Smart Gloves

Luis Gamarra, Andrew Smith, Scott Suarez, Jason Surh

EEL 4914, Group 22

December 2017

Contents

List of Figures

List of Tables

1 Executive Summary

Weightlifting is not often seen as an activity requiring a great deal of thought. Those who lack experience with the sport might consider it to be rather brutish, with emphasis placed on the amount of weight lifted as opposed to the careful practice of each exercise and the required planning that is necessary for an effective workout session. While one could go about a weightlifting session with little regard as to how they target and train each muscle group – after all, there technically aren't any official rules dictating how the average fitness enthusiast should approach a workout – this approach will at the very least decrease workout efficiency and effectiveness, and at the worst can lead to temporary or even permanent injury. Neither of these outcomes are desirable, but are far too often obtained by individuals who lack the knowledge or experience necessary for safe and effective workouts. The aim of our Senior Design project is to create a device that will help these individuals maximize the efficiency of their workouts and avoid serious injury.

Smart Gloves are designed to provide a structured boost to a user's workouts while avoiding any sort of interference with range of motion. Each glove will carry a small PCB on its back that will track the relative position of a user's hands during individually selected exercises through an integrated inertial measurement unit (IMU). Both IMUs will record data through the onboard accelerometer, gyroscope, and magnetometer, sending the data through the PCB to a wireless transmitter that will deliver it to a user device such as a smartphone or laptop. Incoming data from the IMU will first be double integrated to calculate the relative position of each glove, after which any recorded drift will be digitally filtered out. The path each glove creates while performing repetitions of an exercise will be compared real-time to a mathematically modeled "ideal" path, allowing the device to notify users if their form is incorrect and in need of adjustment. Other data, such as the number of reps performed and repetition speed, will be collected and recorded in order to provide feedback that will further maximize a user's workout effectiveness. Users will not need to interact with the standalone console during the bulk of their workout; instead, they will use a personal computational device such as a laptop or smartphone to choose which exercises are tracked and set parameters that will dictate the flow of their workout.

This report will document the entire design process of Smart Gloves, from initial project conceptualization to our final planned implementation. The following section will provide a more in-depth characterization of the project and will outline our underlying motivations, goals, and specifications to be met. The next section will provide an analysis of all relevant research performed and will detail the process through which the hardware and software implemented in our final design were selected. All related engineering design standards will be discussed in the subsequent section, which will also identify and discuss any design constraints that

had the potential to impact any decisions made concerning system design. Following this, the hardware and software design of the project will be reviewed and discussed in detail, which will include all relevant schematics and diagrams pertaining to our design. Once the overall hardware and software design has been laid out, the construction of our project prototype will be described and all testing procedures will be explained in detail. This document will wrap up with the inclusion of all related administrative content, such as discussion of project milestones and financial information.

2 Project Description

This section of the report will describe in detail the underlying motivation behind the design of Smart Gloves, with further emphasis on identifying the driving goals and resulting specifications that are to be met by our final design. Additionally, a house of quality diagram will illustrate the relationship between various user requirements and engineering requirements.

2.1 Project Motivation

As described in brief detail in the Executive Summary, weightlifting is a much more mentally involved activity than the average person would think. While it may seem as simple as repetitively picking heavy objects up and putting them down, there are various practices and concepts that can affect the potency of a lifter's workout. Arguably the most important practice in weightlifting is maintaining proper form for every individual exercise performed. The concept of "form" is essentially the proper way to perform an exercise that maximizes its effectiveness in training the targeted muscle group and minimizes the chance of injuries, like sprains and tears. For example, if a lifter allows their arms to drift too far over their head while flat bench pressing they will shift the weight being supported from off their chest to directly over their shoulders. In this example, the chest is the primary muscle group being worked and is expected to bear the brunt of the weight; by shifting this weight to a secondary muscle group such as their shoulders, they not only defeat the purpose of the exercise but run the risk of causing severe damage to smaller muscle groups.

The concept of using proper form is simple enough to grasp when being discussed, but it is much more difficult to put into practice. A lifter's perception of how they are lifting weights is largely dependent on their position relative to the weights and the ground: while it may look and feel to them that they're pushing a pair of dumbbells up perpendicularly to the ground, they might actually be pushing the dumbbells upward at an angle that activates muscle groups necessary for proper practice of that particular exercise. The stress of supporting heavy weights can also be very mentally taxing and often distracts lifters, preventing them from keeping the mental clarity necessary to maintain proper form. Newcomers to weightlifting often struggle the most with maintaining form, as they lack the experience necessary to keep it in check. Additionally, those who work out without a partner monitoring their form may never even find out what they're doing wrong. As stated multiple times before, the use of improper form can pose a serious threat to even the most seasoned weightlifters, as decreased workout effectiveness and severe injury can completely halt a lifter's progress. Building strength and reshaping your body through weightlifting doesn't occur overnight and requires long-term dedication and

commitment to see any desired change. As such, even the slightest hindrance in a lifter's progress may cause discouragement and in some cases, turn them away from the sport entirely. By tracking a user's form as they lift, Smart Gloves will allow inexperienced lifters to develop proper form and prevent any discouragement from occurring.

Inefficient workouts also serve as a major hindrance to a weightlifter's long-term progress. While it is true that an inefficient workout is better than no workout at all, efficiency is key to making noticeable progress in a timelier fashion. A lifter's workout efficiency is affected by many factors, primary of which is the actual structure of their workout. Unstructured workouts with no thought given to which muscle groups are targeted or how they're trained can prevent lifters from realizing the maximum potential of their efforts in the gym. To prevent this, the user application for Smart Gloves will offer basic workout routines users can follow that target different muscle groups with various intensities. Users will also be able to construct their own workout routines out of the programmed exercises to meet their own goals. Wasted time can also serve as a hindrance to a lifter's workout efficiency; to minimize this, the Smart Gloves application will offer a programmable rest timer to count down between each set a user performs. This will keep lifters motivated and on track, reducing time spent aimlessly between sets.

While the dominant application of Smart Gloves will center on weightlifting, it may potentially serve as a useful tool for physical therapists and their patients. When an individual suffers a severe bodily injury, physical therapy is often employed after initial recovery to rebuild the strength of the affected area through specific exercises. For example, someone who has just recovered from a torn bicep will likely see a physical therapist that will walk them through basic arm exercises to re-strengthen the affected muscles. In the presence of a physical therapist, patients will have no doubt that they're performing any assigned exercise correctly – the therapist would tell them otherwise – but at home they might be a little more uncertain. In this application, proper form is crucial: any mistakes made while performing these exercises can undo progress made during recovery. Under the direction of a physical therapist, patients could use Smart Gloves at home to provide further guidance in recovery efforts by monitoring how exercises are performed. This would potentially allow patients to speed up their overall recovery process and save money that would be spent on more frequent visits to their therapist.

2.2 Project Goals and Objectives

The primary goal of Smart Gloves is to maximize the efficiency of a user's workout while minimizing the risk of severe injury. While this principle will dictate the majority of our design process, several other project goals are shown below:

- Smart Gloves and all included hardware shall not interfere with a user's range of motion during workouts. This would defeat the purpose of our project and would likely pose a negative impact overall. For this reason, we decided to incorporate wireless technology to avoid tangling the user in a cluster of wires. Additionally, all glove-mounted electronics including both PCBs will not exceed the width or length of a user's hand. Large PCBs placed on the backs of a lifter's hands would interfere with a user's range of motion and affect their grip on any weights held.
- Smart Gloves will be intuitive and simple to use. Again, if this were not the case the purpose of our project would be defeated as nobody wants to complicate their workouts if they can help it. Our project will be designed to help users simplify their workouts by taking uncertainties about form or structure out of the equation.
- The total cost of hardware and software will remain low, ensuring affordability for individuals of many differing economic backgrounds. Since weightlifting is an activity that can be enjoyed by poor and rich alike, we want to make sure everyone has the opportunity to practice it safely and effectively.
- Smart Gloves shall have low power consumption, preventing users from having to change batteries with every passing workout. A short battery life would cause more of a headache than anything, and many users would give up on Smart Gloves if they were required to change or charge batteries with every workout.
- All Smart Gloves hardware will be portable and convenient enough for users to take with them to a gym. If the hardware is too bulky to travel with and/or susceptible to damage, it would give users a reason to leave Smart Gloves at home for the sake of convenience.

2.3 Requirements Specifications

Hardware Specifications: Table 2.1 displays several of the defining hardware specifications of our Smart Gloves project that will shape and guide our design decisions.

Table 2.1: Hardware Specifications

Requirement	Specification	Value
1	Glove-mounted electronics shall not exceed width of glove.	Less than 90 x 90mm
2	Glove-mounted electronics shall feature reasonable shock resistance.	Resistance up to 10kg impact
3	Sensor latency regarding position tracking shall be minimized.	Kept below 0.5s latency
4	Position tracking shall be accurate and reliable.	Accuracy within 10-15cm
5	Battery life shall last at least as long as one workout period.	2 hours minimum
6	IMU accelerometer shall accurately track acceleration within reasonable range.	Accuracy up to 3G
7	Gloves shall maintain reasonable wireless communication range with user device.	2m minimum

Software Specifications: Table 2.2 displays several of the defining software specifications of our Smart Gloves project that will shape and guide our design decisions.

