**Conestoga College ITAL**
**Electronics Systems Engineering**
**Capstone Project 1 - EEC74125**

CONESTOGA
Connect Life and Learning

# *Engineering Design and Test Plan*

**For the Barbell P.I. Capstone Project**

**By:**
**4th Year ESE students:**
**Alexander Bradley (7228885)**
**Jeffrey English (7223761)**
**Michael Wright (7218928)**

**April 12, 2019**

# Table of Contents

# Table Of Figures

# 1.0 Introduction

## 1.1 Purpose

The purpose of this document is to provide a technical reader with an organized overview of the design of the Barbell B.I., giving the reader enough information complete a prototype of the product. The final decisions for the functional blocks of the system, and the components that make up the blocks are outlined in this document. All of the design decisions that have been made are justified with reasoning from an engineering perspective. Information in this document is technical in nature, but is simple enough for an outsider to the project to understand.

## 1.2 Project Definition and Description

The focus of this project is creating a cost effective solution that improves weightlifting for users of all abilities. The device will track movement and statistics, providing useful information to the user about their lifts. Information is transmitted wirelessly and presented as a positional graph of the barbells path within a smartphone app. This allows users to make on the fly adjustments to their technique, and allows them to stay safe and efficient while lifting weights.

The Barbell P.I. tracking system uses series of sensors to track the bar path of an exercise so that the user can accurately see their motions, and make adjustments accordingly. The data collected includes a bar path graph, levelness of the bar, velocity, acceleration, and force exerted. The information is presented to the user through a smartphone application with a graphical view of the bars motion, as well as the data from the other sensors. The Barbell P.I. is a cost effective solution to improve lifts, and maintain safety while lifting for users of all ability levels.

## 1.3 System Requirements

Below is a modified list of the MuSCoW requirements. This table shows the most important parts of the project in terms of engineering design. They outline the tasks that require the majority of the engineering design in the Barbell P.I. project in order to complete the respective tasks. For other lesser important aspects of the MuSCoW list created during the project please refer to **[12].** Using the SMART template a set of criteria has been created that must be met for each requirement in the table below.

**Figure 1 - MuSCoW list**

| Requirements | Must | Should | Could | Won't |
|---|---|---|---|---|
| Self contained unit to track position based of acceleration | X | | | |
| Position results are accurate enough to effectively improve workout results | X | | | |
| Phone application capable of showcasing gathered data from sensors | X | | | |
| Bluetooth communication between device and developed phone application | X | | | |
| Low power to avoid battery issues | X | | | |

## 1.4 Smart Goals

**Figure 2 - SMART Goal 1**

| | |
|---|---|
| **Task / Goal Title** | Self contained unit to track position based off acceleration |
| **Specific** | The Barbell P.I. must be able to track position using no outside sources, this will be accomplished by using *Dead Reckoning* **[1]**.<br>This task will need to be focused on early in the project life cycle and will require knowledge working with sensors and Arduino libraries / configuration. |
| **Measureable** | Once the microcontroller is capable of producing usable position based off acceleration gained from the sensor, this task can be thought of as complete. Refining the accuracy of the data will be part of another task. |
| **Attainable** | With knowledge gained from research and previous experience using similar systems, this portion of the project will be possible after a few phases of testing. |
| **Relevant** | This task is the main focus of the project and will be used for the majority of the |

| | |
|---|---|
| | the other tasks |
| **Time-Bound** | Must be completed within the first few phases of the project life cycle in order to integrate with other tasks |

**Figure 3 - SMART Goal 2**

| | |
|---|---|
| **Task / Goal Title** | Position results are accurate enough to effectively improve workout results |
| **Specific** | The other important aspect of this project in order to have it stand out from similar devices is accuracy. This will involve testing of the sensors limits, as well as integrating an algorithm for predetermining the position based off expected results. Other ways to improving accuracy such as the Kalman Filter **[3]**. |
| **Measureable** | The best way to measure accuracy of position for this application will be to perform tests and compare them to outside observations. Since there is no other product available to compare to, reference position must be acquired using film and tracing the bar path by hand. |
| **Attainable** | This task will be difficult to complete in full due to inherent errors within the hardware, as well as the amount of time that will take to implement and test software until it matches with observable measurements. |
| **Relevant** | This task will be important to place our product above similar products by creating accuracy |
| **Time-Bound** | This task must be completed by the end of the project life cycle |

**Figure 4 - SMART Goal 3**

| | |
|---|---|
| **Task / Goal Title** | Phone application capable of showcasing gathered data from sensors |
| **Specific** | In order to show the user the data from the lift they have performed, a smartphone application capable of displaying a graph of the position of the barbell during the lift will be developed. It will be developed using Xamarin as it can be compiled to run on both IOS and Android. |
| **Measureable** | The end goal for the app will be to have an interface where the user can easily read the data gained from the sensors in a graph showing the position over time. There may also be in app suggestions for ways of improving performance. |
| **Attainable** | With previous programming knowledge and assistance from outside sources, this task will take time, but be achievable. |
| **Relevant** | Completion of this task is the only way to allow the user to see what the product is doing therefore making it an integral portion of the project. |
| **Time-Bound** | The app development will be split into multiple parts. App development |

| | |
|---|---|
| | verification will be completed early in the life cycle. Afterwards the development will be a continuous improvement over the course of the project as the sensor output become more refined. |

**Figure 5 - SMART Goal 4**

| | |
|---|---|
| **Task / Goal Title** | Bluetooth communication between device and developed phone application |
| **Specific** | Transfering the data gained from the sensor, to the app is important for reasons stated above. Bluetooth Low Energy (BLE) will be used in order to connect the device to the phone app as this is the most efficient and effective protocol for the devices' purpose . |
| **Measureable** | A finished communication between devices will be shown by having the same data shown on the phone that is on the microcontroller. |
| **Attainable** | The microcontroller selected for this project includes an off-the-shelf BLE chip **[10]** integrated in order to make this goal easier to achieve. In order to reduce communication error, as many processes as possible will be performed on the microcontroller before data is sent. This in turn reduces the amount of data that is needed to be transmitted. |
| **Relevant** | This is how the phone app will receive the information needed to output. |
| **Time-Bound** | This will be focused on later in the project and should not take longer than a few days. |

