Please take a moment to share how this work helps you through this survey. Your feedback will be important as we plan further development of our repository. Follow this and additional works at: http://ideaexchange.uakron.edu/honors_research_projects

Part of the Data Storage Systems Commons, Hardware Systems Commons, Signal Processing Commons, and the Systems and Communications Commons

Recommended Citation

http://ideaexchange.uakron.edu/honors_research_projects/476

This Honors Research Project is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.
System for Workout Information Management

Mark Archual
Ethan Schweinsberg

Department of Computer Engineering

**Honors Research Project**

Submitted to

*The Honors College*

Approved:

Kye-Shin Lee Date 04/24/17
Honors Project Sponsor (signed)

Kye-Shin Lee
Honors Project Sponsor (printed)

Kye-Shin Lee Date 04/26/17
Reader (signed)

Kye-Shin Lee
Reader (printed)

[Signature]

Date 04/26/17
Reader (signed)

Kye-Shin Lee
Reader (printed)

Accepted:

[Signature]

Date 4/27/17
Department Head (signed)

Joan Carletta
Department Head (printed)

[Signature]

Date 04/26/2017
Honors Faculty Advisor (signed)

Ngan Tran
Honors Faculty Advisor (printed)

[Signature]

Dean, Honors College
My main contribution to the System for Workout Information Management was the design and implementation of the web application. The goal of our project was to design a data system that would record various metrics of a swimmer’s workout when using a power rack. To view the recorded data, and interact with the physical system, a web application was designed for use by the coaching staff and swimmers of The University of Akron Women’s Swim team. My responsibilities for the web application included both design and realization of the entire web stack. My teammates focused on the hardware aspects of the design.

The website was designed with NodeJS as the core server architecture. NodeJS was chosen because it provides an asynchronous non-blocking IO based execution structure that was perfect for our application. Essentially, this allows the website to process a lot of user input without slowing down its responsiveness. NodeJS was complimented with a MongoDB database (to store workout information) and a front-end client application that was driven by Backbone. I was responsible for the design of the server and client applications. I collaborated with Ethan Schweinsberg on the database structure and design. The website was physically hosted on a Pine64, which is a low-cost computing platform. The functionality of the website was tested extensively during its design, as well as with the swim team at the ONAT and the during the ECE department’s senior design demonstration day on April 24th, 2017.

Functionality was added to the website through a number of libraries and add-ons that are developed open-source and for free use. Some of these libraries include: Backbone, MongooseJS, and ChartJS. These libraries provide our website with the capability of performing tasks that NodeJS does not support. I was in charge of researching and choosing what libraries we would use, and then implementing them in the design. This was key to the success of the website, as the design of the web application is directly related to the supporting libraries that are chosen.

I also technically supported the design of the project through the design and implementation of an integration algorithm that was key to measuring the distance swam during one repetition of the power rack. Our design utilizes an accelerometer to detect when the swimmer starts and stops a repetition with the power rack. Working alongside Ethan Schweinsberg, we tested various integration methods and determined a process for determining the swimmer’s displacement in the pool based on the acceleration of the weight being pulled. This process is detailed in the report, but it essentially involves a double application of the midpoint integration technique and linear least squares approximation method. This process also reduced the amount of data that was stored in our database per record, yielding a more responsive web application and more reliable data acquisition methods.

Lastly, the Electrical and Computer Engineering department has each design team assign roles for its members. I was the team archivist. My responsibilities included: note taking at meeting with our faculty advisor, coordinating timelines for project milestones, preparing presentations and editing written reports.

-Mark Archual
I worked on the System for Workout Management (S.W.I.M) project as a part of a team of four members. Our goal was to design a system for collecting workout data for the University of Akron Women’s Swimming Team. My main contributions to the project are firmware design and wireless communication.

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 implements algorithms to sense the start time of the swimmer, the stop time of the swimmer, and weight changes of the power rack. Specifically, I wrote the code for the accelerometer, wireless communication, and detection algorithms. With help from the rest of the team, I also worked on integrating all of the different pieces of firmware. To run the system, the firmware rotates through different states. The different states include: waiting for RFID scan, waiting for weight measurement, waiting for swimmer to start, and waiting for the swimmer to stop.

For wireless communication, we used XBees to send packets of data from the sensor board to the server. I worked on getting two XBees to communicate and developed a packet structure for sending data. Also, I wrote code on the server to receive data and store it to a database. To get this to work, I had to spend time figuring out what the ideal packet length should be to reduce the amount of data loss. I also performed tests to make sure that the XBees could communicate across the range of the pool deck. I worked with Mark Archual to ensure that data could reliably get from the sensor board to the server.

Officially, my role on the team was Software Manager. To fulfill this duty, I helped make a GitHub repository for everyone to store their code. I also contributed to testing the system at the pool and making improvements.