Table 2.2: Software Specifications

Requirement	Specification	Value
1	Software shall accurately calculate relative position of gloves from IMU data.	Accuracy within 10-15cm
2	Mathematically modeled paths shall be automatically tailored to a user's body dimensions through brief calibration process or user input (to account for variables such as shoulder width and arm length). The calibration must be done in a short period of time.	Duration less than 5 minutes.
3	Software shall track position and monitor form real-time, notifying user of improper form as it occurs through audio cues in application.	Kept below 0.5s latency
4	Software shall track speed of repetitions and notify user if too fast/slow.	Specification is exercise dependent
5	Application shall feature an intuitive design and be simple to use.	Every feature can be accessed within 5 steps

2.4 House of Quality

A house of quality diagram is a tool that allows the connections and dependencies between important user and engineering requirements to be visualized and presented in a coherent manner. The house of quality diagram seen in 2.1 displays these interconnections that will characterize the development of Smart Gloves.

User requirements are the aspects of design that matter the most to consumers and affect their likelihood to purchase a product. For Smart Gloves, one of the leading user requirements is overall ease of use; weightlifting can be a complicated enough activity and consumers will not be purchasing Smart Gloves to further complicate things. The companion application paired with Smart Gloves and the variety of exercises it's capable of tracking will also be important to all consumers, as an in-depth application featuring a wide variety of trackable exercises will go much further in helping lifters achieve their personal goals. The accuracy of Smart Gloves is also a crucial requirement to users looking to meticulously track their form along with a wide array of other measurables specific measurables. The accuracy of Smart Gloves will be influenced by setup time, which consumers will want to keep as short as possible yet as long as necessary for initial calibration. The form factor, or the size and shape, of all Smart Gloves technology will be another important factor for users to take into consideration. Perhaps the most important user requirement, however, is that the cost of Smart Gloves remain as low as possible without making sacrifices that would affect their performance.

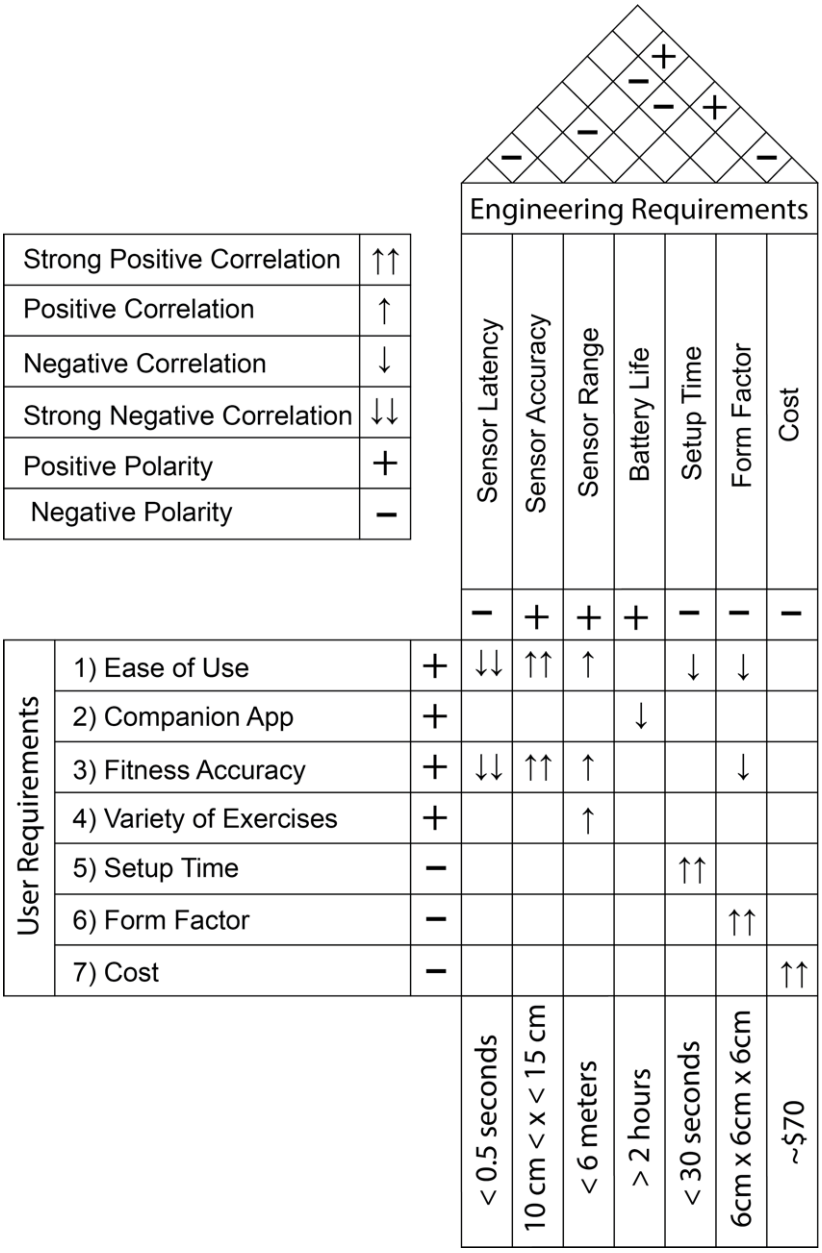


Figure 2.1: House of Quality

Engineering requirements describe the needs of the system from the viewpoint of its engineers and developers. These requirements describe how Smart Gloves will work from more of a technical perspective as opposed to an ease-of-use mindset. First and foremost of these requirements is minimized sensor latency, as noticeable lag in any measurements of user movement would hurt the accuracy of the gloves. All sensors must demonstrate high accuracy as well so that the relative positions derived from acceleration and orientation data are as close to the actual value as possible. The maximum range between glove-mounted electronics and the standalone console must also be reasonably high to prevent loss of wireless signal

if a user is required to move slightly further away from the console for a particular exercise.

Battery life is also an important requirement to be taken into consideration on the development side. A larger battery would improve battery life but take up more space on the PCB, leaving us to find an appropriate balance between battery life and size. Setup time is not only important to users, but also to us as developers of Smart Gloves. The longer the setup time, the greater the opportunity to fine-tune any calibrations made to sensors on the gloves; however, a shorter setup time produces less strain on the battery. Form factor will determine how compact all electronic devices on the glove-mounted PCBs will need to be and will influence the cost of PCB fabrication and the creation of a protective structure for the sensitive electronics. Minimizing cost is also an important requirement on the design front.

3 Research Related To Project

3.1 Similar Existing Projects

The following section of this report describes existing projects with similar scope to our Smart Glove design. These will serve as a basis of inspiration for potential design ideas that can be incorporated into both hardware and software.

3.1.1 Motion Tracking Glove

This motion-tracking glove, shown in Figure 3.1, was developed by Thalikshan Kanesalingam at McMaster University for the purpose of allowing users with mobility issues to control an assistive robotic device. Designed with an integrated IMU, this glove not only tracks relative hand position and orientation but also finger flexion; however, only derivation of relative position and orientation are detailed in this report. Our Smart Glove design will have no need for finger flexion tracking, and as such the lack of this detail in the reviewed report will have no impact on our design. In this design, an ArduIMU Sensor Board was chosen with an integrated accelerometer and gyroscope. This board was interfaced with an Arduino Duemilanove microcontroller that was uploaded with embedded software to process the raw data from the ArduIMU board and convert it into relative positioning data. This data was sent through serial communication to a computer that used it to run a virtual simulation of the user's hand movements.

The methods used by the developer to obtain reliable relative positioning data are highly similar to the methods Smart Gloves will use on received IMU data. Both the angular acceleration provided by the gyroscope and linear acceleration by the accelerometer were double integrated to calculate the relative position and orientation of the user's hand. Because the ArduIMU board in this application was attached to a user-worn glove, a strapdown system was utilized in which sensor data was computed in relation to a base frame of reference [1].

Though very similar in scope to the basis of our Smart Gloves project, a few key differences exist between the hardware implementation of this project and our ideal design. First and foremost, this motion-tracking glove uses wired connections between the ArduIMU board, Arduino microcontroller, and computer that transfer information and supply power to the IMU. Wired connections such as these would restrict user movement and prevent a full range of motion in the case of Smart Gloves. Additionally, the wired setup of this motion-tracking glove eliminates the

need for an on-board power source to power the IMU as the Arduino board provides power directly through a wired connection.

