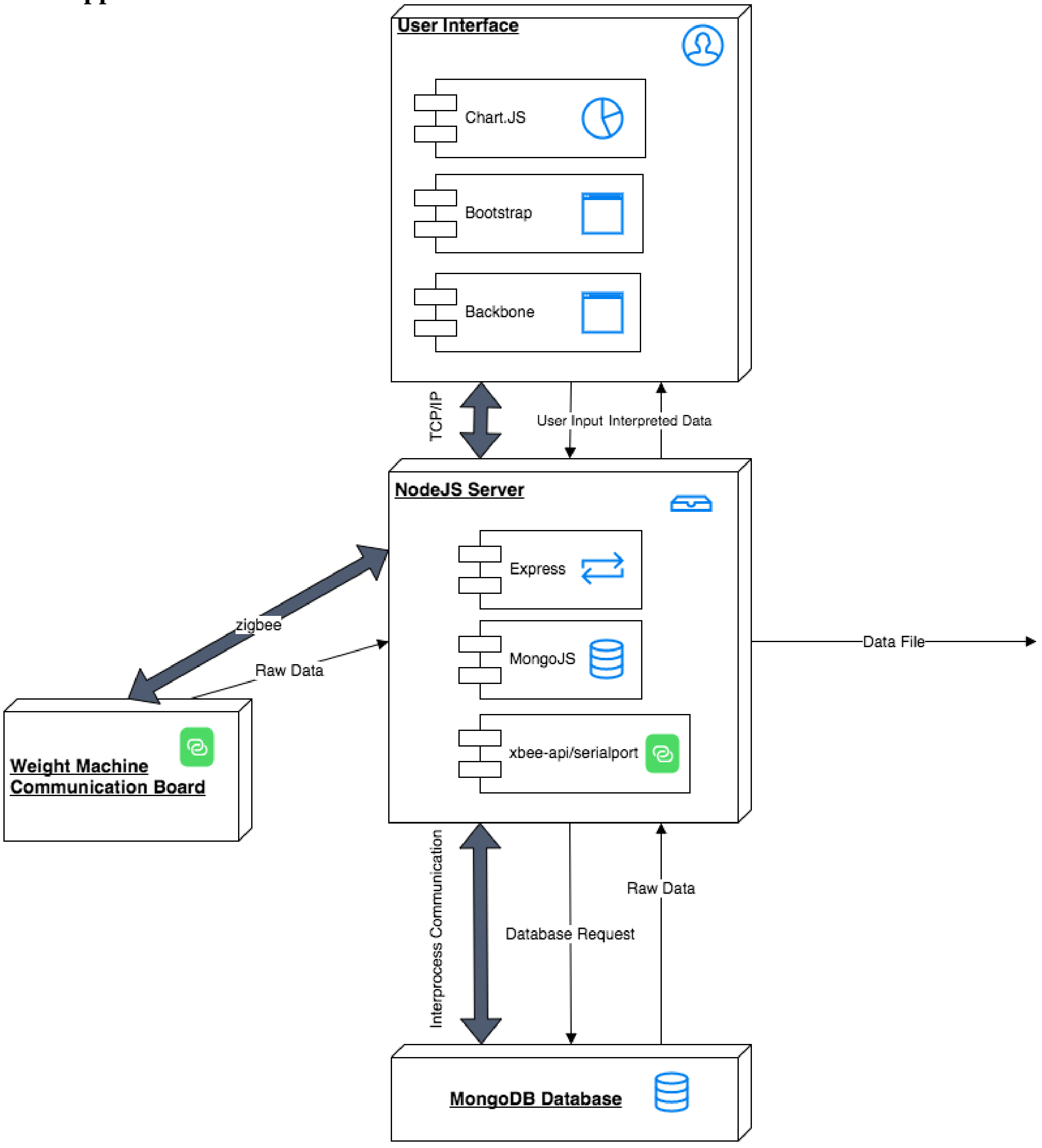


Figure 64: Web Application Level 3 Block Diagram

Operation, Maintenance and Repair Instructions

Section 1.01 Operation

(a) Server and Web Application

The server must be powered on and contain a populated database that follows the formats shown in the report and on the GitHub repository. Installation and setup procedures for the web application and database can be found on the GitHub repository, see Article XII for more information on the GitHub. Once operational, a help menu can be accessed from the homepage of the web application. The homepage of the web application and database will automatically updated within four seconds of a completed repetition. Ensure that the swimmer and RFID tag are paired properly in the database and web application. This can be managed through the 'Add/Edit Swimmer' page of the web application.

(b) Hardware

The display and sensor board must be equipped with a 9V battery as described in Article III, subsection (i). Once powered on the device will prompt the user to scan his/her RFID tag. Next the swimmer will be prompted to weigh the bucket. Once the reading has stabilized, the screen will blink a few times and then set itself to 00:00. The swimmer can now begin a repetition. The system will automatically detect when the swimmer pushes off the wall and stop data acquisition automatically when the system detects the swimmer has finished the repetition. The system has a cool down period equal to the length of time of the repetition plus an additional five seconds to allow adequate time for the swimmer to swim back to the machine. In between repetitions the swimmer can add or remove weight from the bucket. If the more than 10 pounds are added to the bucket, the seven-segment display will automatically show the new weight reading and flash when stabilized. The display will also show the weight reading when the bucket has been tipped to empty water back into the pool. This is detected with the x and y axes of the accelerometer. A new swimmer can be tagged into the system at any time. When the swimmer has completed his/her workout, the system should be shut down and stored until needed again to save battery life.