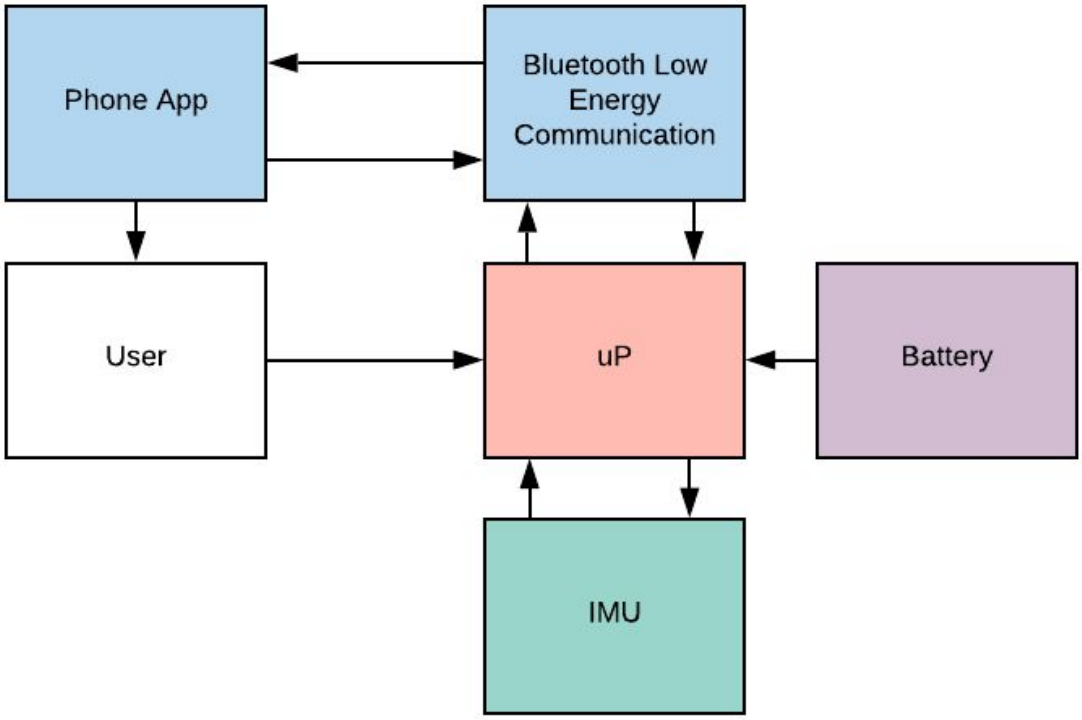
**Figure 6 - SMART Goal 5**

| | |
|---|---|
| **Task / Goal Title** | Low power to avoid battery issues |
| **Specific** | This is largely a design issue as the size of the product needs to be small enough to fit on a barbell comfortably. Power necessary for running the microcontroller, sensor as well as communicating with a phone app, although not high, could have used a larger battery if certain design choices (BLE, power efficient sensor and microcontroller) had not been made. |
| **Measureable** | The battery must fit on the device itself and be able to produce enough power to sustain a 1-2 hour workout session. |
| **Attainable** | With design choices made and the capabilities of the chosen microcontroller **[10]** this will be easily attainable. |
| **Relevant** | In order to make the product a comfortable size while still able to produce enough life time for a complete workout. |
| **Time-Bound** | Completed in design process. |

# 2.0 System Design

## 2.1 System level diagrams & drawing

**Figure 7 - Block Diagram**




## 2.2 Test Plans

During the integration phase of the project there will be 3 main systems that will need to be tested thoroughly in order to complete. These systems are

1. Initial sensor verification
2. Sensor accuracy
3. Phone app development

Due to the nature of Off-the-shelf sensors there will need to be heavy calibration in order to obtain the desired results, thus providing need for a test plan.

### 2.2.1 Initial Sensor Verification

This stage of testing is the most crucial part of the project as it is how the position is deduced using the output from accelerometers in our inertial measurement sensor. In order to accomplish this task, the acceleration measurements from the sensor must first be obtained, and then a crude double integration is performed.

$$a = \frac{dv}{dt};\ v = \frac{ds}{dt};\ \therefore a = \frac{d(ds)}{dt^2};$$

$$v = \int (a)dt;\ s = \int (v)dt;\ \therefore s = \iint (a)dt^2$$

If a raw approximation is taken as a simple sum of the integration function, it can be stated,

$$v(i) = v(i-1) + a(i)t$$
$$p(i) = p(i-1) + v(i)t$$

If it is assumed that Δt is constant, then this is a simple way of determining position based on accelerometer input. Using this model, a one dimensional approximation of the position with some error is gained.

Using Excel, acceleration data has been generated and the above formulas are used to obtain the velocity, and then position based solely on acceleration data.