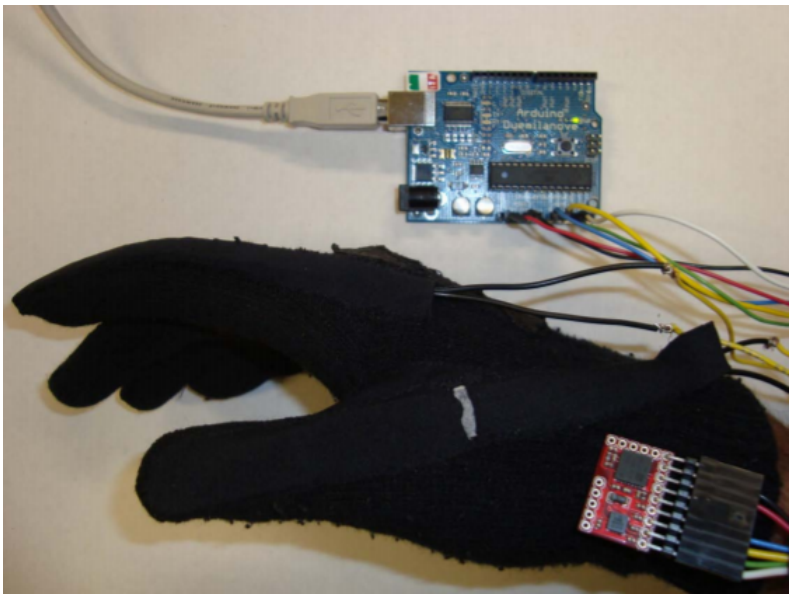


Figure 3.1: Motion-Tracking Glove for Human-Machine Interaction

3.1.2 Integrated Smart Glove

In this application, an integrated smart glove was developed by professionals from various technical institutions in Ireland with the purpose of monitoring a range of hand joint movements in real time in order to facilitate Human Computer Interaction [2]. This device, shown in Figure 3.2, was designed with flexible areas to stretch over the user's fingers and connect to the main PCB. 16 9-axes IMU's were included with the device, with an IMU fitted to each finger joint area on the flexible portion to measure flexion characteristics of each individual finger. A 32-bit AVR microcontroller with single precision floating point capabilities was implemented on the main PCB of this device to allow for complex embedded algorithms that would calculate motion parameters of the user's hand and fingers from each IMU.

Wireless communication technology was also implemented on this device through means of a WLAN device that provides direct wireless data transfer with any device sporting a UART or SPI interface. Additional hardware included on this device consisted of rechargeable batteries, optional storage via microSD, and a USB bootloader/communication interface. The IMUs implemented in this device were the TDK InvenSense MPU-9150, which features integrated gyroscopes, accelerometers, and magnetometers that provide measurements across 9 axes.

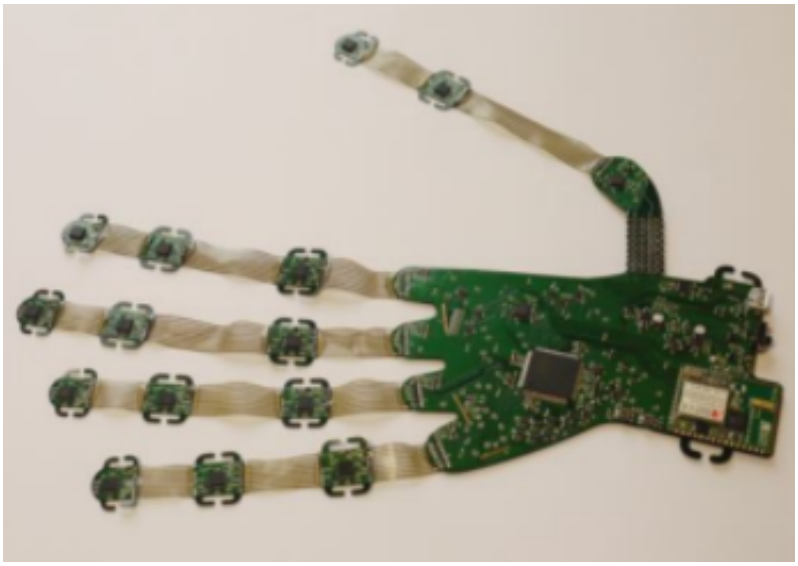


Figure 3.2: Integrated Smart Glove

3.2 Market Analysis

3.2.1 Notch

Notch is a product targeted at boxers and runners as well as physical therapists The purpose to the device is to allow an individual to determine problematic areas with the motion and correct the form. Notch is a compact mobile sensor array that uses a configuration of six sensors with configurable low-power bluetooth radios. The sensors are a series of accelerometers, gyroscopes, and compasses that work to pass data to a mobile device and a companion application. Using the array, Notch is able to record the user's motion and then relay that information to a mobile device in real time. Complex calculations are then applied to the sensor data so the Notch software can render a three dimensional view of the movement on a mobile device. [3]

The Notch array of sensors has a very small profile that allows it to attach easily to garments and various equipment. There is flexibility in the sensor array to add and remove modules from around the body when you want to track various ranges of motion. There is an additional feature that supports haptic feedback to let the user know if they have made a good or bad move.

All in all, Notch seems to be a solid contender in the motion tracking marketplace. With the 6-sensor kit pricing in at $387, Notch is meant for the higher end of the

wearable sensor tracking market. That being said, the customization and functionality of this product seems to warrant the price.

3.2.2 OptiTrack

OptiTrack is a company with a variety of products and services. Theres products and services range from Virtual Reality and Robotics, to Movement Sciences and Animation. For the purpose of this document, we will focus on the Movement Sciences area of their business. [4]

OptiTrack offers native supports for a variety of sensors such as force plates, electromyographies (EMGs), and analog signal displays. The outputs of these sensors are readily analyzed in Motive, or other third party biomechanics packages such as Visual3D, The MotionMonitor, or MATLAB.

Motive is Optitrack's production motion capture software. Motive offers high precision, biomechanically relevant motion capture using a variety of motion marker sets. Motive supports a variety of biomechanics marker sets, with a focus on anatomically valid configurations. The simplicity of these marker sets offer a unique blend of both performance and usability. Motive's real time labeling engine makes use of a clear understanding of a subject's unique skeletal structure. Due to this understanding, Motive can deliver highly accurate labels, even during occlusion between limbs and other subjects in view. When additional labeling is needed, there are industry standard naming conventions that can added for a familiar workflow.

3.2.3 Moov

Moov Now is a sports wearable device designed to be a sports coach for running, cycling, swimming, and cycling; with the added benefit of being a daily activity tracker. The design of Moov Now is very simple. The device is a small disc that is held in place by a latticed rubber strap. The device and band is light enough to where it is practically unnoticeable to the wearer. [5]

Even while having a small form factor, this device has months of battery life, step tracking, sleep monitoring, fitness updates, cross training, and run coaching. The core activities for Moov Now are Run and Walk, Cycling, Cardio Boxing, Total Body Workout, and Swimming. Moov Now has a companion app on a mobile device. The mobile app offers an extensive amount of configurations and exercises to track.

The sensor array for Moov Now is pretty extensive. Moov Now utilizes nine-axis Omni Motion sensor technology. Doing so allows Moov Now to track movement in

three dimensions. Moov Now offers pin-point accuracy for sleep detection without the need to manually tell the device when the user sleeps and wakes up.

3.3 Relevant Technologies

The following section will contain a brief analytical description of all relevant technologies that will be present in our Smart Gloves design. Following a brief description of all relevant technologies, an in-depth analysis of products with potential for use in our design will be presented.

3.3.1 Inertial Measurement Units - IMUs

Considering the intended application of Smart Gloves, inertial measurement units are arguably the hardware component with the highest attributed level of significance. IMUs are electronic modules that contain on-board accelerometers, gyroscopes, and sometimes magnetometers that provide data about a body's movement that can be translated into relative or absolute positioning information. While accelerometers and gyroscopes of decades past were originally bulky and unreliable, thanks to modern advances in microelectromechanical systems (MEMS) these components have undergone major decreases in size along with increases in accuracy and compatibility with other devices.

Modern IMUs are categorized as either analog or digital: digital IMUs output data using serial protocols such as I2C, SPI or USART while analog IMUs provide a varying output voltage that must be converted to a digital value using analog to digital converter (ADC) modules [6]. For the purposes of our design, only digital IMUs will be taken into consideration to eliminate the need for unnecessary additions such as ADC modules. IMUs have a wide range of applications and are commonly used in a wide variety of consumer, military, and industrial technologies.

One of the two major components of IMUs are accelerometers, which measure the linear acceleration of an attached rigid body. Modern MEMS accelerometers consist of near-microscopic proof masses attached to springs that allow the mass a small range of motion. When experiencing acceleration in a given direction, the mass is displaced from its steady-state position and this displacement is converted into an electrical signal through electrostatic, or capacitive, sensing [7]. This electrical signal is then interpreted to provide information about the acceleration of the device in the direction measured. Modern accelerometers have the capability to measure acceleration individually across the x, y, and z axis, which would classify them as having 3 degrees of freedom (DOF). Accelerometers with 3 DOF can essentially measure the acceleration of an object in 3-dimensional space, by which

relative positioning information can be obtained through double integration and filtering of any observed errors.