Section 1.02 Maintenance

(a) Server and Web Application

Each year the database should be updated as the swim team roster changes. This can be done through the 'Add/Edit Swimmer' and 'Delete Swimmer' pages of the web application.

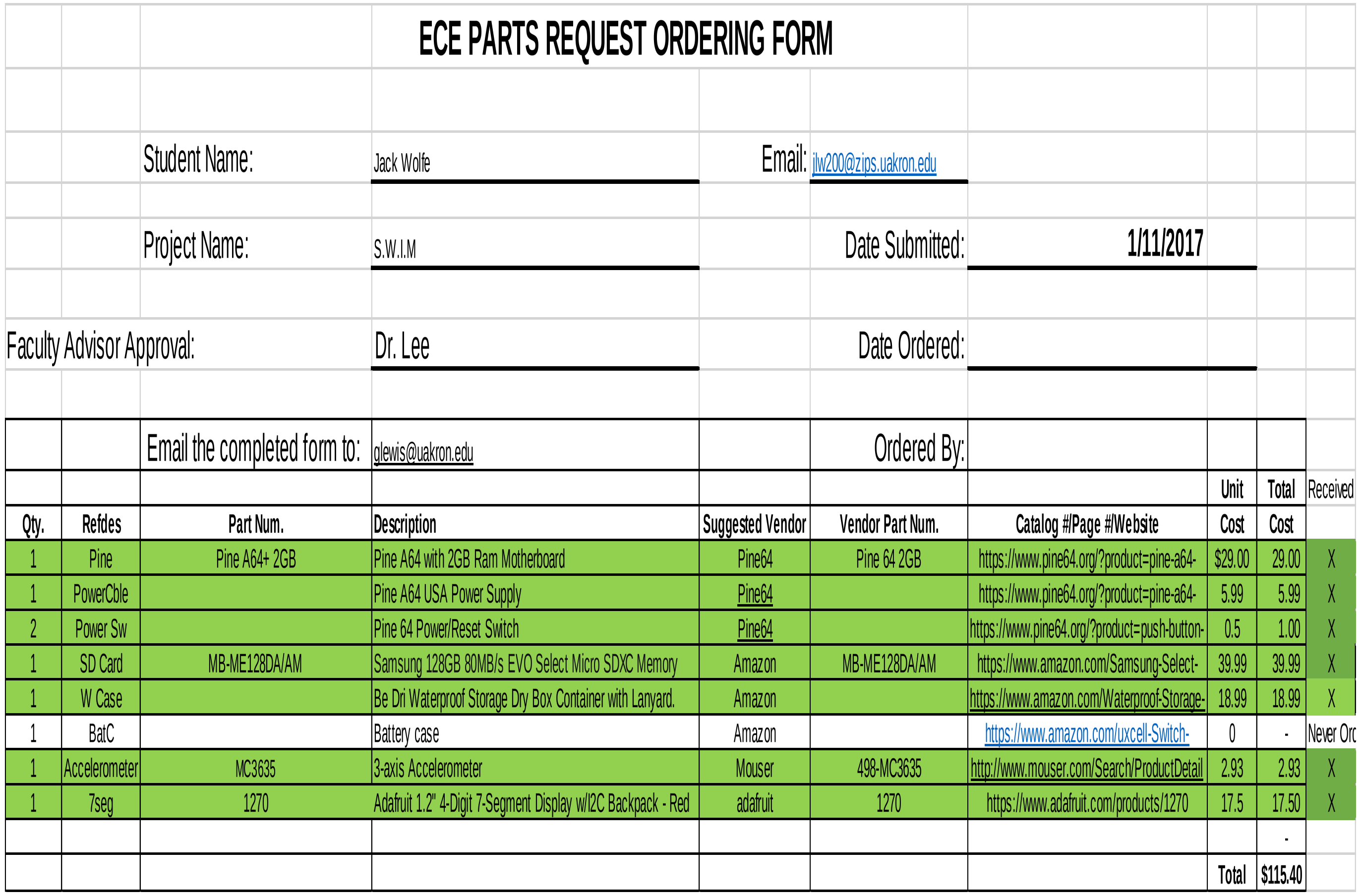
(b) Hardware

Replace batteries, and other components, as needed. The battery will need to be changed when the scale no longer resets to a zero value. Also, perform a regulatory visual inspection to ensure waterproofing is still adequate and cables are not damaged.