-Ethan Schweinsberg
S.W.I.M.
(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

12/7/16
# Contents

Article I. Abstract ........................................................................................................ 7

Section 1.01 Problem Statement .............................................................................. 7

(a) Need .................................................................................................................. 7

Section 1.01 Objective ............................................................................................ 7

Section 1.02 Background ......................................................................................... 7

(a) Patent Search .................................................................................................... 7
(b) Article Search ..................................................................................................... 8
(c) Other Source Search .......................................................................................... 9

Section 1.03 Marketing Requirements .................................................................... 9

Section 1.04 Objective Tree .................................................................................... 10

Section 1.05 Demonstration of Ideas ...................................................................... 10

Article II. Design Requirements Specifications ....................................................... 11

Section 1.01 Web Server and Application ................................................................ 11

Section 1.02 Sensor and Communication Board ..................................................... 12

Article III. Accepted Technical Design .................................................................. 14

Section 1.01 Hardware Design ............................................................................... 14

(a) Theory of Operation ......................................................................................... 14
(b) Block Diagrams .................................................................................................. 14
(c) Sensor Board ...................................................................................................... 18
(d) Accelerometer .................................................................................................... 20
(e) Weight Measurement ........................................................................................ 21
(f) RFID System ....................................................................................................... 24
(g) Communication Module ..................................................................................... 25
(h) Display Module .................................................................................................. 27
(i) Overall Power Consumption .............................................................................. 28
(j) Bucket Weight ..................................................................................................... 28

Article IV. Software Design ..................................................................................... 29

Section 1.01 Theory of Operation ............................................................................. 29

(a) Firmware ........................................................................................................... 29
(b) Using Acceleration Data to Determine Athlete Position .................................... 33
(b) Filter design .................................................................................................................. 39
(c) Web Application .......................................................................................................... 45
(d) Block Diagrams ............................................................................................................ 50

Article V. Parts List ............................................................................................................ 58
Article VI. Project Schedules .............................................................................................. 59
Article VII. Design Team Information .................................................................................. 60
Article VIII. Conclusions and Recommendations .............................................................. 61
Article IX. References .......................................................................................................... 62
Article X. Appendices ........................................................................................................... 63
Figure 29: Matlab Code for Polynomial Approximation .......................................................... 36
Figure 30: Linear curve fit mapping bucket height to swimmer distance ................................ 37
Figure 31: Swimmer distance curve after applying linear relationship .................................. 38
Figure 32: 10th order polynomial approximation of swimmer distance curve ......................... 39
Figure 33: Unfiltered accelerometer data .................................................................................. 40
Figure 34: Frequency Spectrum of Unfiltered Accelerometer Data .......................................... 40
Figure 35: Frequency response of 2nd order Butterworth filter .............................................. 41
Figure 36: Frequency response of 2nd order Butterworth filter .............................................. 42
Figure 37: Frequency spectrum of acceleration data after 2nd order Butterworth filter .......... 43
Figure 38: Frequency spectrum of acceleration data after 4th order Butterworth filter .......... 43
Figure 39: Acceleration data after 2nd order Butterworth filter ............................................ 44
Figure 40: Acceleration data after 4th order Butterworth filter ............................................ 44
Figure 41: Swimmer/Id mapping entry example ....................................................................... 45
Figure 42: Workout data entry example .................................................................................... 45
Figure 43: Data amounts with and without polynomial approximation .................................... 46
Figure 44: Web application flow diagram .................................................................................. 47
Figure 45: Node event loop ....................................................................................................... 48
Figure 46: Web application file structure .................................................................................. 48
Figure 47: Pseudocode of routing using Express ..................................................................... 48
Figure 48: Pseudocode for data retrieval using MongoJS ....................................................... 49
Figure 49: Pseudocode for XBEE communication with Javascript .......................................... 49
Figure 50: Diagram showing Coach Peresie interacting with the Backbone Framework .......... 50
Figure 51: Level 0 Software Block Diagram .......................................................................... 50
Figure 52: Level 1 Software Block Diagram .......................................................................... 51
Figure 53: Level 2 Embedded Software Block Diagram .......................................................... 52
Figure 54: Level 2 Web Application Block Diagram ............................................................... 54
Figure 55: Level 3 Embedded Software Block Diagram .......................................................... 55
Figure 56: Web Application Level 3 Block Diagram ............................................................... 57
Figure 57: Implementation gantt chart ..................................................................................... 59
Table 1: Web Server and Application Design Requirement Specifications .................................. 12
Table 2: Sensor and Communication Board Design Requirement Specifications ....................... 13
Table 3: Sensor Acquisition Level 0 Module .................................................................................. 14
Table 4: Server/Database Level 0 Module .................................................................................... 14
Table 5: Voltage Regulator Level 1 Module .................................................................................. 15
Table 6: 7-Segment Display Level 1 Module ................................................................................ 16
Table 7: Accelerometer Level 2 Module ....................................................................................... 16
Table 8: Load Cell Level 2 Module .............................................................................................. 16
Table 9: RFID Level 2 Module ..................................................................................................... 17
Table 10: Microprocessor Level 2 Module ..................................................................................... 17
Table 11: Communication Level 2 Module .................................................................................... 17
Table 12: Relative error results for second order Butterworth Filter ........................................... 35
Table 13: Relative error results for fourth order Butterworth Filter ............................................ 35
Table 14: User Application Level 0 Module .................................................................................. 50
Table 15: Embedded Software Level 1 Module ............................................................................. 51
Table 16: Web Application Level 1 Module .................................................................................. 51
Table 17: Wireless Communication Interface Level 2 Module ..................................................... 52
Table 18: Signal Processor Level 2 Module .................................................................................. 52
Table 19: RFID Interface Level 2 Module .................................................................................... 53
Table 20: Main Software Loop Level 2 Module ........................................................................... 53
Table 21: User Interface Level 2 Module ...................................................................................... 54
Table 22: Server Level 2 Module ................................................................................................. 55
Table 23: Database Level 2 Module .............................................................................................. 55
Table 24: A/D Converter Level 3 Module ..................................................................................... 56
Table 25: Digital Filter Level 3 Module ........................................................................................ 56
Table 26: Start and Stop Detection Algorithm ............................................................................. 56
Table 27: Parts List ....................................................................................................................... 58
Article I. 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.