Gyroscopes, the second of the two major components in IMUs, use a similar process to measure the angular velocity and orientation of an object. Similar to accelerometers, gyroscopes incorporate multiple proof masses suspended by springs where displacement of the masses is measured by electrostatic transducers and converted into electrical signals [8]. The simplest of gyroscopes measure angular velocity over one axis, but the majority of modern gyroscopes can measure angular velocity over 3 axes, corresponding to roll, pitch, and yaw. In this regard gyroscopes can offer 3 DOF; when coupled with a 3 DOF accelerometer the two components create an IMU with 6 DOF. Many IMUs also include magnetometers, which are designed to measure the magnetic fields surrounding it. If this magnetometer is designed to measure magnetic fields across 3 DOF, the IMU is characterized as measuring across 9 DOF.

Though IMUs don't quite provide exact positioning data, the rotational and acceleration data provided can be analyzed to provide a reasonably accurate description of the IMU's relative position. This is done through single integration of the angular velocity and double integration of linear acceleration. These methods will be further analyzed in following sections.

3.3.2 Wireless Technologies

There a few wireless technologies that are relevant to our interests for the communications with the gloves. At the moment it appear that the following options are available to us.

Wi-fi - Needless to say wi-fi is definitely a consideration when identifying routes we can take. Although, in our case, it seems like it would be a less than ideal due to the need to utilize TCP and tunnel through a router (unless we broadcast from a local node). Ideally we want minimal latency so while this is an option maybe it would be better to say last resort.

Zigbee - These are essentially low power radios that are utilized in combination with high level communication profiles to create local networks between devices. The limits of range are not extensive (10-100m) however advantage lies with low power consumption, ability to create mesh networks, less interference (if you aren't already using Zigbee products) and security. Mainly used for home automation and security. [9]

RF transceivers - By far the cheapest option would be to utilize RF transceivers, however this would have the caveat of not providing reliable communication between modules unless we programmed our own protocol and create filters in ac-

cordance. This type of communication, while longe range, would not be good at reliably and securely transmitting our data. Given that we need to integrate data on the accelerometer data we receive it's imperative the data isn't off when we receive it to correctly calibrate.[10]

Bluetooth/Bluetooth LE - This technology is seen almost everywhere and you more than likely have a device capable. Bluetooth was created in order to provide communication with very short range devices which is why you open think of wireless mouses of headsets when considering the technology [Wireless 1]. Bluetooth LE differs from bluetooth as you would think, Low Energy, so less power consumption, it's designed to go into a low power sleep, pooling until a connection is reestablished. In many ways Zigbee and Bluetooth LE are extremely similar however Zigbee manages to have the advantage with respect to range while Bluetooth LE has the advantage on interfacing since so many devices already use bluetooth.[9]

Near Field Communication (NFC) - This tech makes life easier by making it similar to do transactions and exchange digital content with a touch. Mainly utilized for access control, health care, coupon and payments the range for this is fairly limited (about 4cm). The benefit of this is that the tech is inherently secure due to the short range and is versatile. Unfortunately as the gloves will be moving a fair range this technology isn't applicable for our case. [11]

Bluetooth LE communication is the ideal candidate when considering these options due a few advantages. First, so many devices are already compatible with bluetooth, the attach rate for bluetooth technology is almost 100 percent for phones, tablets, and laptops [11]. This would give us flexibility in the finalized design for the communication system and really opens up options for us software wise. Second by ensuring low power consumption we can decide to utilize a battery while prolonging the life of the power source. This would enable us to minimize the amount of time the user spends charging the units and extend the amount of time they can utilize the hardware.

3.3.3 Bluetooth Low Energy, Expanded

Bluetooth technology has a layered architecture, consisting of 4 layers total. The lowest of these layers is the Lowest Layers or the controller. This layer basically is responsible for discovering other devices, making connections, exchanging data, security, low power modes, and is normally baked into the bluetooth chip (aka bluetooth controler). Above that is the Upper layers. These basically interface with the lower layers to provide more complex functionality like splitting up big pieces of data to send our or reassembling data, streaming music, etc. Above the Upper Layers is the Profiles layer. This layer basically handles the protocol information and how two technologies are going to talk to each other. It expects things to

adhere to a certain usage models as defined by the Bluetooth Special Interest Group (SIG). Finally the top layer is the Bluetooth Application itself. This performs the tasks of the MMI (Man to Machine Interface) so the communication can be utilized. At this layer you can set files up for transfer, search for devices, browse files of a remote device, connect a headset, ect.[11]

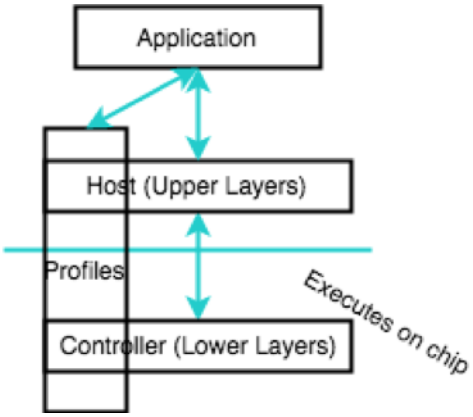


Figure 3.3: Bluetooth Architecture Layers visualized

Connecting to a bluetooth devices require two devices. First device B needs to be discoverable, in this case device B would be our gloves, then device A would make an inquiry. An inquiry just pulls all discoverable bluetooth devices in the area. At this point B needs to be connectable and device A can then create a connection to device A. Once connected, B is considered the Slave and A the master. A connection basically mean packets can go from A to be B and vice versa. Once a connection is no longer needed devices can disconnect and either device can initiate this.

There are two classes of devices in a bluetooth connection. A central device (master) and a peripheral device (slave). A device classed as master can connect to multiple devices simultaneously and the slave align themselves to the master clock intervals. A slave can only connect to a single device at time and is usually a peripheral (mouse, keyboard, ect). Each side communicates with the other during a connection interval at a minimum of 7.5 millisecond intervals called a communication event[11]. Deciding to use this technology would therefore guarantee a maximum of 133 updates per second in the ideal case. However, every platform isn't supported equally. For example, iOS 9.2 and iPhone 6 the smallest connection interval supported is 15 milliseconds while android supports the standard minimum of 7.5 milliseconds. Due to this latency it is ideal that the majority of the actual leg work when it comes to positional tracking is done on the microcontroller if this technology is chosen.

Considering we are utilizing motion tracking, which may require more frequent updates, significant thought has been given as to whether or not this is the tech we

wanted to use for communication. While researching different LMUs we have found the refresh rate is usually around 100Hz which matches up nicely with our 133Hz refresh rate over bluetooth communication. As long as we can still get data we should face no issues with Bluetooth LE.

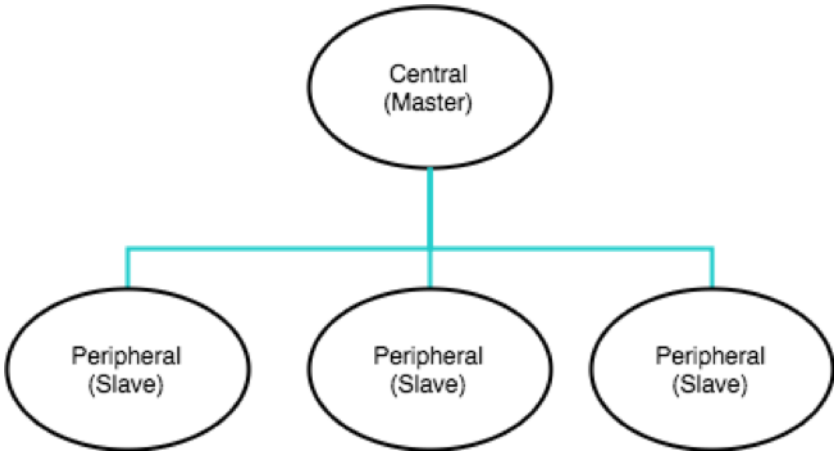


Figure 3.4: An example of a typical connection setup

3.3.4 Microcontrollers

An Introduction to Microcontrollers

In embedded design a microcontroller is, to put it simply, a small computer on a single integrated circuit that manages the majority of the of the calculation and data processing. The unit is usually self contained meaning the all the needed hardware is stored on chip such as processor, memory, RAM, and oscillator. You can find microcontrollers in the majority your embedded devices such as telephones, cars, appliances, mice, headsets, etc.