**Figure 8 - Simulated Data for A&V**

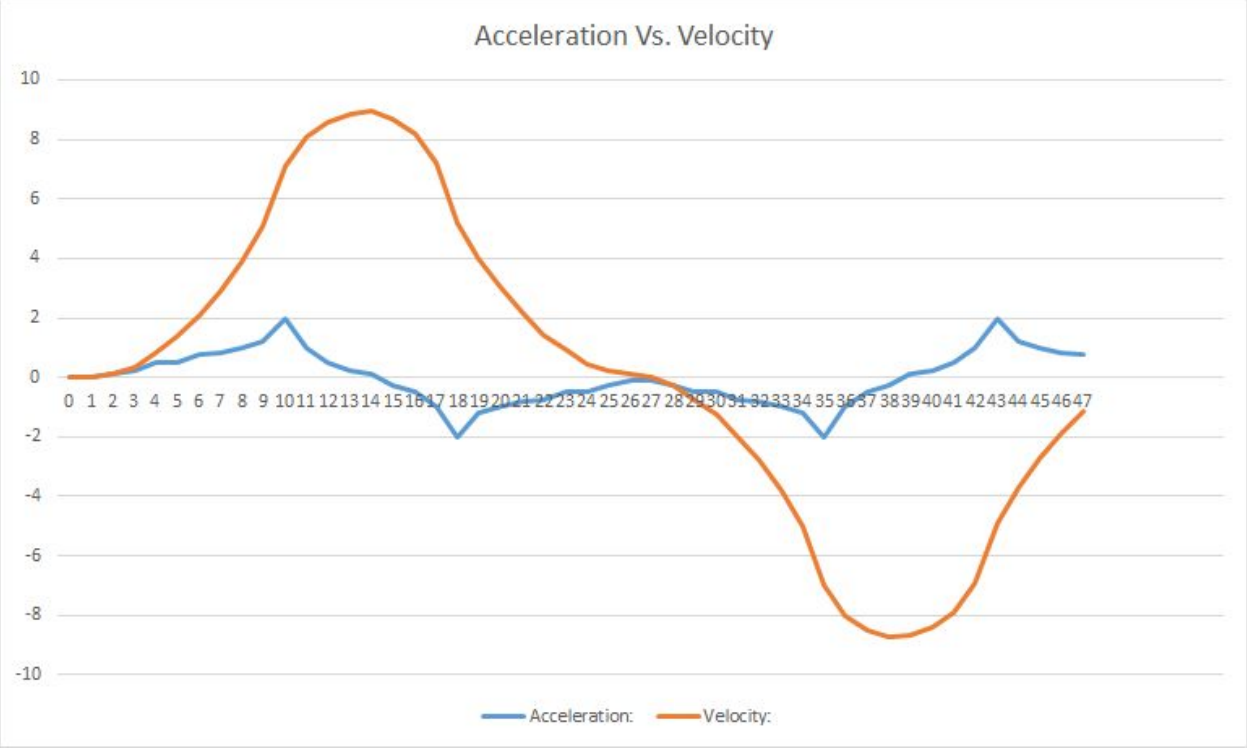
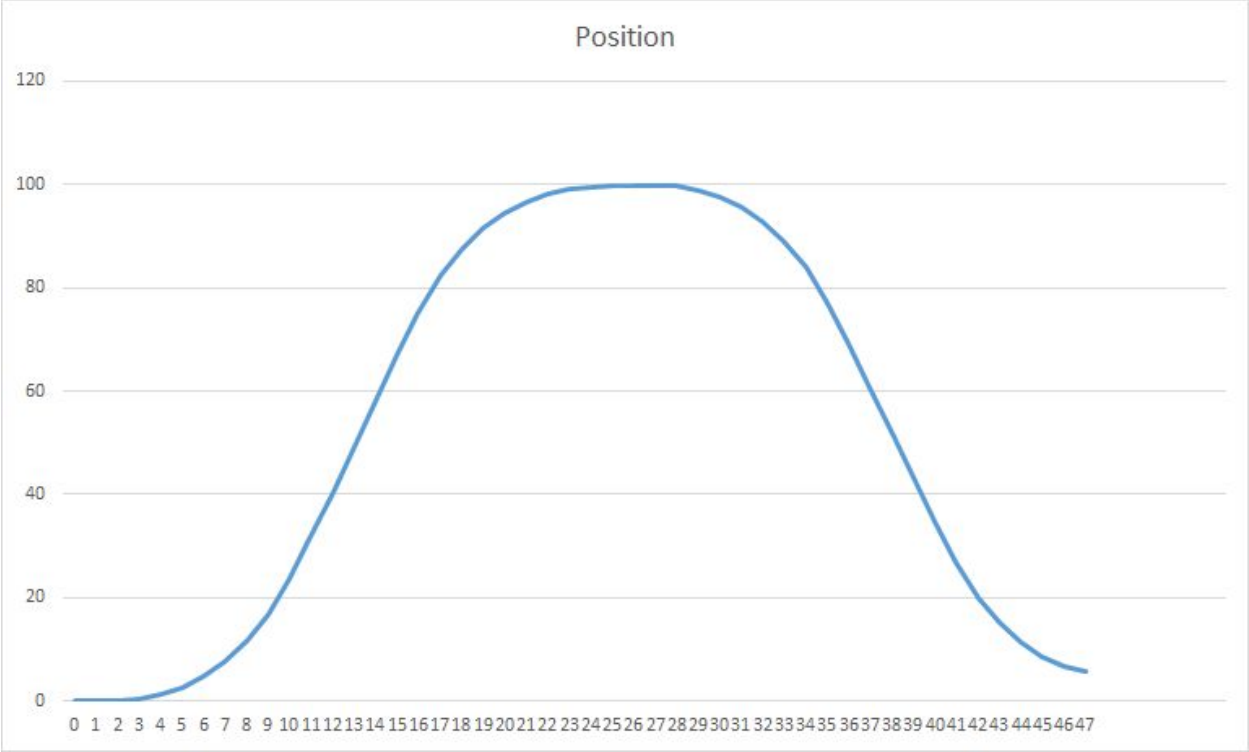


**Figure 9 - Simulated Data for Position**




The information gained this way is useful, however it is obviously very prone to error, and is only useful for gathering the distance the bar has travelled. The IMU used in the Barbell P.I. has a 3-axis accelerometer which allows the device to track position in 3D space. This will require specialized graphing software for plotting in 3D space, or a different form of showing the different axis information.

Finishing this section of the project will involve programming the math above and performing a multitude of tests in order to determine that the position obtained is actually usable data. Refining this data is part of the integration and testing phase of the project and will need much more time to be completed and perfected.