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.

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.

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. 2015/0133748A1. In this patent, a wearable device is described that contains various networking technology (RFID, bluetooth, wifi, 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.

(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.

(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.

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]
Section 1.04  Objective Tree

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

![Objective Tree Diagram]

Figure 1: Objective Tree

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]
### Article II. 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.

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

<table>
<thead>
<tr>
<th>Marketing Requirements</th>
<th>Engineering Requirements</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Must be able to withstand ingress of water up to 1m of submersion for a time duration of 30 minutes</td>
<td>Sensor boards will be attached to weight rack on pool deck.</td>
</tr>
<tr>
<td>3</td>
<td>Must be able to communicate at a distance of up to 50m</td>
<td>The calculated max distance for the furthest workout area is 49.5m.</td>
</tr>
<tr>
<td>1</td>
<td>Must be able to measure weight (up to 100kg or 220lbs), as well as, distance swam, and time per repetition.</td>
<td>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.</td>
</tr>
<tr>
<td>1, 6</td>
<td>The device must be able to display weight, repetition time, and distance swam to an accuracy of 1 pound, 100 ms.</td>
<td>The swimmers will need to see the weight used before starting each repetition. Also the swimmers would like to</td>
</tr>
<tr>
<td>5.</td>
<td>The device must have a portable power source.</td>
<td>The device needs to be portable and freestanding.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>3.</td>
<td>The sensor board must autonomously update user information to server and web application.</td>
<td>The swimmers should focus on their workout, and the coaching staff on evaluating the swimmers, not operating the data system.</td>
</tr>
</tbody>
</table>

**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*
Article III. 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 web server. 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. This information can then be accessed via a web application.

(a) Block Diagrams

1. Level 0

![Block Diagram]

Figure 2: Hardware Level 0 Block Diagram

<table>
<thead>
<tr>
<th>Module</th>
<th>Sensor Acquisition Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
<tr>
<td>Inputs</td>
<td>• 6V DC Battery Pack</td>
</tr>
<tr>
<td>Outputs</td>
<td>• Interpreted Sensor Data</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 3: Sensor Acquisition Level 0 Module

<table>
<thead>
<tr>
<th>Module</th>
<th>Server/Database Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
<tr>
<td>Inputs</td>
<td>• Interpreted Sensor Data, • 120V AC Outlet</td>
</tr>
<tr>
<td>Outputs</td>
<td>• Web Application Displayed Data</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 4: Server/Database Level 0 Module
2. Level 1

![Figure 3: Sensor Acquisition Unit Level 1 Block Diagram](image)

<table>
<thead>
<tr>
<th>Module</th>
<th>Voltage Regulator</th>
<th>7-Segment Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
<tr>
<td>Inputs</td>
<td>• 6V DC Battery Pack</td>
<td>• 5V DC</td>
</tr>
<tr>
<td>Outputs</td>
<td>• ±5V DC</td>
<td>• Displayed Information</td>
</tr>
<tr>
<td>Outputs</td>
<td>• +3.3 DC</td>
<td>• Data to be displayed</td>
</tr>
<tr>
<td>Description</td>
<td>The voltage regulator will take in 6V 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.</td>
<td>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</td>
</tr>
</tbody>
</table>

Table 5: Voltage Regulator Level 1 Module

Table 5: 7-Segment Display Level 1 Module
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**