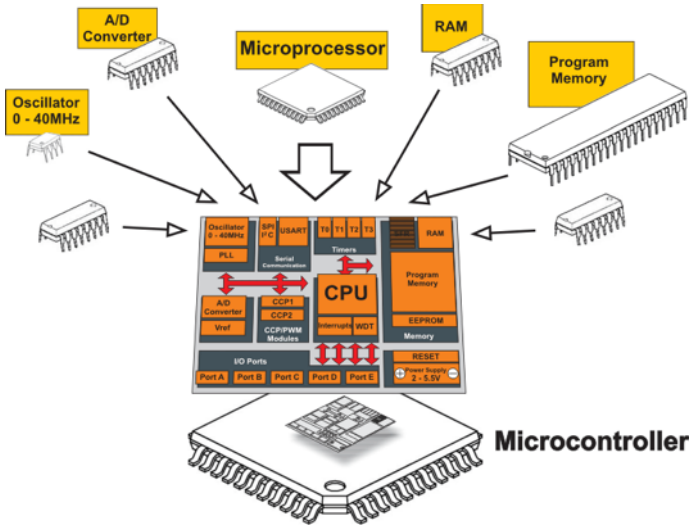


Figure 3.5: Basic Microcontroller layout [Appendix B.2]

Common features that are import to consider when selecting a microcontroller are the suite of instructions that are available to you or the instruction set architecture. The instruction set architecture is usually designed by an external company such as ARM and licensed to chip manufacturers who utilize and distribute microcontrollers to customers. In addition to processing microcontrollers also must support interrupt which enable the microcontroller to respond in real time to events either outside or within. Interrupts enable the processor to switch to something immediately of priority such as temperature changes or waking up from sleep mode at the bequest of a timer. Program size is also important to consider for microcontroller selection since the program that you are pushing must be able to fit onto your chosen design.[12]

3.3.5 Power Sub-System

This section will review the basic theory of the components expected to be implemented into the Smart Gloves power sub-system. A comprehensive review of the electronic parts to be implemented into the Smart Gloves power system can be found in section 3.4.

Linear Voltage Regulators

Linear regulators are considered to be a key ingredient to almost every power supply used across a wide array of electronic devices. These devices take a constant DC supply voltage, usually between set parameters, and convert it into a specific DC output voltage that remains constant despite changes in load current or input

voltage. For example, if an electronic circuit was powered by 15V supply voltage but required 3.3V to supply power to integrated devices, a 3.3V linear regulator would be implemented to provide a constant 3.3V across all devices.

Providing a stable power input is critical for a quality and consistent user experience. Without such a device an unstable fluctuation could damage critical components, interrupt communication between devices, or preemptively shut down and restart devices mid way. Luckily implementation of linear regulators is relatively foolproof; linear regulators are often some of the cheapest components of any given electrical circuit due to their simplistic design. In Figure 3.6 below there is a simple implementation of a voltage regulator showcasing how cheaply they can be produced.

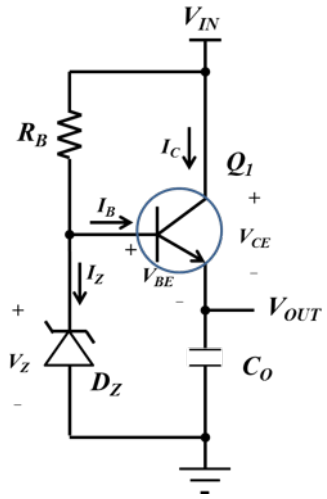


Figure 3.6: Shows the design of a simple NPN linear voltage regulator

Battery Source

Battery selection for our application will a rather simple experience after including a linear regulator in series since this will properly manage power output. Although it is important we select a battery that will be able to last an adequate amount of time in order to serve our purposes. A preliminary choice will be selected in component selection but due to the modular nature of the batteries we can easily switch between separate models in subsequent iterations. The voltage regulator enables us with with this flexibility. There is one caveat with this selection though. The battery needs to be able to supply the minimum current at or above the rated voltage. If it isn't able to meet this requirement then our selection would not be able to fully power the circuit.

Battery Life Monitering and Charging

In wearable technologies it is crucial that end user be able to recharge the battery of their devices and monitor the amount of remaining battery life in a reliable way. In order to measure remaining battery life we can utilize a microcontroller's Analog to Digital interface to get an accurate reading of the current voltage levels from the battery. This would also offer protection user side by providing the microcontroller with an idea of when it needs to prepare to shut off due to low battery levels enabling our software environment to have some flexibility in that manor. In order to properly measure remaining battery life it is notable that we might have to step down the battery voltage in order for our microcontroller to get an accurate reading. Doing this is rather simple by utilizing a resistor in series with the ADC interface.

Charging the device would require us to utilize a charge controller such as the MCP73831/2. These devices are small portable and compatible with input voltages from usb. This would simplify our design by only requiring us to connect up the battery and usb power in order to charge the device taking care of the cycle time and component isolation for us.[13]

3.3.6 USB to UART Hub for Microcontroller Programming

In order to program the device optimally we would ideally like a USB to UART interface in order to easily push code to the processor. In order to actually utilize this interface we will need to first create a bootloader on the microcontroller that would accept code to be pushed from this interface. We would also utilize this interface in order to charge the battery when plugged into USB. For information on charging technologies see section 3.3.4.3

3.3.7 Bootloading a Microcontroller

To be able to actually write to the microcontroller through the USB to UART interface we first need to bootload the microcontroller and allow it to interface as such. Basically when we receive the chip at first it's a blank slate. What a bootloader does is put an application at that base address the microcontroller starts up at that manages things like connecting to UART and accepting new software. It will also start the existing application if there is no new software to be pushed to the memory.

To bootload a microcontroller you have to utilize a tool such as a J-Link Segger to connect the microcontroller directly to your computer and manually push the bootloader onto it. As for the actual bootloader there are common ones abound for

any architecture we will consider so it shouldn't be too troublesome to find one that would fit our purposes.

3.3.8 Communication Protocols

In this section we will discus the common communication protocols that can be utilized to communicate between our microcontroller and peripheral devices such as the IMU. Examples of these protocols include the UART or Universal Asynchronous Receiver-Transmitter interface that we will but hooking up via USB to push our software to, SPI, and I2C. We will explore the basics of each of these communication protocols and mention their relevance to our project.

3.3.9 Universal Asynchronous Receiver-Transmitter (UART)

A Universal Asynchronous Receiver-Transmitter (UART) is a specific type of computer hardware for asynchronous serial communication in which both the data format and the transmission speeds are configurable. The beauty of the asynchronous nature of this hardware is that two or more events do not have to happen at the same time. The computer control timing protocol for a specific operation begins upon receipt of an indication or signal, and the a process is executable. These actions are a type of call and response and are not real time.

Figure 3.7: UART Communication Diagram

For UART hardware, the electric signaling levels and methods are handled by a driver circuit external to the UART. A UART is usually an individual part used for serial communications over a computer or peripheral device serial port. This type of communication will prove useful for the Smart Gloves project.

Smart Gloves will use low cost embedded systems will use the UART in conjunction with the CPU to sample the state of our sensor data from the input ports and the directly manipulate the output port for data transmission to our mobile companion application.

3.3.10 Serial Peripheral Interface Bis (SPI)

SPI is a communication profiles utilizes in embedded programming for short distance communication. Developed by Motorola in the 1980s the communication scheme didn't actually come into mainstream popularity until after Motorola's patent expired in 2006. SPI utilizes two lines for its communication, one for the data and one for the clock signal to keep the data transfer synchronized. Reasons for popularity are because it is extremely easy to implement. All that is needed hardware wise is simple shift register for the data and a clock output connection and you are good to go.

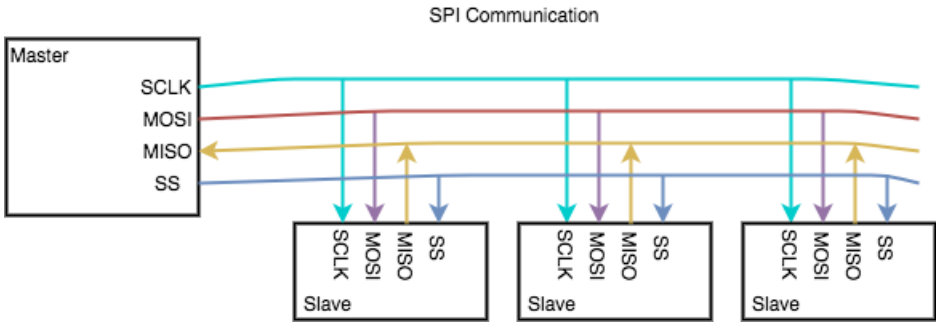


Figure 3.8: SPI Communication Diagram

While two lines provided a one way communication between devices, which will be more than enough for our purposes since we will only need to interface with the IMU by utilizing one of these technologies, we can also connect a third line for MISO (master in, slave out) communication for sending data back two ways. The slave would send the data over this new line in accordance with the clock signal. You can also chain devices together on the same lines by utilizing the selection line or Slave Select (SS). SPI should not really be utilized for long distance communication because it is susceptible to loss of data from corruption. If you plan to communicate at a longer distance consider utilizing protected data communication schemas such as TCP.[14]

So to cap this section off the advantages of SPI are that it's relatively fast, has a simple hardware implementation by requiring only a shift register/clock interface, and can support multiple slaves connected at once. Also due to the simplistic nature of the protocol if we needed to programmatically create a custom communication schema using SPI we could technically do so which provides us flexibility in that regard.