Testing Procedures

Each software and hardware component was individually tested and implemented before full-scale system integration. For full-scale system integration, the system was taken to the pool, and many tweaks were implemented in real-time and tested. A video showing the functionality of the system can be found online at: https://www.youtube.com/watch?v=0puRJ8HO66c

Financial Budget



ECE PARTS REQUEST ORDERING FORM

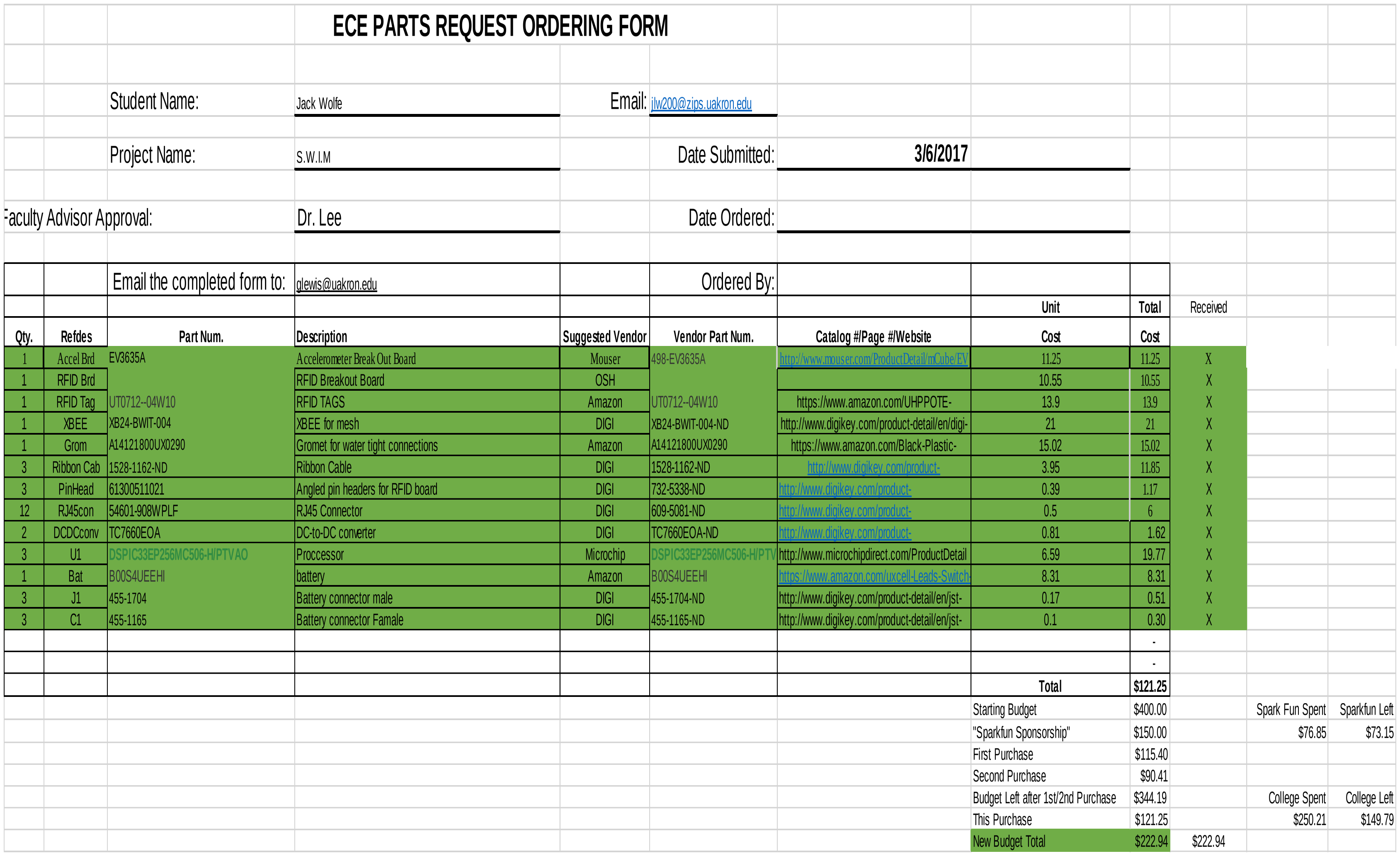
Student Name:	Jack Wolfe
Email:	jlw200@zips.uakron.edu
Project Name:	S.W.I.M
Date Submitted:	1/11/2017
Faculty Advisor Approval:	Dr. Lee
Date Ordered:	
Email the completed form to:	glewis@uakron.edu
Ordered By:	