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

**Table 7: Accelerometer Level 2 Module**

<table>
<thead>
<tr>
<th>Module</th>
<th>Load Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
<tr>
<td>Inputs</td>
<td>• Regulated Voltage (±5V)</td>
</tr>
<tr>
<td>Outputs</td>
<td>• Sensor voltage</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Table 8: Load Cell Level 2 Module**
<table>
<thead>
<tr>
<th>Module</th>
<th>RFID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
<tr>
<td>Inputs</td>
<td>• Regulated Voltage</td>
</tr>
<tr>
<td>Outputs</td>
<td>• RFID tag data</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

*Table 9: RFID Level 2 Module*

<table>
<thead>
<tr>
<th>Module</th>
<th>Microprocessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
</tbody>
</table>
| 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*

<table>
<thead>
<tr>
<th>Module</th>
<th>Communication Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Tanner Daniels/Jack Wolfe</td>
</tr>
</tbody>
</table>
| 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*
(b) Sensor Board

The sensor board contains the microprocessor, RFID module, communication module, and the display module. The sensor board will be housed in a water tight container, which also connects to the accelerometer, and load cell via watertight wired connection. The microprocessor communicates to all of the sensors and receives raw data back to interpret. It then transmits information from the communication module to the web server. This information is then analyzed by the web server and displayed on the web application for a user to view.

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 our system is to update with in 5 seconds of a workout. This processor also contains 2 UART, 2 I2C, and 2 SPI communication ports. Since 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.

![Figure 5: Eagle CAD dsPIC33EP256MC506 circuit Schematic](image-url)
Figure 6: Eagle CAD Power Rail Circuit Schematic

Figure 7: Eagle CAD Connector Schematic
(c) Accelerometer

The accelerometer chosen is a MC3635A 3-axis accelerometer. This particular accelerometer was chosen as the sampling frequency was from 3-1300 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 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 Host machine 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]

(d) 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 two load cells as the left and right branches of the bridge, shown in Figure 9. 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 9.

![Figure 9: Example of linear relationship of the change in voltage vs. the applied load](image-url)
The full Wheatstone bridge circuit can be seen below as

![Wheatstone Bridge Diagram](image)

Figure 10. The voltage that is measured is the voltage across the nodes V1 and V2.

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 11: Amplification circuit.

![Amplification Circuit Diagram](image)

Figure 11: Load Cell Full Wheatstone Bridge Configuration

We chose to have a gain of about $4000\left(\frac{\sqrt{V}}{V}\right)$, as this coupled with our microprocessors 12 bit ADC gives us accuracy down to 0.6µV.
A three stage amplification circuit 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:

\[ Gain_{DA} = \frac{R_2}{R_1} \Rightarrow \frac{2700\Omega}{47\Omega} \]

The Inverting Amplifier stage gain is determined by the equation:

\[ Gain_{IA} = -\frac{R_4}{R_3} \Rightarrow -\frac{3300\Omega}{47\Omega} \]

Therefore the overall gain of the amplification circuit is determined by:

\[ Gain_{Overall} = Gain_{DA} \times Gain_{IA} = \left(\frac{R_2}{R_1}\right)\left(\frac{R_4}{R_3}\right) \Rightarrow \frac{2700\Omega}{47\Omega} \times \frac{3300\Omega}{47\Omega} = \frac{-4033.499321}{(47\Omega)^2} \]

Our PCB designed amplifier is shown in figure D.

Figure 11: Amplification circuit
In order to properly weigh the buckets weight, a platform is designed and constructed. The platform consists of a bottom plate, the middle area containing 2 load cells and 2 overweight safety brackets, and a top plate. The design creates a heavy duty scale to weigh the buckets. The scale can also be fixed to the bucket bracket if desired to allow the swimmers to not have to install it every time.

(e) 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 set up to that individual swimmer. This will then log the data recorded at that machine to a certain swimmer, inside the database, so the user may be able to track their progress using the web application.

The RFID reader chosen to be used is an ID-12LA. This module has an integrated antenna, range of about 2 inches, and communicates via I2C. The main reason we chose this module was because it has a small compact design making it simpler to implement, as well as, make water resistant.
(f) 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 14. Another drawback of Bluetooth is that it can only support up to 7 active...
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:

\[ a^2 + b^2 = c^2 \]

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 on to be used is 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 SX module will be used to utilize the Zigbee communication protocol. This XBee module is a surface mount device, low power (max 55mA), and communicates at minimum 1.5 miles.

(g) Display Module

A display will be used to indicate informative details to the swimmers while in the water. The information being displayed will be the weight, when in weight reading mode, as well as the overall time the swimmer was swimming for 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.