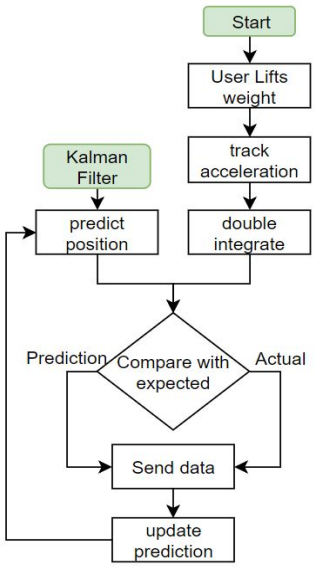
### 2.2.2 Sensor Accuracy Verification

An accurate reading of the bars position over time is the main focus of this project. The major issue with determining position from the formula above, is that the error is accumulative. In order to achieve a level of accuracy that can provide a graph with minimal percent error, there will need to be more than a simple sum approximation. After extensive research on Dead Reckoning **[1]** and Inertial Measurement Units **[4]** (INUs), it has been found that the best way to create accurate results is through the use of the Kalman Filter **[3]**.

The Kalman Filter is an algorithm that makes an educated guess about where the system is going, or what it's going to do next. The basis of the Kalman Filter is that if the current and previous state of more than one variable in the system is known, then next state can be estimated based on gaussian

probabilities. Using the probabilities of these estimates and the actual gathered data from the sensor as well as from how the system should work, an accurate measurement can be made by multiplying the probabilities.

**Figure 10 - Sensor Flowchart**




Below is a simple way of looking at how the Kalman Filter Works
Measurement error - the probability curve of where the IMU will show the position being
State Estimate error - the probability curve of where the software thinks the position is

Step 1. Find Kalman Gain (KG) $$KG = \frac{Eest}{Eest + Emeas}$$

If the measurement is accurate KG = 1

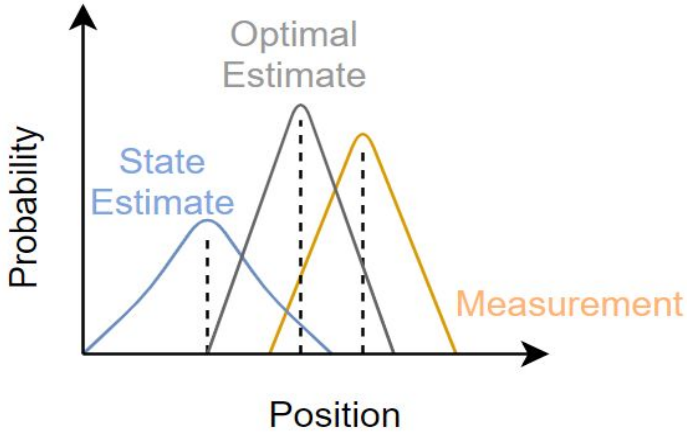
If the estimate is stable KG = 0

Step 2: Calculate the Current Estimate (EST) $$EST_t = EST_{t-1} + KG(meas - EST_{t-1})$$

Step 3: Calculate new State Estimate error $$Eest_t = (1 - KG)(Eest_{t-1})$$

As time goes on, the State Estimate error will increase in accuracy. This is important since the Measurement error of the system will become less accurate over time.

**Figure 11 - Estimate Graph**



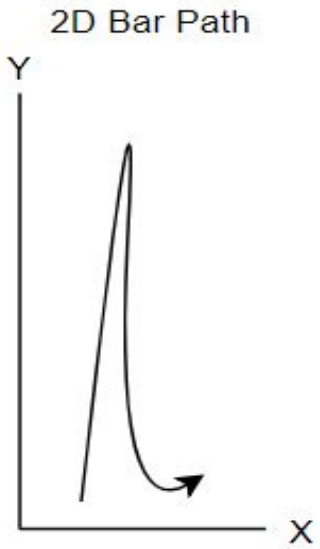

This phase of the project will involve the programming the concepts above with additional math involved that can be found in **[KF example]**. After an optimal estimate is gathered, there will be a multitude of tests to prove that the predicted position is accurate.

### 2.2.3 Phone App Verification

To make the data above useful to a user, it must be represented in a clear and meaningful way. This will require a well designed phone application which makes use of advanced graphs, charts and resources to help the user understand the data and how to make adjustments based on what they are seeing.

A 2-dimensional graph will be the primary source of information to the user. It will be a combination of 2 distance v. time graphs to give the best representation of the bar path. A sample might look like this:

**Figure 12 - Bar path example**



This view demonstrates what a possible simple lift might look like, with Y representing height and X representing distance forward or backward. To make this even more in depth, if the user clicks on a certain section of the graph it will show them their current time, velocity, and acceleration. Secondary 1-dimensional graphs will also be added for velocity v. time and acceleration v. time.

To verify the accuracy of these graphs it will be necessary to create a set of known movements and distances (say move straight up 1 meter) and compare to the graph generated. Simple plan for testing is detailed in the following table.

**Figure 13 - Plotting Test Plan**

| Test | Expected (m) | Measured (m) | Error (%) |
|---|---|---|---|
| Straight up | 1 | tbd | tbd |
| Straight forward | 1 | tbd | tbd |
| Diagonal | 1 | tbd | tbd |
| ... with more complexity | | | |

## 2.3 Safety, Societal and environmental issues associated with the project and a mitigation plan

### 2.3.1 How our solution may impact the environment and society

The Barbell P.I. device is designed to improve the safety of weightlifting. Since injuries can be common when dealing with any weights, making a lift safer will help keep users healthier and able to continue to exercise/move freely. No direct negatives to the environment will come from this product.

### 2.3.2 Describe required safety precautions if the product may cause any safety hazard

Although this device is aimed at improving aspects of a lift, not all factors can be accounted for with the data the product will generate. Some of these non-detectable issues include angles of joints, bracing of the core, etc. and can cause injury. This means that although the device will aid some users immediately, it is paramount to beginners to follow correct form. To mitigate this a document should be included with the device with either links to technical form videos or perhaps a simple message to consult a personal trainer.

## 2.4 Description of Off-the-shelf Components

### 2.4.1 Microcontroller

The microcontroller chosen to develop with is the Adafruit Feather. The Feather met all of the hardware needs, is low cost, and uses familiar Arduino libraries. The on board battery charging circuit is also an addition that is important to in order to cut down on size, as an additional battery module is not needed.