As for the disadvantages of the SPI, it requires many connections with the microcontroller, taking up possible GP/IO pins that could be utilized elsewhere. The communication must be well defined in advance and both components should understand the timing intervals. This second part should not be an issue for us since SPI

has been around for awhile and is fairly standardized as for how communication is performed. Another negative point is that the master controls the communication between devices and separate lines are required for any slave communication. [14]

3.3.11 I2C Protocol

I2C (Inter-Integrated Circuit), is a multi-master, multi-slave, packet switched, single-ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors). It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in a short distance, intra-board communication. A particular strength of I2C is the capability of a microcontroller to control a network of device chips with just two general purpose I/O pins and software. Many other bus technologies used in similar applications, such as Serial Peripheral Interface Bus, require more pins and signals to connect devices.[15]

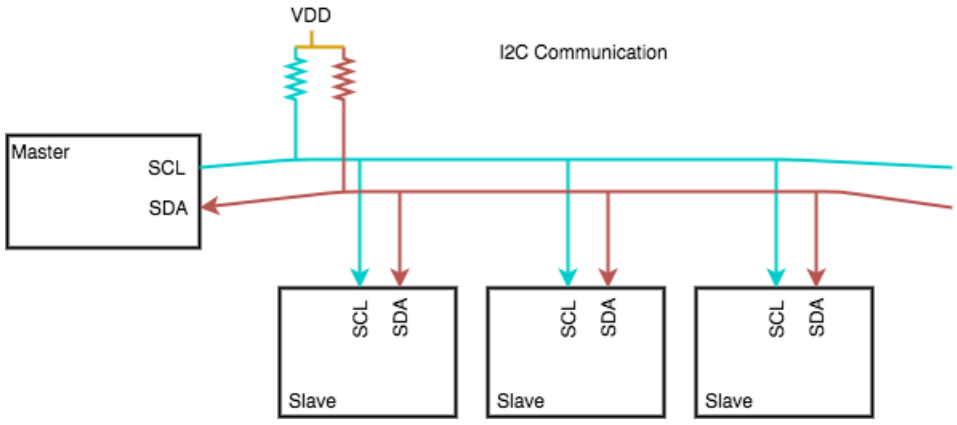


Figure 3.9: SPI Communication Diagram

The two main communication lines for I2C are the SDA and SCL lines. SCL is the clock signal controlled by the master device while the SDA line is the data line. For communication over I2C the data line, SDA, is pulled low by either a master or a slave. Once this happens this device has taken 'control' of data communications. Basically this is the device saying hey I got something to transmit let me take the reins. Once a device takes control, other devices won't be able to transmit over the SDA line. Data rates for I2C devices communicate at around 100 - 400Hz.

There is some overhead with I2C since every 8bits sent an extra bit, or the meta data bit, must also be transmitted. Basically the way communication works is message are broken up into two types. An address frame, which basically just lets the master select the slave when interfacing with multiple units, and one or more data frames. Data frames are simple communication intervals of arbitrary length

controlled by either the master or the slave. Stop conditions for data transfer are by a low to high ($0 \rightarrow 1$) change on SDA while the clock signal SCL is high.[15]

3.3.12 Piezo Buzzer

A Piezo buzzer is a useful circuit element that will allow us to make beeps, tones and alerts from the gloves. Lightweight and simple, this cheap component basically stretches or compresses in accordance with an alternating electric field, in effect producing a sound. To interface a microcontroller with a Piezo Buzzer all we need to do is connect it to a PWM (Pulse Width Modulation) pin. Once connected this would allow us to set the frequency of the Piezo Buzzer and the corresponding sounds produced by managing the oscillation of the PWM unit. This will enable us to make complex sound patterns such as a bootup tune or simple ones to alert the user to some sort of movement correction. Since these elements are so cheap to include and easy to interface with it would be an excellent option for audio output.

3.4 Potential Component Review

The following section analyzes the defining characteristics of various electrical components with potential to be implemented in our Smart Gloves design. This section will not directly compare each component, as in-depth comparisons will be made in section 3.5 that will determine the selection of each part.

3.4.1 IMU Consideration

TDK invenSense ICM-20948 - The ICM-20948 MotionTracking Device is a low-power IMU designed by TDK InvenSense with a wide variety of applications from consumer electronics such as smartphones and tablets to wearable fitness sensors. This multi-chip module features two dies in a 3x3x1mm quad flat no-leads (QFN) package: one die consists of a 3-axis accelerometer, 3-axis gyroscope, and a Digital Motion Processor (DMP) while the other die consists of a 3-axis magnetometer. All three sensors combined provide motion-tracking technology across 9 axes of orientation. Various electrical and qualitative/quantitative characteristics of this IMU can be seen in Table 3.1.

The main draw to this product is its extremely low operating power and input voltage requirements: TDK InvenSense claims that it is “the world’s lowest power 9-axis MotionTracking device” currently on the market [16]. Power consumption can be minimized according to the operation mode specified by the user and is fur-

ther managed through the on-board DMP. The DMP performs other tasks critical to maximizing runtime efficiency of the embedded system, such as offloading computation of motion processing algorithms from the host processor, saving MIPS, and controlling background runtime calibration of the accelerometer, gyroscope, and magnetometer which maintains ideal operating conditions for each and minimizes error from received data. All DMP features are completely independent of the host OS and embedded architecture. The ICM-20948 features both I2C and SPI serial interfaces for communication and data interface, both popular digital communication protocols. Other features of this device include user-programmable interrupts, embedded temperature sensors, and 20kg shock reliability.

Table 3.1: Basic Properties of ICM-20948

Specification	**Value**	**Significance**
Supply Voltage Range (VDD)	1.71-3.6V	Considerably low supply voltage
Digital I/O Supply Voltage (VDDIO)	1.71-1.95V	Considerably low I/O supply voltage
Power Consumption	5.6mW (9-axis enabled, DMP disabled, VDD 1.8)	Very low power consumption
Digital Communication and I/O Formats	I2C (100/400KHz), SPI (7MHz)	Both very common communication protocols

Both the accelerometer and gyroscope on the ICM-20948 have user-configurable ranges for sensory input, which are further configured by selectable output data rates (ODR) and low pass filters. The accelerometer includes full-scale ranges of +/-2g, +/-4g, +/-8g, and +/-16g that measure across 3 axes, of which the output data is converted to a digital format through integrated 16-bit ADCs. The 3-axis gyroscope is programmable to ranges of +/-250 dps, +/-500 dps, +/-1000 dps, and +/-2000 dps, of which the output data is also converted to a digital format through 16-bit ADCs. Both sensors feature integrated self-test protocols to ensure accuracy of output data and are regularly calibrated by the DMP. The magnetometer, like other sensors, measures across 3 axes with a full scale measurement range of +/-4900 uT and output data resolution of 16-bits. An internal magnetic source allows it to perform self-test functions to ensure sensor accuracy.

TDK InvenSense MPU-9250 - The MPU-9250 MotionTracking device is a MCM fabricated by TDK InvenSense that fits two dies in on 3x3x1mm QFN package: one die houses a 3-axis gyroscope and 3-axis accelerometer while the other houses a 3-axis magnetometer. A dedicated I2C sensor bus provides full 9-axis MotionFusion output through an on-board DMP that fuses sensor data while regulating power consumption and maintaining background sensor calibration. Like the ICM-20948, potential applications of this device include consumer electronics such as motion-controlled gaming devices and wearable fitness sensors. Some basic electrical and qualitative/quantitative characteristics may be seen in Table 3.2.

Table 3.2: Basic Properties of MPU-9250

Specification	**Value**	**Significance**
Supply Voltage Range (VDD)	2.4-3.6V	Standard Supply Voltage Range
Digital I/O Supply Voltage (VDDIO)	1.71-VDD	I/O requires lower voltage than supply
Power Consumption	8.75mW (9-axis enabled, VDD = 2.5V)	Very low power consumption
Digital Communication and I/O Formats	I2C (100/400KHz), SPI (100KH - 1MHz)	Standard communication protocols, easy to use

This device is very similar to the ICM-20948, sporting many of the same features such as an on-board DMP that fuses sensor data to provide fully digitized output of motion tracking information. Like with the ICM, the MPU-9250 DMP controls background calibration of each sensor and offloads motion data processing from the host processor to save valuable MIPS and maximize runtime efficiency for the user. Compared to the ICM-20948, however, the MPU-9250 operates at a higher supply voltage and experiences slightly higher power consumption. Both I2C and SPI serial interfaces are supported for external communication and data transfer with other embedded hardware. Other features include a precision clock with 1 percent drift from -40 to 85 Celsius and 10kg shock tolerance [17].

The accelerometer on the MPU-9250 measures across the X, Y, and Z axis to provide 3 DOF with a user-programmable full scale range of +/-2g, +/-4g, +/-8g, and +/-16g. The analog output of the accelerometer is converted to a 16-bit digital output through three integrated 16-bit ADCs, which is complemented by additional

user-programmable low-pass filters and other digital filters. Gyroscopic data is also measured across 3 axes with user-programmable full scale ranges of +/-250 dps, +/-500 dps, +/-1000 dps, and +/-2000 dps. Digitally-programmable low-pass filters and three 16-bit ADCs complement the gyroscope to provide 16-bit digital output. The 3-axis magnetometer provides a full scale measurement range of +/-4000uT with 14-bit output data resolution. All three sensors utilize self-testing functions to ensure accuracy of reported data and to maximize sensory efficiency.