A 7-segment display will be used to display all of the information. A 7-segment display was chosen as it has 7 meters of visibility and four digits to allow for accurate time (ss:ms) as well as weight up to 9999 lbs.
(h) **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 using 4AA alkaline batteries in series. The standard alkaline battery has 1800-2600[mAh] of battery storage.

\[
\text{BatteryLife}_{\text{Min}} = \frac{1800[\text{mAh}]}{\sum I_{\text{sensors}}} \\
\Rightarrow \frac{1800[\text{mAh}]}{I_{\text{Microprocessor}} + I_{RFID} + I_{\text{Accelerometer}} + I_{\text{XBEE}} + I_{\text{LoadCells}} + I_{\text{Display}}} \\
\Rightarrow \frac{1800[\text{mAh}]}{60[\text{mA}] + 35[\text{mA}] + 36[\mu\text{A}] + 55[\text{mA}] + 1.25[\text{mA}] + 500[\text{mA}]} \Rightarrow \frac{1800[\text{mAh}]}{651.286[\text{mA}]} \\
\Rightarrow 2.763762771[\text{hours}] \\
\]

\[
\text{BatteryLife}_{\text{Max}} = \frac{2600[\text{mAh}]}{\sum I_{\text{sensors}}} \\
\Rightarrow \frac{2600[\text{mAh}]}{I_{\text{Microprocessor}} + I_{RFID} + I_{\text{Accelerometer}} + I_{\text{XBEE}} + I_{\text{LoadCells}} + I_{\text{Display}}} \\
\Rightarrow \frac{2600[\text{mAh}]}{60[\text{mA}] + 35[\text{mA}] + 36[\mu\text{A}] + 55[\text{mA}] + 1.25[\text{mA}] + 500[\text{mA}]} \Rightarrow \frac{2600[\text{mAh}]}{651.286[\text{mA}]} \\
\Rightarrow 3.99210178[\text{hours}] \\
\]

(i) **Bucket Weight**

Using both load cells in opposite corners of the scale, the weight is evenly distributed therefore the max weight that 2-50kg sensors can measure is 100kg, or \(\approx 220\) pounds.

- 25 gallon bucket filled full of water:
- Weight of water per gallon \(\approx 8.36\) [pounds/gallon]
- \(25 \times 8.36 = 209\) pounds
- \(25[\text{gallons}] \times 8.36[\frac{\text{pounds}}{\text{gallons}}] = 209[\text{pounds}]\)
Article IV. Software Design

Section 1.01 Theory of Operation

(a) 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 will be designed as a state machine consisting of two states, ‘Swimming’ and ‘Not Swimming’. Upon power-up, the firmware will default to the ‘Not Swimming’ state. In this state, the firmware will cycle between obtaining the weight of the power rack, checking the RFID interface for user interaction, and reading the accelerometer. This code will also implement an algorithm to determine if the swimmer has pushed off the wall and begun swimming. Figure 17 shows the functionality of this state in detail.

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 18 shows this functionality in detail.
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. For example, an interrupt service routine will handle collecting data from the accelerometer every 10 ms. The ISR will perform different tasks based on the corresponding state of the firmware. Figure 19 shows pseudocode for how this will work.

```c
// 10 ms interrupt routine for accelerometer
ISR()
{
    accel_val = read_accel_12c();

    if (isSwimming)
    {
        transmit(accel_val);

        if(done_swimming())
        {
            update_display();
            transmit(Id);
            isSwimming = False;
        }
    }
    else if (push_off_wall())
    {
        isSwimming = True;
    }
}
```
The same concept can be applied to the other peripherals. For determining the weight of the power rack the firmware will have an ISR to check the voltage of the load cell. Also, this same ISR will check if the RFID interface was activated. If the swimmer is swimming, no code in this routine is actually executed since the weight is not going to change and the user is not going to switch machines during a workout. Figure 20 shows pseudocode that can be used to implement this.

```c
// 100 ms interrupt for load cell
ISR()
{
    If (!isSwimming)
    {
        If (bucket_weight_change())
        {
            bucket_weight = read_a2d();
            update_display(bucket_weight);
        }
        If (rfid_activated())
        {
            Id = read_tag();
            update_swimmer(Id);
        }
    }
}
```

Finally, an ISR will be used to handle wireless communication via the XBee. The pseudocode below shows how data will be transmitted. If large packets of data need to be sent, they will be stored in a buffer and transmitted over time through this ISR. This way, communication will not block other processes. See Figure 21 below for an example of this.

```c
// 10 ms interrupt transmit
ISR()
{
    if (is_new_tx_data())
    {
        tx_byte = get_tx_byte();
        transmit_byte(tx_byte);
    }
}
```

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 22 shows data taken from the beginning of a 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 2.6039m/s². Similarly, it can be shown that the acceleration clearly dampens at the end of the workout. See Figure 23 for a plot of the acceleration at the end of a swim.
This dampening happens at about 30s, which is when the swimmer starts to return to the wall. A spike-detection algorithm can be implemented in real time on the microcontroller to detect these scenarios.