### 2.4.2 Sensor

The sensor module used for the purposes of prototyping is the Adafruit 9-DOF IMU Breakout. This board includes a 3-axis accelerometer, 3-axis magnetometer, and a 3-axis gyroscope all within a low cost, small, self-contained unit. The 3-axis accelerometer is the primary reason for choosing this unit. After some preliminary testing to determine whether this board will be accurate enough in gathering acceleration for the Barbell P.I., it is decided that this board will be used as an inexpensive placeholder for testing and validation before spending more on a more accurate sensor.

### 2.4.3 Wireless Communication

Bluetooth Low-Energy (BLE) is the chosen communication method between the Barbell P.I. and the mobile phone app. The decision was made because of BT being a well established and supported protocol, BLE is low power and will prolong device life, as well, the Adafruit Feather has an integrated BLE module making it an easy choice.

### 2.4.4 Software

**Arduino**

Arduino is the software compatible with the microcontroller and sensor that is used within this project. Its libraries are easily integrated into their hardware components making it easy to work with. There are also a multitude of sources online for basic programming assistance if the need arises. The difficulty will be creating an algorithm for increasing accuracy of the position of the device, and decreasing constant drift.

**Xamarin**

Xamarin is an IDE for producing mobile applications. It features a compiler that generates to both Android and iOS, an integrated emulator for quick debugging. It can be run as an add-on to Visual Studio which is available through the College and therefore is free to use for our group. The user interface (UI) is generated in xaml with C# code running behind it.

# 3.0 Design of Functional Units

## 3.1 Sensor Module

### 3.1.1 Description

The sensor module is the way that the data is gathered from the device that is on the barbell itself. The main type of sensor that is being used is a 3-axis accelerometer, that tracks the acceleration of the device, which is then integrated twice to output the position of the bar itself.

### 3.1.2 Design

The design choice of using an IMU for position tracking rather than an outside source to track the position of the bar is to try and create an easy to use, simple, self-contained unit. A device that is capable of capturing data gained during a workout while still being accurate and unobtrusive is the goal of this project. The 3-axis accelerometer chosen allows for information to be gathered in all three dimensions.

### 3.1.3 Tools to be Used

In order to retrieve data from the sensor itself, a microcontroller and program is needed to retrieve analog data through GPIO ports on the Adafruit Feather. The language that will be used for programming the microcontroller is C/C++ in the Arduino IDE.

### 3.1.4 Test Requirements and Methods/Plan

Testing of the sensor will be in multiple steps.

1. Verification that the sensor is working as expected
   Wiring the sensor to the microcontroller and uploading a basic program that outputs the raw data gathered from the sensor.
2. Integrating the acceleration in order to obtain the position in a crude form
   By using the method stated in section 2.2.1 and comparing those measurements off of external measurements of the barbell.

Step 2 of testing the initial sensor module will take time as multiple tests will be performed in order to show that the position gained is at least accurate for low $\Delta t$

## 3.2 Sensor Accuracy / Kalman Filter

### 3.2.1 Description

In order to accurately determine the position of the barbell during a workout, there needs to be an additional step aside from double integration of the IMUs acceleration. The Barbell P.I. will have an integrated Kalman Filter in order to achieve this accuracy. The Kalman Filter will improve the accuracy by estimating the position based on previously known position and velocity. These estimates are gained through gaussian probabilities and covariance matrices that have been researched in previous stages of the project **[11].** Since the IMU, and the method of integration are not perfect, this prediction will help negate the error created and give a consistent accurate prediction.

### 3.2.2 Design

The reasoning behind using the Kalman Filter is that during the research phase of the project, no other means of determining the position could be found that compete with the Kalman Filter. Another algorithm that could potentially allow for an accurate estimation of the errors is the Moving Average algorithm. It works essentially in the same manner but without the adaptive nature of the Kalman Filter.

### 3.2.3 Tools to be Used

Since the Kalman Filter will be processed on the microcontroller before the final result is sent to the phone application, this will need to be programmed in C/C++ in the Arduino IDE.

### 3.2.4 Test Requirements and Methods/Plan

Implementation of the Kalman Filter will be done in the same manner as stated in section 3.1.4 because of the two sections dependance on eachother.

## 3.3 Phone Application

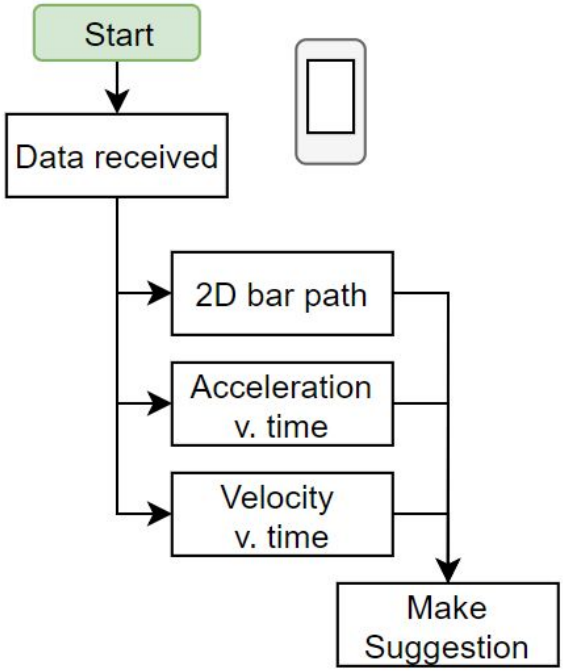
### 3.3.1 Description

The phone app will be the only way for the user to see the data coming from the Barbell P.I. device. As such it must have a user friendly UI and be compatible with new generations of cell phones. The user will use the app to view a 2-3d graph of the bar movement as well as charts on velocity, acceleration and force production.

### 3.3.2 Design

The idea for using a phone instead of a computer to display the data comes from the ease of access of using your smartphone while still in the gym, allowing the user to immediately check their results. The other main design choice was whether it would be best to develop the app for Android, iOS, or both. The best option would be compatibility with both operating systems as this increases the available market by x2 (if a 50/50 split for android/apple phones is assumed).

**Figure 14 - Application Flowchart**




### 3.3.3 Tools to be Used

The development software chosen for this project is Xamarin Forms, a subset of Visual Studio. This was decided upon as Xamarin allows multi-platform compiling and has an integrated emulator for quick debugging. The group also has access to a recent factory reset android phone for testing the software in a real world scenario with data coming from the Barbell P.I. device.