Qty.	Refdes	Part Num.	Description	Suggested Vendor	Vendor Part Num.	Catalog #/Page #/Website	Unit Cost	Total Cost	Received
1	Pine	Pine A64+ 2GB	Pine A64 with 2GB Ram Motherboard	Pine64	Pine 64 2GB	https://www.pine64.org/?product=pine-a64-	$29.00	29.00	X
1	PowerCble		Pine A64 USA Power Supply	Pine64		https://www.pine64.org/?product=pine-a64-	5.99	5.99	X
2	Power Sw		Pine 64 Power/Reset Switch	Pine64		https://www.pine64.org/?product=push-button-	0.5	1.00	X
1	SD Card	MB-ME128DA/AM	Samsung 128GB 80MB/s EVO Select Micro SDXC Memory	Amazon	MB-ME128DA/AM	https://www.amazon.com/Samsung-Select-	39.99	39.99	X
1	W Case		Be Dri Waterproof Storage Dry Box Container with Lanyard.	Amazon		https://www.amazon.com/Waterproof-Storage-	18.99	18.99	X
1	BatC		Battery case	Amazon		https://www.amazon.com/uxcell-Switch-	0	-	Never Orc
1	Accelerometer	MC3635	3-axis Accelerometer	Mouser	498-MC3635	http://www.mouser.com/Search/ProductDetail	2.93	2.93	X
1	7seg	1270	Adafruit 1.2" 4-Digit 7-Segment Display w/I2C Backpack - Red	adafruit	1270	https://www.adafruit.com/products/1270	17.5	17.50	X
								-	
							Total	$115.40	

Table 27: Parts Order Number 1

	ECE PARTS REQUEST ORDERING FORM								
Student Name:	Jack Wolfe		Email:	jlw200@zips.uakron.edu					
Project Name:	S.W.I.M		Date Submitted:	**2/1/2017**					
pproval:	Dr. Lee		Date Ordered:						
Email the completed form to:	glewis@uakron.edu		Ordered By:						
					Unit	Total	Received		
Part Num.	**Description**	**Suggested Vendor**	**Vendor Part Num.**	**Catalog #/Page #/Website**	**Cost**	**Cost**			
KIT-13198	RFID Starter kit	Sparkfun	KIT-13198	https://www.sparkfun.com/products/13198	49.95	49.95	X		
WRL-11697	Xbee Explorer Dongle	Sparkfun	WRL-11697	https://www.sparkfun.com/products/11697	24.95	24.95	X		
BOB-13878	Load Sensor Combinator	Sparkfun	BOB-13878	https://www.sparkfun.com/products/13878	1.95	1.95	X		
INA827AIDGKR	Instrimentational Amplifier	DIGIKEY	296-30588-1-ND	http://www.digikey.com/product-	3.01	12.04	X		
RC0805FR-0780R6L	80.6 OHM Resistor	DIGIKEY	311-80.6CRDKR-ND	http://www.digikey.com/products/en?keyword	0.1	0.2	X		
LMK212SD104JG-T	.1uF CAP	DIGIKEY	587-1133-1-ND	http://www.digikey.com/products/en?keyword	0.33	1.32	X		
					Total	**$90.41**			
					Starting Budget	$400.00		Spark Fun Spent	Sparkfun Left
					"Sparkfun Sponsorship"	$150.00		$76.85	$73.15
					First Purchase	$115.40			
					Budget Left after 1st Purchase	$434.60		College Spent	College Left
					This Purchase	$90.41		$128.96	$271.04
					New Budget Total	$344.19	$344.19		