(a) Using Acceleration Data to Determine Athlete Position

As per marketing requirements, the system needs to determine position of an athlete at intervals during the exercise. It is proposed that 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:

\[
\begin{align*}
    v(t) &= \int a(t) dt \\
    x(t) &= \int v(t) dt
\end{align*}
\]

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.

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.
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 25 shows how the data acquisition was controlled in Matlab:

```
Start 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
m.SampleRate

Stop Script

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
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 26: 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.
Table 12: Relative error results for second order Butterworth Filter

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

Table 13: Relative error results for fourth order Butterworth Filter

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

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 27: 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:

\[ y = m \times x + B \]

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 by simply evaluating a polynomial over the appropriate time interval. This second approximation will help greatly simplify the database in both complexity and size.

```matlab
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 29: Matlab Code for Polynomial Approximation

Figure 30 shows the relationship between the swimmer’s displacement in the pool and the bucket’s height using the linear least squares approximation method.
This fit yielded an $R^2$ value of 0.9986. This relationship, $0.42251 + 11.2707 \times x$ 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 31 for results.
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 32: 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’.

(b) 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 33, a plot of acceleration data that was collected during a trial.
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.
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.

![2nd Order Butterworth Filter Frequency Response](image)

*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 37: Frequency spectrum of acceleration data after 2nd order Butterworth filter

Figure 38: 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 39: Acceleration data after 2nd order Butterworth filter

Figure 40: Acceleration data after 4th order Butterworth filter
Depending on the type of data, either one of these filters will be implemented on the server to process the acceleration data before it is numerically integrated. Since longer workouts were more accurate with the 4th order filter, the 4th order filter will be used for data from workouts lasting longer than 10 seconds. Otherwise, the 2nd order filter will be used. This will result in smoother acceleration curves, which will increase the accuracy of the numerical integration.

(MJA,EES)

(c) 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, that houses workout information. Finally, a front-end will be designed with MarionetteJS that will provide a dynamic interface for the end-user.

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 in real time. 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.

The database houses two main collections. A collection stores data following a document

Swimmer/ID Mapping Entry

```
{
  "Name": "Jack Wolfe",
  "ID": "00-14-2F-01-2A-45"
}
```

Figure 41: Swimmer/Id mapping entry example

Workout Data Entry

```
{"Date": "2016-10-16T07:45:30",
 "ID": "00-14-2F-01-2A-45",
 "Weight": 30.4,
 "Time": 9.93400001525879,
 "Position":[
 0.346622211336391,
 0.696716129559691,
 -1.20125256121348,
 0.736965774800871,
 0.0692168558302586,
 -0.171449618345341,
 0.0615930469844533,
 -0.818855281252946,
 0.0106191126232948,
 -0.008054813219564667,
 0.000117206584831277]```
model approach (like json). One collection, ‘workouts’, will be used to store information from a repetition of the weight machine. A second collection, ‘ID’ will also be required to map a swimmer’s name to a unique RFID tag. See Figure 41 and Figure 42 for examples of each collection’s structure.

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 43 below.

<table>
<thead>
<tr>
<th>With Polynomial Approximation</th>
<th>Without Polynomial Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data amount per entry:</strong></td>
<td><strong>Data Amount per entry:</strong></td>
</tr>
<tr>
<td>• (62) characters for variable names</td>
<td>• (20 * 100) 16-bit acceleration values</td>
</tr>
<tr>
<td>• (1) 64-bit float for time</td>
<td>• (20 * 100) 16-bit timestamps</td>
</tr>
<tr>
<td>• (10) 64-bit floats for position</td>
<td>• (62) characters for variable names</td>
</tr>
<tr>
<td>Total: 84-bytes</td>
<td>Total: 8062-bytes</td>
</tr>
</tbody>
</table>

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

Figure 43: Data amounts with and without polynomial approximation

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 44 describes the overall execution of the web server, each component of which will be explained accordingly.
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, which 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.
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, Webpack, ExpressJS, MongoJS (not to be confused with the database itself), 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 will be relatively straightforward, as we will only require the source code and the package.json file.

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 will follow this structure, which is shown to the right.

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 types into a web browser. The routes will be defined in app.js, which will then call the corresponding file to execute in the ‘routes’ folder. For example, the endpoint of a route of the application can be defined as /data. The corresponding code in app.js then is:

```javascript
var express = require('express');
var data = require('./routes/data'); // Route definition for database connection
var app = express();
app.use('/data', data); // Route to database
```

With a route defined to /data, a corresponding data.js file is needed to contain code that will be executed. The MongoJS library will be used for connecting to, and querying, Mongo Databases. The corresponding logic in data.js is:

```javascript

```
```javascript
var express = require('express');
var router = express.Router();
var mongojs = require('mongojs'); //Needed to connect to the database