### 3.3.4 Test Requirements and Methods/Plan

The items that will need to be tested are: 1) ease of use of UI and 2) ease of understanding of the data.

1. The app itself will need to be simple enough to use by someone with perhaps limited technical knowledge. This will require a UI that is easy to follow and clear instructions to setup the Barbell P.I. device. Testing for this step will require a user outside of the group that has no knowledge of the project to navigate the app given simple directions and perform initial setup of a Barbell P.I.
2. Upon initial thought by the group, to a novice lifter the data from the device may seem confusing or even intimidating at first glance. It will be the duty of the app (and perhaps an instruction

manual included with the Barbell P.I) to explain how to read the data and graphs and apply that to improving their lifts. This step is paramount as without understanding the data the device is useless to the user.

# 4.0 Risk Analysis and Risk Management Plan

Below are the revised risks that are most important to the success of this project and its development.

**Figure 15 - Risk Management**

| Risk | Description | Severity | Management Plan |
|---|---|---|---|
| Bar position tracking | Sensors must be capable of tracking the bar path using accelerometers. | HIGH | Tackle this task from the beginning. All members will begin work on the sensor design, implementation, and testing to be certain that the output is acceptable. |
| Bar position accuracy | Accurate position | HIGH | Focus early on programming a Kalman Filter to estimate and correct position. |
| Mobile app development | Engineering team has no prior smartphone application development experience. | HIGH | Enlist outside expert to assist in preliminary application design and integration with Bluetooth connection. |

# 5.0 References and Bibliography


[1] “Dead reckoning,” *Wikipedia*, 05-Mar-2019. [Online]. Available: https://en.wikipedia.org/wiki/Dead_reckoning. [Accessed: 14-Mar-2019].

[2] “Inertial navigation system,” *Wikipedia*, 08-Mar-2019. [Online]. Available: https://en.wikipedia.org/wiki/Inertial_navigation_system. [Accessed: 14-Mar-2019].

[3] “Kalman filter,” *Wikipedia*, 13-Mar-2019. [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter#Example_application. [Accessed: 14-Mar-2019].

[4] “Inertial measurement unit,” *Wikipedia*, 27-Feb-2019. [Online]. Available: https://en.wikipedia.org/wiki/Inertial_measurement_unit. [Accessed: 14-Mar-2019].

[5] “Defense Advanced Research Projects Agency,” *Defense Advanced Research Projects Agency*. [Online]. Available: https://www.darpa.mil/program/micro-technology-for-positioning-navigation-and-timing. [Accessed: 14-Mar-2019].

[6] “Defense Advanced Research Projects Agency,” *Defense Advanced Research Projects Agency*. [Online]. Available: https://www.darpa.mil/news-events/2013-04-10. [Accessed: 14-Mar-2019].

[7] “A simple dead reckoning algorithm on Arduino,” *Creative Electronic!*, 01-Jan-1970. [Online]. Available: http://bluelemonlabs.blogspot.com/2013/07/blog-post.html. [Accessed: 14-Mar-2019].

[8] Seb Madgwick Research, “3D Tracking with IMU,” *YouTube*, 08-Mar-2011. [Online]. Available: https://www.youtube.com/watch?v=6ijArKE8vKU. [Accessed: 14-Mar-2019].

[9] K. Townsend, “Adafruit 9-DOF IMU Breakout,” *Introduction | Adafruit 9-DOF IMU Breakout | Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/adafruit-9-dof-imu-breakout. [Accessed: 14-Mar-2019].

[10] “Introducing the Adafruit nRF52840 Feather,” *Overview | Introducing the Adafruit nRF52840 Feather | Adafruit Learning System*. [Online]. Available: https://learn.adafruit.com/introducing-the-adafruit-nrf52840-feather. [Accessed: 03-Apr-2019].

[11] T. Babb, “How a Kalman Filter Works in Pictures,” *Bzarg*. [Online]. Available: https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/. [Accessed: 04-Apr-2019].

[12] A Bradley, J. English, and M. Wright. “User Requirements and Project Plan,” Bachelor of Electronic Systems Engineering - Capstone Project, Conestoga College, 2019.

[13] okreylos, *YouTube*, 05-Jun-2014. [Online]. Available: https://www.youtube.com/watch?v=_q_8d0E3tDk. [Accessed: 12-Apr-2019].

[14] Augmented Startups, *YouTube*, 19-Aug-2016. [Online]. Available: https://www.youtube.com/watch?v=bm3cwEP2nUo. [Accessed: 12-Apr-2019].

[15] *Understanding Kalman Filters, Part 3: Optimal State Estimator Video - MATLAB*. [Online]. Available:https://www.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator--1490710645421.html. [Accessed: 12-Apr-2019].

# Appendix

| System: | Arduino Nano | Raspberry Pi Zero W | Adafruit Feather (nRF52840) | Adafruit Flora | BLEPad - Wearable Dev board |
|---|---|---|---|---|---|
| Power Supply: | 7-12V via micro USB | 5V via micro USB | 3.5-16V via USB or battery | 3.5-16V via USB or battery | 3.7V via USB or battery |
| Battery Terminals | No | No | Yes with charging circuit | Yes with charging circuit | Yes |
| Current Draw Idle: | 2.5mA | 100mA | 2mA | 2mA | 2mA |
| Current Draw Maximum: | 19mA | 350mA | 10mA | 20mA | 20mA |
| Ethernet: | No | No | No | No | No |
| HDMI: | No | Yes (Mini HDMI) | No | No | No |
| RCA Video Out: | No | No | No | No | No |
| 3.5mm Audio: | No | No | No | No | No |
| Onboard Microphone: | No | No | No | No | No |
| USB: | Yes (micro USB) | Yes x2 (Micro USB) | Yes (Micro USB) | Yes (Micro USB) | Yes (Micro USB) |
| GPIO: | 22 pins | 17 pins | 21 pins | 6 pins | 18 pins |
| SD Card: | No | Yes | No | No | No |
| Wi-Fi: | No | Yes | No | No | No |
| Bluetooth | With module board | Yes | Yes | With module board | Yes |
| Clock Speed: | 16MHz | 1GHz | 64MHz | 8MHz | 8MHz |
| Architecture: | AVR Enhanced RISC | ARM v6 | ARM M4F | ATMEGA32u4 8bit | ATMEGA32u4 8bit |
| Ram: | 2KB | 512MB | 256KB | 2.5KB | 2.5KB |
| Storage: | 32KB | Provided by SD card | 1MB | 32KB | 32KB |
| Size: | 18 x 45 x 7mm | 30 x 65 x 7mm | 51 x 23 x 7.2mm | 45mm round x 7mm | 31mm round x 7mm |
| Weight: | 7g | 9g | 6g | 4.7g | 4g |
| Cost: | $22.00 | $49.95 | $24.95 | $14.95 | $18.00 |