Table 28: Parts Order Number 2



ECE PARTS REQUEST ORDERING FORM

Student Name: Jack Wolfe Email: jlw200@zips.uakron.edu

Project Name: S.W.I.M Date Submitted: **3/6/2017**

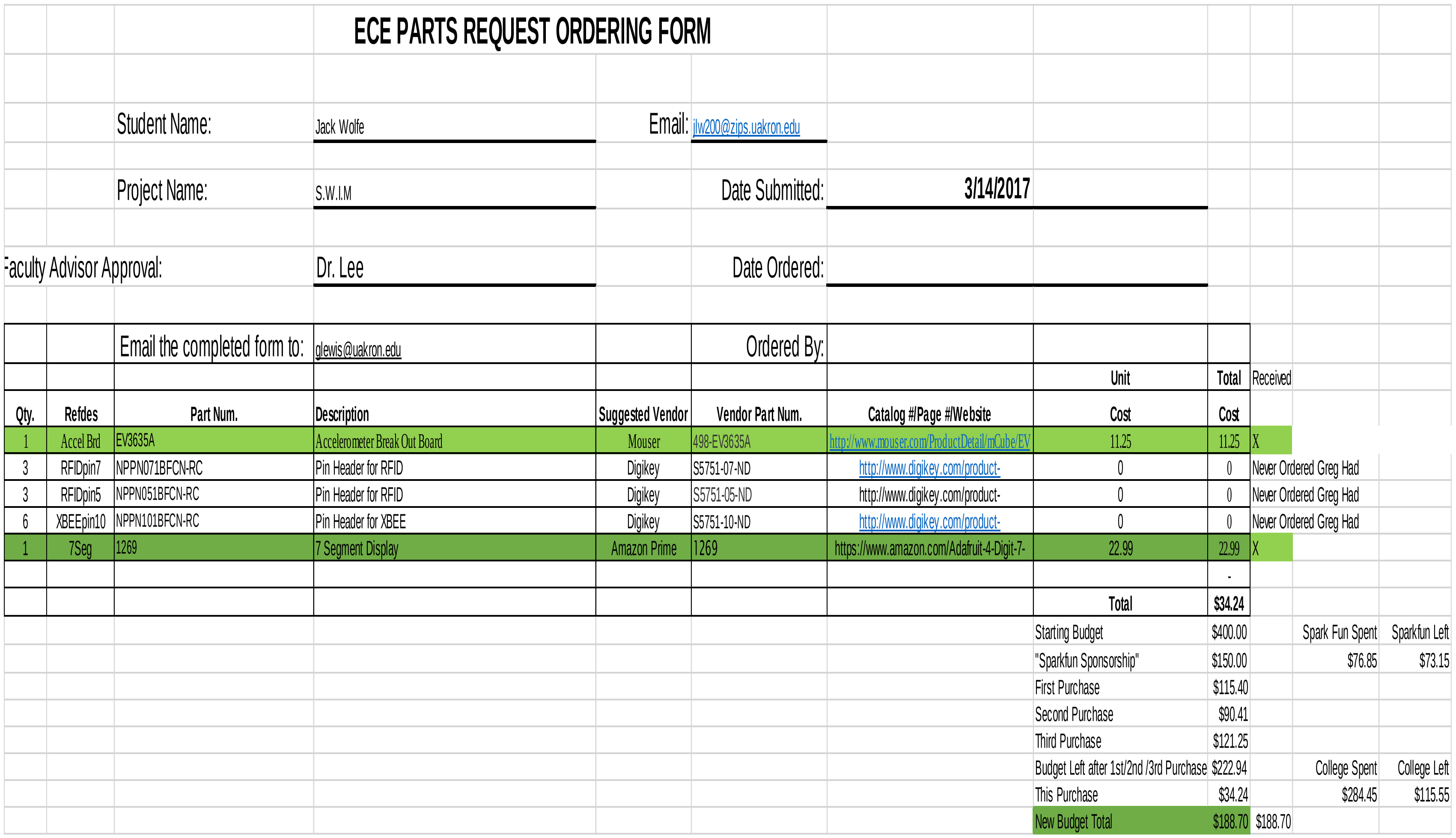
Faculty Advisor Approval: Dr. Lee Date Ordered:

Email the completed form to: glewis@uakron.edu Ordered By:

Qty.	Refdes	Part Num.	Description	Suggested Vendor	Vendor Part Num.	Catalog #/Page #/Website	Unit Cost	Total Cost	Received
1	Accel Brd	EV3635A	Accelerometer Break Out Board	Mouser	498-EV3635A	http://www.mouser.com/ProductDetail/mCube/EV	11.25	11.25	X
1	RFID Brd		RFID Breakout Board	OSH			10.55	10.55	X
1	RFID Tag	UT0712--04W10	RFID TAGS	Amazon	UT0712--04W10	https://www.amazon.com/UHPPOTE-	13.9	13.9	X
1	XBEE	XB24-BWIT-004	XBEE for mesh	DIGI	XB24-BWIT-004-ND	http://www.digikey.com/product-detail/en/digi-	21	21	X
1	Grom	A14121800UX0290	Gromet for water tight connections	Amazon	A14121800UX0290	https://www.amazon.com/Black-Plastic-	15.02	15.02	X
3	Ribbon Cab	1528-1162-ND	Ribbon Cable	DIGI	1528-1162-ND	http://www.digikey.com/product-	3.95	11.85	X
3	PinHead	61300511021	Angled pin headers for RFID board	DIGI	732-5338-ND	http://www.digikey.com/product-	0.39	1.17	X
12	RJ45con	54601-908WPLF	RJ45 Connector	DIGI	609-5081-ND	http://www.digikey.com/product-	0.5	6	X
2	DCDCconv	TC7660EOA	DC-to-DC converter	DIGI	TC7660EOA-ND	http://www.digikey.com/product-	0.81	1.62	X
3	U1	DSPIC33EP256MC506-H/PTVAO	Proccessor	Microchip	DSPIC33EP256MC506-H/PTV	http://www.microchipdirect.com/ProductDetail	6.59	19.77	X
1	Bat	B00S4UEEHI	battery	Amazon	B00S4UEEHI	https://www.amazon.com/uxcell-Leads-Switch-	8.31	8.31	X
3	J1	455-1704	Battery connector male	DIGI	455-1704-ND	http://www.digikey.com/product-detail/en/jst-	0.17	0.51	X
3	C1	455-1165	Battery connector Famale	DIGI	455-1165-ND	http://www.digikey.com/product-detail/en/jst-	0.1	0.30	X
								-	
								-	
							Total	$121.25	

Starting Budget	$400.00		Spark Fun Spent	Sparkfun Left
"Sparkfun Sponsorship"	$150.00		$76.85	$73.15
First Purchase	$115.40			
Second Purchase	$90.41			
Budget Left after 1st/2nd Purchase	$344.19		College Spent	College Left
This Purchase	$121.25		$250.21	$149.79
New Budget Total	$222.94	$222.94		