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

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

router.get('/', function(req, res, next) {
    //start of callback function
    db.data.find(function(err, docs){
        //get all the information and put it into docs
        if (err) return;
        res.json(docs);
    }); //end callback
});

module.exports = router;
```

Figure 48: Pseudocode for data retrieval using MongoJS

More sophisticated examples can be built from this basic structure. Notably, this design chooses the MogoJS library instead of the MongoDB library because the program syntax is more straightforward.

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 examples show basic communication with an XBee radio:

```
var xbee_api = require('xbee-api');
var C = xbee_api.constants;
var xbeeAPI = new xbee_api.XBeeAPI();

// Something to send
var frame_obj = {
    type: C.FRAME_TYPE.AT_COMMAND,
    command: "NI",
    commandParameter: [ ],
};
console.log(xbeeAPI.buildFrame(frame_obj));

// Something to receive
var raw_frame = new Buffer([ 0x7E, 0x00, 0x13, 0x97, 0x55, 0x00, 0x13, 0xA2, 0x00, 0x40, 0x52, 0x2B, 0xAA ]);
console.log(xbeeAPI.parseFrame(raw_frame));
```

Figure 49: Pseudocode for XBEE communication with Javascript

3. Front-end

The front-end of the web application will be supported by Backbone.JS. 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.

[MJA]
A ‘collection’ helps manage views with several models. For example, a view may contain a button and a table of information. Separate models may be required to manage the logic for the button and the table. In this scenario a collection would be a natural fit to manage each.

Backbone is a natural fit for this application because the website should be updated as quickly as possible. By structuring the web application this way, a framework is in place for fluid data updates.

A few more libraries are needed to fully flesh out the web application. Marionette is a library that extends the BackboneJS library to help abstract and simplify design of Backbone applications for the programmer. ChartJS provides support for displaying workout data in dynamic charts. The templating engine, Handlebars, will be used to help manage the structure of the front-end code. Lastly, Webpack will be used to build our code into an executable file.

**(d) Block Diagrams**

**Level 0**

![User Application Block Diagram](image)

*Figure 51: Level 0 Software Block Diagram*

<table>
<thead>
<tr>
<th>Module</th>
<th>User Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Mark Archual</td>
</tr>
<tr>
<td>Inputs</td>
<td>User Input, Sensor Input</td>
</tr>
<tr>
<td>Outputs</td>
<td>Raw Data Files, Interpreted Results</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

*Table 14: User Application Level 0 Module*
Level 1

Figure 52: Level 1 Software Block Diagram

<table>
<thead>
<tr>
<th>Module</th>
<th>Embedded Software Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td>Inputs</td>
<td>Wireless communication, Athlete input, Sensor values</td>
</tr>
<tr>
<td>Outputs</td>
<td>Wireless communication, Processed sensor values, Display control</td>
</tr>
<tr>
<td>Description</td>
<td>The embedded software application handles gathering data from sensors, digital filtering, and wireless communication.</td>
</tr>
</tbody>
</table>

Table 15: Embedded Software Level 1 Module

<table>
<thead>
<tr>
<th>Module</th>
<th>Web Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Mark Archual</td>
</tr>
<tr>
<td>Inputs</td>
<td>Processed Sensor Values</td>
</tr>
<tr>
<td>Outputs</td>
<td>Raw Data Files, Interpreted Results</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

Table 16: Web Application Level 1 Module
Level 2: Embedded Software

![Figure 53: Level 2 Embedded Software Block Diagram]

<table>
<thead>
<tr>
<th>Module</th>
<th>Wireless Communication Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td>Inputs</td>
<td>Compiled exercise data</td>
</tr>
<tr>
<td>Outputs</td>
<td>Server messages</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

*Table 17: Wireless Communication Interface Level 2 Module*

<table>
<thead>
<tr>
<th>Module</th>
<th>Signal Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td>Inputs</td>
<td>Raw acceleration data, Strain gauge signal</td>
</tr>
<tr>
<td>Outputs</td>
<td>Processed data</td>
</tr>
<tr>
<td>Description</td>
<td>The signal processor is responsible for processing raw sensor outputs into meaningful data.</td>
</tr>
</tbody>
</table>

*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>
<thead>
<tr>
<th>Module</th>