| | Embedded Linux System Selection - Paired Comparrison | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCA Video Out: | 3.5mm Audio: | Onboard Microphone: | USB: | GPIO: | SD Card: | Wi-Fi: | Bluetooth | Clock Speed: | Architecture: | Ram: | Storage: | Size: | Weight: | Cost: | Geometric Mean: | Weight: |
| Power Supply: | 0.1 | 0.1 | 0.1 | 0.5 | 2 | 0.5 | 0.3 | 6 | 1 | 1 | 0.5 | 0.5 | 4 | 4 | 0.5 | 0.727649787 | 0.023497606 |
| Battery Terminals | 0.1 | 0.1 | 0.1 | 1 | 2 | 0.3 | 0.2 | 4 | 1 | 0.5 | 0.2 | 0.2 | 5 | 5 | 0.5 | 0.568998434 | 0.018374363 |
| Current Draw Idle: | 0.1 | 0.1 | 0.1 | 0.5 | 5 | 0.3 | 0.2 | 7 | 1 | 0.5 | 0.2 | 0.2 | 2 | 2 | 0.5 | 0.436858568 | 0.014107240 |
| Current Draw Maximum: | 0.1 | 0.1 | 0.1 | 0.5 | 5 | 0.3 | 0.2 | 7 | 1 | 0.5 | 0.2 | 0.2 | 1 | 1 | 0.5 | 0.396975216 | 0.012819309 |
| Ethernet: | 1 | 1 | 1 | 1 | 8 | 8 | 1 | 8 | 3 | 2 | 4 | 5 | 10 | 10 | 2 | 3.373420917 | 0.108936083 |
| HDMI: | 1 | 1 | 1 | 1 | 8 | 8 | 1 | 8 | 3 | 5 | 4 | 5 | 10 | 10 | 2 | 3.523871455 | 0.113794502 |
| RCA Video Out: | 1 | 1 | 1 | 1 | 8 | 8 | 1 | 8 | 3 | 5 | 4 | 5 | 10 | 10 | 2 | 3.523871455 | 0.113794502 |
| 3.5mm Audio: | 1 | 1 | 1 | 1 | 8 | 8 | 1 | 8 | 3 | 5 | 4 | 5 | 10 | 10 | 2 | 3.523871455 | 0.113794502 |
| Onboard Microphone: | 1 | 1 | 1 | 1 | 8 | 8 | 1 | 8 | 3 | 5 | 4 | 5 | 10 | 10 | 2 | 3.523871455 | 0.113794502 |
| USB: | 1 | 1 | 1 | 1 | 8 | 8 | 1 | 8 | 1 | 2 | 4 | 5 | 10 | 10 | 1 | 2.205686609 | 0.071227061 |
| GPIO: | 0.125 | 0.125 | 0.125 | 0.125 | 1 | 8 | 1 | 3 | 0.5 | 0.5 | 4 | 5 | 10 | 10 | 2 | 0.716667453 | 0.023142960 |
| SD Card: | 0.125 | 0.125 | 0.125 | 0.125 | 0.125 | 1 | 1 | 8 | 5 | 2 | 2 | 3 | 8 | 8 | 2 | 1.03758388 | 0.033506143 |
| Wi-Fi: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 | 5 | 4 | 0.1 | 0.1 | 8 | 8 | 2 | 1.71717745 | 0.055451896 |
| Bluetooth | 0.125 | 0.125 | 0.125 | 0.125 | 0.333333333 | 0.125 | 0.125 | 1 | 0.5 | 0.5 | 0.4 | 0.3 | 1 | 1 | 0.5 | 0.251294503 | 0.008114919 |
| Clock Speed: | 0.333333333 | 0.333333333 | 0.333333333 | 1 | 2 | 0.2 | 0.2 | 2 | 1 | 0.5 | 1 | 1 | 7 | 7 | 1 | 0.821582951 | 0.026530940 |
| Architecture: | 0.2 | 0.2 | 0.2 | 0.5 | 2 | 0.5 | 0.25 | 2 | 2 | 1 | 3 | 1 | 7 | 7 | 0.3 | 0.910970298 | 0.029417478 |
| Ram: | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.5 | 10 | 2.5 | 1 | 0.333333333 | 1 | 1 | 8 | 8 | 0.5 | 1.034400006 | 0.033403280 |
| Storage: | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.333333333 | 10 | 3.333333333 | 1 | 1 | 1 | 1 | 8 | 8 | 0.5 | 1.006165454 | 0.032491564 |
| Size: | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.125 | 0.125 | 1 | 0.142857142 | 0.142857142 | 0.125 | 0.125 | 1 | 1 | 0.1 | 0.195099600 | 0.006300247 |
| Weight: | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.125 | 0.125 | 1 | 0.142857142 | 0.142857142 | 0.125 | 0.125 | 1 | 1 | 0.1 | 0.195099600 | 0.006300247 |
| Cost: | 0.5 | 0.5 | 0.5 | 1 | 0.5 | 0.5 | 0.5 | 2 | 1 | 3.333333333 | 2 | 2 | 10 | 10 | 1 | 1.275857942 | 0.041206004 |
| | | | | | | | | | | | | | | | | Sum: | |
| | | | | | | | | | | | | | | | | 30.96697449 | 1 |