Bosch Sensortec BMX055 - The Bosch Sensortec BMX055 absolute orientation sensor packs a 16-bit gyroscope, 12-bit accelerometer, and geomagnetic sensor into one 3x4.5x0.95mm LGA 20-pin package. Each sensor measures across 3 perpendicular axes, allowing for full sensory capabilities across 9 axes. This device is capable of detecting and quantizing general motion across a wide range of applications in consumer electronics, such as handheld devices and gaming controllers. Digital bi-directional SPI and I2C interfaces allow for optimal system integration and communication with other devices. Table 3.3 showcases a few of the defining electrical characteristics of this device.

The BMX055 allows for a high level of programmability on the user's end, with each MEMS sensor offering individual operation according to user selection. Power consumption is also user-controllable to a high degree, with multiple low-power modes available and configurable around a programmable interrupt engine that provides contextual status of each sensor. This level of flexible programmability allows the BMX055 to be tailored for use on a wide variety of user applications. Integrated First In First Out (FIFO) memories allow the device to buffer inertial sensor data and maximize the efficiency and accuracy of each sensor.

Table 3.3: Basic Properties of BMX055

Specification	**Value**	**Significance**
Supply Voltage Range (VDD)	2.4-3.6V	Standard supply voltage range
Digital I/O Supply Voltage (VDDIO)	1.2-3.6V	I/O requires lower voltage than supply
Power Consumption	5.63mA, 14.1mW (full operation, VDD=2.5V)	Low power, but higher than TDK InvenSense
Digital Communication and I/O Formats	SPI (4-wire, 3-wire), I2C, 4 interrupt pins	Standard communication protocols, easy to use

The on-board processor on the BMX055 is fully compatible with Bosch FusionLib software, which intelligently fuses 9 axes of motion sensing data to provide an absolute orientation vector in the form of quaternion or Euler angles. This robust software would handle the bulk of motion processing, fusing raw sensor data together into a form that would be much easier to work with than the raw data itself. FusionLib software also features the built-in capability to minimize drift and sensor offset by managing background calibration of each sensor and applying drift cancellation algorithms such as kalman filters to the digital data output.

The BMX055 MEMS accelerometer offers programmable full-scale ranges at +/-2g, +/-4g, +/-8g, and +/-16g with low-pass filter bandwidths of 1kHz - ¡8Hz [18]. The analog output of this sensor measures acceleration across 3 axes, providing a cohesive 12-bit digital output through 3 included ADC modules. Gyroscopic measurement is also fully-programmable by users, with full-scale ranges of +/-125 dps, +/-250 dps, +/-500 dps, +/-1000 dps, and +/-2000 dps. Drawing less than 5mA of current when fully-powered, this sensor measures roll, pitch, and yaw across 3 axes and produces a 16-bit digital output through 3 included ADC modules. The magnetometer features typical measurement ranges of +/-1300uT across the X and Y axes and +/-2500uT across the Z-axis, with magnetic field resolution of 0.3uT. An integrated ADC module converts the analog output of this sensor into a 16-bit digital output. Each MEMS sensor supports programmable motion-triggered interrupt-signal generation for use in a variety of applications, allowing for fine-tuned detection of any changes in position or orientation [18].

Bosch Sensortec BNO055 - The Bosch Sensortec BNO055 intelligent 9-axis absolute orientation sensor is a complete System in Package (SiP) device that fuses data from a triaxial accelerometer, gyroscope, and geomagnetic sensor using a 32-bit cortex M0+ microcontroller. Designed in a 28 pin 3.8x5.2x1.1mm LGA package, the BNO055 features intelligent power management with multiple power modes available to support an ample range of user operation configurations. Digital bi-directional I2C and UART interfaces allow for optimum system integration with optional HID-I2C capabilities for direct interface with devices running Windows 8 OS. The integration of MEMS sensors and sensor fusion technology on a single device allows for a wide array of applications directly pertinent to our Smart Gloves design. Some basic electrical and power consumption characteristics are shown in Table 3.4.

The BNO055, like the BMX055, offers a high level of user programmability, with multiple selectable low-power modes available to fit a wide variety of applications. Each MEMS sensor is also individually programmable to a certain degree, with separate operation modes available to allow the user to control aspects such as power consumption, accuracy, and power up speed. Raw data output from each sensor is also available depending on user configuration, allowing the user to obtain simple rotation vectors and linear acceleration values in relation to gravity.

Table 3.4: Basic Properties of BNO055

Specification	Value	Significance
Supply Voltage Range (VDD)	2.4-3.6V	Standard supply voltage range
Digital I/O Supply Voltage (VDDIO)	1.7-3.6V	I/O supply voltage lower than VDD
Power Consumption	12.3mA, 36.9mW (VDD=3V, VDDIO=2.5V)	Higher power consumption than others
Digital Communication and I/O Formats	I2C, UART, optional HID-I2C	Standard communication protocols, easy to use

The on-board 32-bit microcontroller intelligently fuses all sensor data together using Bosch FusionLib software, which provides absolute orientation vectors in the form of quaternion or Euler angles across 9 axes of motion detection. User-programmable interrupts assist users in determining what types of motion should wake the BNO055 or its MEMS sensors up from standby or suspended power mode, allowing the BNO055 to spend more time in a low-power mode until full operational use is required. The included FusionLib software also intelligently manages drift and sensor offset by maintaining background calibration procedures for each sensor and by digitally filtering output data to cancel any detected drift.

The BNO055 accelerometer has a high level of programmable functionality, with acceleration ranges of +/-2g, +/-4g, +/-8g, +/-16g, selectable low-pass filters with bandwidths between 1kHz - ¡8Hz, and multiple operation modes to manage power consumption and 14-bit data output [19]. The gyroscope features a similar level of programmability, with selectable angular velocity measurement ranges from +/-125 dps to +/-2000 dps, programmable low-pass filter bandwidths from 523Hz-12Hz, and multiple operation modes from normal operation, advanced power save, fast power up, and deep suspend. The magnetometer features magnetic field detection at typical ranges of +/-1300uT over the X and Y axes and +/-2500uT over the Z-axis, with a selection of individual operating modes that balance accuracy and power efficiency. Both the accelerometer and gyroscope feature on-chip interrupt controllers with programmable motion-triggered interrupt generation. All three MEMS sensors measure across 3 axes for a combined total of 9 DOF, with all sensor data fused into cohesive angular velocity output.

3.4.2 Microcontroller Considerations

The microcontroller we choose will be the main processing unit on gloves attached to a customized pcb design. Selection of the microcontroller takes several considerations. Chief among them is the ease of integration/development, power consumption of the microcontroller, and flexibility offered to us. This decision is definitely not one to be taken lightly as the wrong choice could seriously hinder our project if we've built it upon a poor base. After researching, the following have been selected as considerations, and note that the information is sourced from the corresponding datasheets.

ATmega32U4

This ATmega32U4 is a high performance low power microcontroller based on 'Advanced RISC Architecture'. This chip hosts the following features: first it provides a wide range of instructions hosting 135 with 32 general purpose registers at its core. 32K bytes of programmable flash memory with 'read-while-write' capabilities, 1K byte of EEPROM, 2.5K bytes SRAM, and 26 general purpose I/O lines. Four flexible timers with compare mode and pulse width modulation capabilities. A compliant JTAG test interface utilized for on-chip debugging and programming. Six power saving modes including an idle mode which stops the CPU while preserving SRAM, Timer/Counters, SPI port, and interrupt system to continue operation.[20]

We can expect the current draw to be from 2-14mA depending upon the voltage and frequency chosen. This would mean a power consumption of anywhere from 5.4-77 mW during active use. This could prove problematic for production application however should be fine for our current utilization given a quality option for battery supplementation. Unfortunately there is no on chip support for an Floating point manipulation meaning all calculation would have to be emulating adding a significant amount of overhead to our calculations on the microchip.[20]

Table 3.5: Basic properties of ATmega32U4

Specification	Value	Significance
Clock speed	16MHz	Moderate - High Speed
Memory	32 kB flash 2.5 kB RAM	Moderate onboard memory
Floating Point Support	None	Difficult manipulation of IMU data
Supply Voltage Range (VDD)	2.7 - 5.5V	Variable voltage supply
Digital I/O Supply Voltage (VDDIO)	High: 2.3(min) Low: 0.5(max)	Compatible i/o range with components
Power Consumption	5.4-77 mW	large and variable power consumption
Digital Communication and I/O Formats	1xTWI(100-400KHz) 2xSPI(125KH-8MHz), 1xUART(up to 12MHz)	Wide variety of communication interfaces

nRF52832

The nRF52832 differs significantly from the other microcontrollers listed because it's not only a microcontroller, but it also features full hardware support on-chip for Bluetooth 5 LE. This effectively kills two birds with one stone by providing the Bluetooth support we require baked into the microcontroller. Simplifying and shrinking our PCB design if we choose to utilize this component.