Table 29: Parts Order Number 3

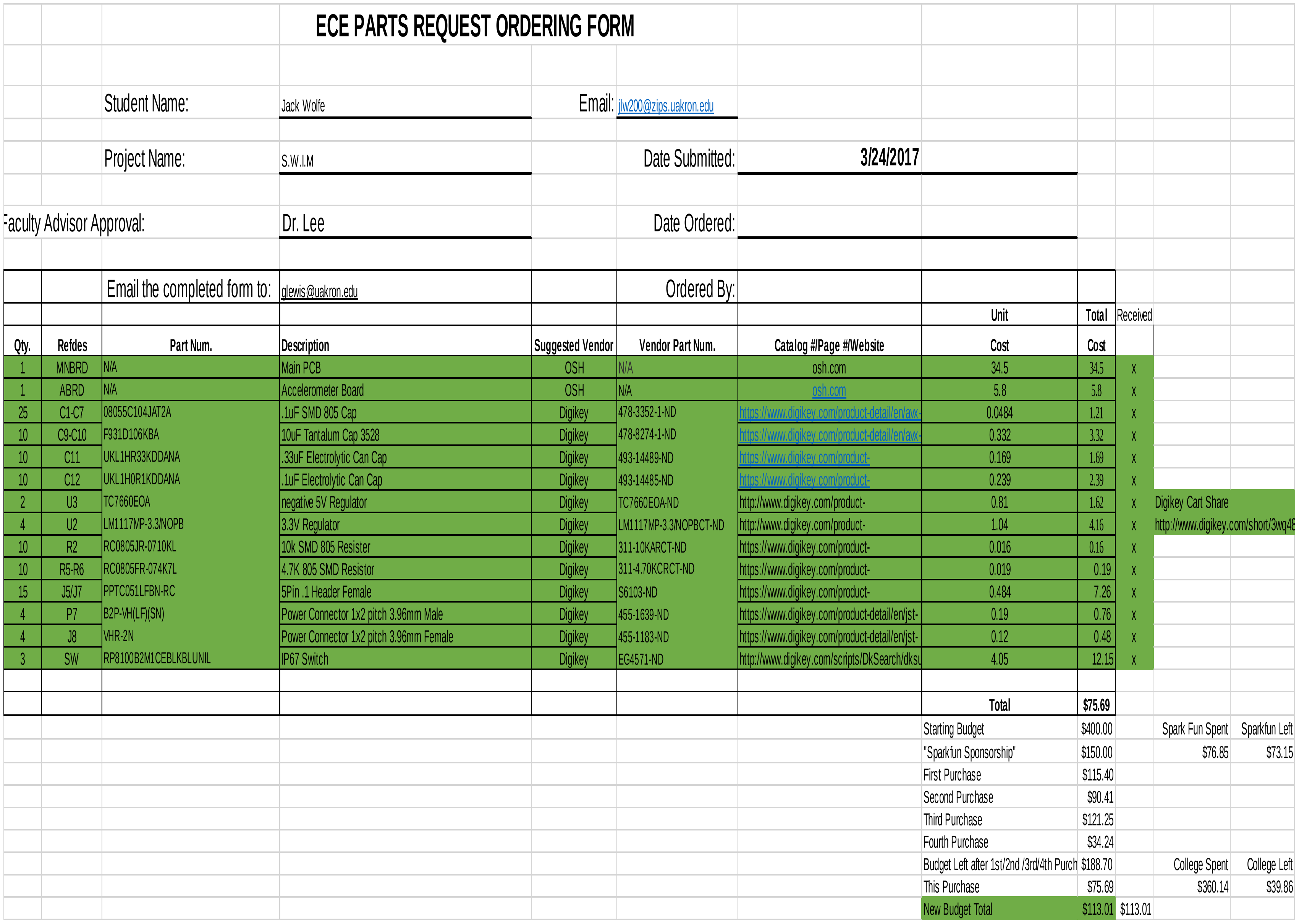


ECE PARTS REQUEST ORDERING FORM

Student Name:	Jack Wolfe	Email:	jlw200@zips.uakron.edu
Project Name:	S.W.I.M	Date Submitted:	3/14/2017
Faculty Advisor Approval:	Dr. Lee	Date Ordered:	
Email the completed form to:	glewis@uakron.edu	Ordered By:	

Qty.	Refdes	Part Num.	Description	Suggested Vendor	Vendor Part Num.	Catalog #Page #Website	Unit Cost	Total Cost	Received		
1	Accel Brd	EV3635A	Accelerometer Break Out Board	Mouser	498-EV3635A	http://www.mouser.com/ProductDetail/mCube/EV	11.25	11.25	X		
3	RFIDpin7	NPPN071BFCN-RC	Pin Header for RFID	Digikey	S5751-07-ND	http://www.digikey.com/product-	0	0	Never Ordered Greg Had		
3	RFIDpin5	NPPN051BFCN-RC	Pin Header for RFID	Digikey	S5751-05-ND	http://www.digikey.com/product-	0	0	Never Ordered Greg Had		
6	XBEEpin10	NPPN101BFCN-RC	Pin Header for XBEE	Digikey	S5751-10-ND	http://www.digikey.com/product-	0	0	Never Ordered Greg Had		
1	7Seg	1269	7 Segment Display	Amazon Prime	1269	https://www.amazon.com/Adafruit-4-Digit-7-	22.99	22.99	X		
								-			
							Total	$34.24			
							Starting Budget	$400.00		Spark Fun Spent	Sparkfun Left
							"Sparkfun Sponsorship"	$150.00		$76.85	$73.15
							First Purchase	$115.40			
							Second Purchase	$90.41			
							Third Purchase	$121.25			
							Budget Left after 1st/2nd /3rd Purchase	$222.94		College Spent	College Left
							This Purchase	$34.24		$284.45	$115.55
							New Budget Total	$188.70	$188.70		