| | | | Embedded Linux System Selection - Selection Chart | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| System: | Geometric Mean: | Weight: | Arduino Nano | | Raspberry Pi Zero W | | Adafruit Feather (nRF52840) | | Adafruit Flora | | BLEPad - Wearable Dev board | |
| | | | Rating: | Weight: | Rating: | Weight: | Rating: | Weight: | Rating: | Weight: | Rating: | Weight: |
| Power Supply: | 0.7276497876 | 0.02349760671 | 5 | 0.1174880335 | 5 | 0.1174880335 | 5 | 0.1174880335 | 5 | 0.1174880335 | 5 | 0.1174880335 |
| Battery Terminals | 0.5689984342 | 0.01837436312 | 0 | 0 | 0 | 0 | 7 | 0.1286205419 | 7 | 0.1286205419 | 5 | 0.09187181561 |
| Current Draw Idle: | 0.4368585688 | 0.01410724089 | 7 | 0.09875068622 | 3 | 0.04232172267 | 8 | 0.1128579271 | 8 | 0.1128579271 | 8 | 0.1128579271 |
| Current Draw Maximum: | 0.396975216 | 0.01281930904 | 7 | 0.08973516327 | 2 | 0.02563861808 | 9 | 0.1153737813 | 7 | 0.08973516327 | 7 | 0.08973516327 |
| Ethernet: | 3.373420917 | 0.1089360834 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HDMI: | 3.523871455 | 0.1137945025 | 0 | 0 | 5 | 0.5689725123 | 0 | 0 | 0 | 0 | 0 | 0 |
| RCA Video Out: | 3.523871455 | 0.1137945025 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3.5mm Audio: | 3.523871455 | 0.1137945025 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Onboard Microphone: | 3.523871455 | 0.1137945025 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| USB: | 2.205686609 | 0.07122706191 | 5 | 0.3561353096 | 6 | 0.4273623715 | 5 | 0.3561353096 | 5 | 0.3561353096 | 5 | 0.3561353096 |
| GPIO: | 0.7166674533 | 0.02314296004 | 7 | 0.1620007203 | 5 | 0.1157148002 | 7 | 0.1620007203 | 3 | 0.06942888012 | 5 | 0.1157148002 |
| SD Card: | 1.03758388 | 0.03350614312 | 0 | 0 | 1 | 0.03350614312 | 0 | 0 | 0 | 0 | 0 | 0 |
| Wi-Fi: | 1.71717745 | 0.05545189603 | 0 | 0 | 1 | 0.05545189603 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bluetooth | 0.2512945037 | 0.008114919453 | 3 | 0.02434475836 | 8 | 0.06491935562 | 8 | 0.06491935562 | 3 | 0.02434475836 | 8 | 0.06491935562 |
| Clock Speed: | 0.8215829515 | 0.0265309403 | 4 | 0.1061237612 | 8 | 0.2122475224 | 6 | 0.1591856418 | 2 | 0.0530618806 | 2 | 0.0530618806 |
| Architecture: | 0.9109702988 | 0.02941747826 | 5 | 0.1470873913 | 5 | 0.1470873913 | 5 | 0.1470873913 | 3 | 0.08825243477 | 3 | 0.08825243477 |
| Ram: | 1.034400006 | 0.03340332801 | 5 | 0.16701664 | 8 | 0.2672266241 | 7 | 0.2338232961 | 5 | 0.16701664 | 5 | 0.16701664 |
| Storage: | 1.006165454 | 0.03249156467 | 5 | 0.1624578234 | 2 | 0.06498312934 | 8 | 0.2599325174 | 8 | 0.2599325174 | 8 | 0.2599325174 |
| Size: | 0.1950996005 | 0.006300247398 | 8 | 0.05040197918 | 3 | 0.01890074219 | 6 | 0.03780148439 | 7 | 0.04410173178 | 8 | 0.05040197918 |
| Weight: | 0.1950996005 | 0.006300247398 | 5 | 0.03150123699 | 2 | 0.0126004948 | 8 | 0.05040197918 | 9 | 0.05670222658 | 9 | 0.05670222658 |
| Cost: | 1.275857942 | 0.04120060041 | 6 | 0.2472036025 | 2 | 0.08240120082 | 6 | 0.2472036025 | 9 | 0.3708054037 | 8 | 0.3296048033 |
| | Sum: | | | | | | | | | | | |
| | 30.96697449 | 1 | | 1.760247106 | | 2.256822558 | | 2.192831582 | | 1.938483449 | | 1.953694887 |

| Option | Marmalade | Ionic | Xamarin | Android Studio |
|---|---|---|---|---|
| Platform | Cross | Cross | Cross | Android |
| Language | C,C++ | Java | C# | Java, C, C++ |
| Emulator | Yes | Yes | Yes | Yes |
| Cost | Commercial | Free | Free | Free |
| Recommended | Yes | No | Yes | No |

| | | Options | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Marmalade | | Ionic | | Xamarin | | Android Studio | |
| Criteria | Wgt | Rating | Score | Rating | Score | Rating | Score | Rating | Score |
| Platform | 0.2 | 4 | 0.8 | 4 | 0.8 | 4 | 0.8 | 2 | 0.4 |
| Language | 0.4 | 4 | 1.6 | 1 | 0.4 | 3 | 1.2 | 4 | 1.6 |
| Emulator | 0.15 | 4 | 0.6 | 4 | 0.6 | 4 | 0.6 | 4 | 0.6 |
| Cost | 0.15 | 1 | 0.15 | 4 | 0.6 | 4 | 0.6 | 4 | 0.6 |
| Recommended | 0.1 | 2 | 0.2 | 2 | 0.2 | 4 | 0.4 | 1 | 0.1 |
| Total | 1 | 15 | 3.35 | 15 | 2.6 | 19 | 3.6 | 15 | 3.3 |
| Rank | | | 2 | | 4 | | 1 | | 3 |
| Choice | Xamarin | | No | | No | | Yes | | No |