<th>Main Software Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Designer</strong></td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td><strong>Inputs</strong></td>
<td>Server messages, processed data, Athlete identification information</td>
</tr>
<tr>
<td><strong>Outputs</strong></td>
<td>Compiled exercise data</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

---
**Level 2: Web Application**

**Figure 54: Level 2 Web Application Block Diagram**

<table>
<thead>
<tr>
<th>Module</th>
<th>User Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Mark Archual</td>
</tr>
<tr>
<td>Inputs</td>
<td>Interpreted Data</td>
</tr>
<tr>
<td>Outputs</td>
<td>User Input</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

*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**

![Level 3 Embedded Software Block Diagram](image)

Figure 55: Level 3 Embedded Software Block Diagram
<table>
<thead>
<tr>
<th>Module</th>
<th>A/D Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td>Inputs</td>
<td>Analog strain gauge signal</td>
</tr>
<tr>
<td>Outputs</td>
<td>Digital strain gauge signal</td>
</tr>
<tr>
<td>Description</td>
<td>The A2D Converter converts the analog signal from the strain gauge into a digital value that accurately represents the amount of strain.</td>
</tr>
</tbody>
</table>

*Table 24: A/D Converter Level 3 Module*

<table>
<thead>
<tr>
<th>Module</th>
<th>Digital Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td>Inputs</td>
<td>Digital strain gauge signal</td>
</tr>
<tr>
<td>Outputs</td>
<td>Filtered strain gauge signal</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

*Table 25: Digital Filter Level 3 Module*

<table>
<thead>
<tr>
<th>Module</th>
<th>Start and Stop Detection Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer</td>
<td>Ethan Schweinsberg</td>
</tr>
<tr>
<td>Inputs</td>
<td>Raw acceleration data</td>
</tr>
<tr>
<td>Outputs</td>
<td>Filtered acceleration data</td>
</tr>
<tr>
<td>Description</td>
<td>Start and Stop Detection Algorithm utilizes the accelerometer data to determine when a swimmer starts or stops a swim.</td>
</tr>
</tbody>
</table>

*Table 26: Start and Stop Detection Algorithm*
Figure 56: Web Application Level 3 Block Diagram
## Article V. Parts List

<table>
<thead>
<tr>
<th>Qty.</th>
<th>Refdes</th>
<th>Part Num.</th>
<th>Description</th>
<th>Cost</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pine</td>
<td>Pine A64+ 2GB</td>
<td>Pine A64 with 2GB Ram Motherboard</td>
<td>29.00</td>
<td>29.00</td>
</tr>
<tr>
<td>2</td>
<td>Waterproof Case</td>
<td>Be Dri Waterproof Storage Dry Box Container with Lanyard. Perfect for Boating, Fishing, Hunting, Camping, &amp; Hiking.</td>
<td>18.99</td>
<td>37.98</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PowerCble</td>
<td></td>
<td>Pine A64 USA Power Supply</td>
<td>5.99</td>
<td>5.99</td>
</tr>
<tr>
<td>2</td>
<td>Power Sw</td>
<td></td>
<td>Pine 64 Power/Reset Switch</td>
<td>0.5</td>
<td>1.00</td>
</tr>
<tr>
<td>1</td>
<td>XBee</td>
<td>XB9X-DMUS-001</td>
<td>XBEE-SX</td>
<td>35</td>
<td>35.00</td>
</tr>
<tr>
<td>1</td>
<td>XBee</td>
<td></td>
<td>Xbee For pine 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BT-USB</td>
<td>UD100-G03</td>
<td>USB bluetooth adapter</td>
<td>39.95</td>
<td>39.95</td>
</tr>
<tr>
<td>1</td>
<td>RFID</td>
<td>Kit-13198</td>
<td>RFID Atenna-kit</td>
<td>49.95</td>
<td>49.95</td>
</tr>
<tr>
<td>1</td>
<td>Accelerometer</td>
<td>MC3635</td>
<td>3-axis Accelerometer</td>
<td>2.93</td>
<td>2.93</td>
</tr>
<tr>
<td>1</td>
<td>BatC</td>
<td></td>
<td>Battery case</td>
<td>5.99</td>
<td>5.99</td>
</tr>
<tr>
<td>1</td>
<td>7seg</td>
<td>1270</td>
<td>Adafruit 1.2&quot; 4-Digit 7-Segment Display w/I2C Backpack - Red</td>
<td>17.5</td>
<td>17.50</td>
</tr>
<tr>
<td>1</td>
<td>Board2</td>
<td></td>
<td>Wifi/sensor Board</td>
<td>50</td>
<td>50.00</td>
</tr>
<tr>
<td>1</td>
<td>Board3</td>
<td></td>
<td>Display Board</td>
<td>50</td>
<td>50.00</td>
</tr>
<tr>
<td>1</td>
<td>Popcorn</td>
<td></td>
<td>Popcorn Parts</td>
<td>20</td>
<td>20.00</td>
</tr>
<tr>
<td>1</td>
<td>Battery</td>
<td></td>
<td>Either a rechargeable battery or disposable</td>
<td>20</td>
<td>20.00</td>
</tr>
<tr>
<td>1</td>
<td>cables</td>
<td></td>
<td>Various cables for connections</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>waterproof connectors</td>
<td></td>
<td>Various connectors</td>
<td>5</td>
<td>50.00</td>
</tr>
</tbody>
</table>

*Table 27: Parts List*
Article VI. Project Schedules
Article VII. 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
Article VIII. Conclusions and Recommendations
Article IX. References


Article X. Appendices

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