Table 30: Parts Order Number 4



ECE PARTS REQUEST ORDERING FORM

		Student Name:	Jack Wolfe		Email:	jlw200@zips.uakron.edu					
		Project Name:	S.W.I.M		Date Submitted:	3/24/2017					
Faculty Advisor Approval:			Dr. Lee		Date Ordered:						
		Email the completed form to:	glewis@uakron.edu		Ordered By:						
							Unit	Total	Received		
Qty.	Refdes	Part Num.	Description	Suggested Vendor	Vendor Part Num.	Catalog #Page #Website	Cost	Cost			
1	MNBRD	N/A	Main PCB	OSH	N/A	osh.com	34.5	34.5	x		
1	ABRD	N/A	Accelerometer Board	OSH	N/A	osh.com	5.8	5.8	x		
25	C1-C7	08055C104JAT2A	.1uF SMD 805 Cap	Digikey	478-3352-1-ND	https://www.digikey.com/product-detail/en/avx-	0.0484	1.21	x		
10	C9-C10	F931D106KBA	10uF Tantalum Cap 3528	Digikey	478-8274-1-ND	https://www.digikey.com/product-detail/en/avx-	0.332	3.32	x		
10	C11	UKL1HR33KDDANA	.33uF Electrolytic Can Cap	Digikey	493-14489-ND	https://www.digikey.com/product-	0.169	1.69	x		
10	C12	UKL1H0R1KDDANA	.1uF Electrolytic Can Cap	Digikey	493-14485-ND	https://www.digikey.com/product-	0.239	2.39	x		
2	U3	TC7660EOA	negative 5V Regulator	Digikey	TC7660EOA-ND	http://www.digikey.com/product-	0.81	1.62	x	Digikey Cart Share	
4	U2	LM1117MP-3.3/NOPB	3.3V Regulator	Digikey	LM1117MP-3.3/NOPBCT-ND	http://www.digikey.com/product-	1.04	4.16	x	http://www.digikey.com/short/3wq48	
10	R2	RC0805JR-0710KL	10k SMD 805 Resister	Digikey	311-10KARCT-ND	https://www.digikey.com/product-	0.016	0.16	x		
10	R5-R6	RC0805FR-074K7L	4.7K 805 SMD Resistor	Digikey	311-4.70KCRCT-ND	https://www.digikey.com/product-	0.019	0.19	x		
15	J5/J7	PPTC051LFBN-RC	5Pin .1 Header Female	Digikey	S6103-ND	https://www.digikey.com/product-	0.484	7.26	x		
4	P7	B2P-VH(LF)(SN)	Power Connector 1x2 pitch 3.96mm Male	Digikey	455-1639-ND	https://www.digikey.com/product-detail/en/jst-	0.19	0.76	x		
4	J8	VHR-2N	Power Connector 1x2 pitch 3.96mm Female	Digikey	455-1183-ND	https://www.digikey.com/product-detail/en/jst-	0.12	0.48	x		
3	SW	RP8100B2M1CEBLKBLUNIL	IP67 Switch	Digikey	EG4571-ND	http://www.digikey.com/scripts/DkSearch/dksu	4.05	12.15	x		
							Total	$75.69			
							Starting Budget	$400.00		Spark Fun Spent	Sparkfun Left
							"Sparkfun Sponsorship"	$150.00		$76.85	$73.15
							First Purchase	$115.40			
							Second Purchase	$90.41			
							Third Purchase	$121.25			
							Fourth Purchase	$34.24			
							Budget Left after 1st/2nd /3rd/4th Purch	$188.70		College Spent	College Left
							This Purchase	$75.69		$360.14	$39.86
							New Budget Total	$113.01	$113.01		

Table 31: Parts Order Number 5

Project Schedules

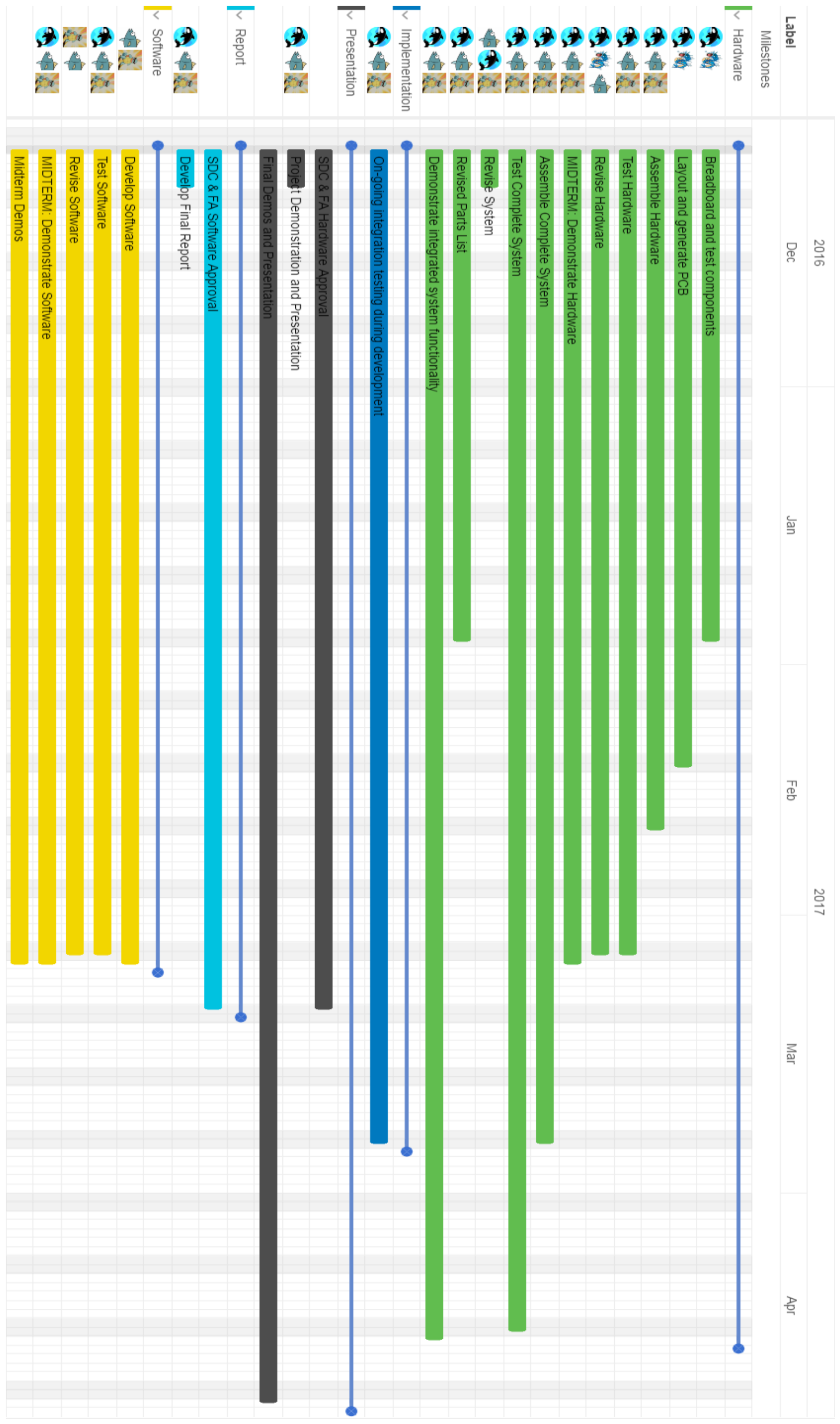


Figure 65: Implementation Gantt chart

Design Team Information

Dr. Kye Shin Lee | Faculty Advisor

Mark Archual | Team Archivist | Computer Engineer

Tanner Daniels | Hardware Lead | Electrical Engineer

Ethan Schweinsberg | Software Lead | Computer Engineer

Jackie Wolfe III | Team Lead | Electrical Engineer

Conclusions and Recommendations

The dynamics and chemistry of the team for this project were strong and this was key to the success of the project. Each member was willing to work outside their specific project responsibilities for the benefit of the group. There was also strong and constant communication between members about the status of key milestones. This project met its base design requirements, however, there is room for improved functionality. Future teams should consider: fine tuning the swimmer displacement calculation, improving the accuracy of the scale, enabling a more sophisticated ZigBee protocol that allows for communication between several modules and improves packet transmission reliability. Another hardware component that could be added is a visual sensor (i.e. IR, laser, etc.) that can detect when the buckets pass through a specific position. This can be paired with the accelerometer to have better start and stop detection times. This would also allow for specific distances to bet set and swam. Future students should start implement portions of the project that are the most difficult or unknown first, and maintain an aggressive integration schedule that allows extra for testing. A good plan is to implement each subsystem independently and ensure proper operation before integrating with other systems.

References

Mathworks,. *Accelerometer Orientation*. 2016. Web. 5 Dec. 2016.

Mathworks,. *Matlab Mobile Sensors*. 2016. Web. 5 Dec. 2016.

Science Kids,. *Pool Diagram*. 2016. Web. 5 Dec. 2016.

Tim Smith,. *Matlab Mobile*. 2016. Web. 5 Dec. 2016.

Osmani, Addy. *Developing Backbone.js Applications*. N.p.: O'Reilly Media, 2013. Print.

Appendices

All software and hardware design files can be found at: https://github.com/SDP-DT04

PartsOrderMaster.xlsx

Budget Excel File