The actual bluetooth radio receiver is a 2.4GHz radio transmitter that is compatible with 1Mbps and 2Mbps Bluetooth® low energy mode. Maximum throughput on this connection would be intervals of 7.5ms with 6 packets sent per connection. Each packet contains a maximum of 20 Bytes so that's 120 bytes per 7.5ms. Discussion will have to be had as to whether or not that benchmark is acceptable for our purposes before deciding to select this as our main bluetooth communication device.[21]

As for the microcontroller itself it is a single chip, highly flexible 64 MHz, 32-bit ARM Cortex-M4F based processor with 512kB flash and 64kB RAM that implements 16 and 32-bit instructions in order to maximize code density and performance. There is also a support for single-precision floating point manipulations which will be necessary for accurate analysis of the IMU data. Typical current consumption of the CPU while active with an input voltage of 3.3Volts is around 3.7mA yielding a power consumption of around 12.2mW. It also features 32 general purpose I/O pins, a 3x 4-channel pulse with modulator, 2x I2C compatible interfaces and a 3x SPI with the ability for Easy DMA. Easy DMA allows for the writing to and reading from the Data Ram without CPU involvement.[21]

Table 3.6: Basic properties of nRF52832

Specification	**Value**	**Significance**
Clock speed	64MHz	High Speed
Memory	512 kB flash 64 kB RAM	High onboard memory
Floating Point Support	Single Precision FPU	Easy manipulation of IMU data
Supply Voltage Range (VDD)	1.7 - 3.6V	Variable voltage supply
Digital I/O Supply Voltage (VDDIO)	High: VDD - 0.4 Low: VSS	Compatible i/o range with components
Power Consumption	12.2mW (at full use)	Low power consumption
Digital Communication and I/O Formats	2xI2C(100-400KHz) 3xSPI(125KH-8MHz) UART(up to 1MHz)	Wide variety of communication interfaces

TM4C123GH6PM

The TM4C123GH6PM is a 32-bit ARM Cortex-M4 80MHz processor with a floating point unit, 256KB Flash, 32KB SRAM, and 2KB of EEPROM. The device features 8 UART modules, 4 SPI modules, and 3 I2C modules with USB 2.0 support.

Two analog to digital converters each with a maximum sample rate of one million samples/second and Pulse width modulation. It also features a mixed 16/32 bit instruction set to further empower the space conservation of its 256KB flash.[22]

This is essentially a beefed up version of the nRF52832 sporting the same architecture but featuring a faster clock rate and more digital communication interfaces. Of course it does forgo the built in bluetooth low energy support. So if in the end we decided to utilize an exterior bluetooth communication device or wanted to utilize something other than Bluetooth low energy this would be a contender due to the baked in support of a floating point unit. Of course given the clock speed on other microcontrollers it might be possible to emulate the floating point unit instruction set but doing so might be tedious.

Table 3.7: Basic properties of TM4C123GH6PM

Specification	**Value**	**Significance**
Clock speed	80MHz	High Speed
Memory	256 kB flash 32 kB RAM	High onboard memory
Floating Point Support	Single Precision FPU	Easy manipulation of IMU data
Supply Voltage Range (VDD)	3.15 - 3.6V	Low variability in voltage supply
Digital I/O Supply Voltage (VDDIO)	High: 2.4 (min) Low: 0.4 (max)	Compatible i/o range with components
Power Consumption	33 - 148 mW (at full use)	Low-moderate power consumption
Digital Communication and I/O Formats	3xI2C(100-400KHz), 4xSPI(125KH-8MHz) 8xUART(up to 1MHz)	Wide variety of communication interfaces

Considering a current draw of 10-45mA at 3.3 Volts the power consumption of this device would clock in at around 33 - 148 mW during normal operations. This is a higher current draw then the nRF52832 which features a similar architecture, this is most likely due to the TM4's robust features. Overall this chip, while powerful

and featuring a single precision floating point unit, is perhaps to powerful for our application.[22]

MSP432P401R

The MSP432P401R, like the TM4C123GH6PM and nRF52832 previously mentioned takes use of the ARM 32-Bit Cortex - M4F CPU with floating point unit and memory protection. Although this one is slightly less powerful, clocking in at at maximum frequency of 48MHz. It features a SAR analog to digital converter with 16-bit precision and up to Msps with two window comparators. Up to 256KB of Flash Main Memory with simultaneous reading and writing capability. Up to 64KB of SRAM, and 32KB of ROM initialized with MSP432 Peripheral driver libraries. The MSP432P401R also features a wide range of input voltage it can consider, accepting a range of 1.6 to 3.7 volts it is reminiscent of the nRF52832. Current consumption maxes out at around 8.4mA which at 3.3 volts yields a power consumption of 27.7mW.[23]

For communication this microcontroller has up to four eUSCI.A Modules which are compatible with UART (with automatic baud rate detection), SPI (up to 16Mbps) and IrDA encoding/decoding. Up to Four eUSCI.B Modules which can utilize either I2C(with multiple slave addressing) as well as SPI (up to 16Mbps). It accepts up 48 I/Os with interrupt and wake-up capability with a tunable DCO reaching up to 48MHz and Support for both low frequency and High Frequency Crystal Oscillators.[23]

The Peripheral driver libraries that are embedded on chip in the 32KB of ROM is an interesting, additional caveat that the MSP432P401R offers over its competitors. This support creates a software layer abstraction between the programmer and the hardware. This abstractions allows the programmer to utilize the built-in functions of the MSP432P401R to create higher level code. Saving some legwork on the lower end of things. This abstraction is also supported on other MSP430/MSP432 platforms making the software highly portable between these microprocessors.[23]

The MSP432P401R seems to be more suitable to our means than the TM4C123GH6PM since it is a less powerful module and lower power. It also has one of the second lowest clock speeds of the microcontrollers considered so far. The power consumption is also one of the lowest seen however it is worth considering that without the built in bluetooth capabilities of the nRF52832 this chip would require an external means for wireless communication which would bring this chip into a similar category of power consumption. Support for a FPU is also helpful as we can do numerical processing of the IMU data on chip which, as mentioned previously, will lighten the load on our wireless communication throughput needs.

Table 3.8: Basic Properties of MSP432P401R

Specification	**Value**	**Significance**
Specification	Value	Significance
Clock speed	48MHz	Moderate - High Speed
Memory	256 kB flash 64 kB RAM	High onboard memory
Floating Point Support	Single Precision FPU	Easy manipulation of IMU data
Supply Voltage Range (VDD)	1.62 - 3.6V	High variability in voltage supply
Digital I/O Supply Voltage (VDDIO)	High: VDD - 0.4 Low: VSS	Compatible i/o range with components
Power Consumption	27.7 mW (at full use)	Low-moderate power consumption
Digital Communication and I/O Formats	4xI2C(100-400KHz) 8xSPI(125KH-16MHz) 4xUART(up to 16MHz)	Wide variety of communication interfaces

3.5 Component Selection

The following section will outline the process through which all major components to be used in our design were selected. The major characteristics and features of all potential components reviewed in section 3.4 will be compared and contrasted, after which one will be chosen for use in our final implementation of Smart Gloves.

3.5.1 IMU Selection

Section 3.4.1 reviewed the characteristic electrical and sensor-related specifications of four IMUs from two MEMS electronics companies, Bosch Sensortec and TDK InvenSense. Through careful review and consideration of these specs, in addition to other considerations such as price, available quantity, and previous experience with certain components, our group decided that our Smart Gloves design will implement the use of the Bosch BNO055 Intelligent 9-axis Absolute Orientation Sensor. The process through which this decision was made will be thoroughly detailed in this section, with Table 3.9 providing a side-by-side comparison of IMU specifications from each device to assist in detailing the differences and similarities between each.

Table 3.9: IMU Comparisons

Specification	**ICM-20948**	**MPU-9250**	**BMX055**	**BNO055**
Avg Power Consumption (VDD=3.3V)	10.3mW	12.2mW	18.6mW	40.6mW
Operating / Logic Voltage	1.71-3.6V / 1.7-1.9V	2.4-3.6V / 1.7-3.6 V	2.4-3.6V / 2.4-3.6V	2.4-3.6V / 1.7-3.6V
Digital Interface	I2C, SPI	I2C, SPI	I2C, SPI	I2C, UART
Accelerom. Sensitivity (LSB/g)	2g: 16,384 4g: 8,192 8g: 4,096	2g: 16,384 4g: 8,192 8g: 4,096	2g: 1,024 4g: 512 8g: 256	2g: 4,096 4g: 2,048 8g: 1,024
Gyroscope Sensitivity (LSB/dps)	250: 131 500: 65.5 1000: 32.8	250: 131 500: 65.5 1000: 32.8	250: 131.2 500: 65.6 1000: 32.8	250: 131.2 500: 65.6 1000: 32.8
Price per Unit*	$8.68	$10.63	$7.99	$12.07
Availability	In stock/no lead time	In stock/no lead time	Out of stock	In stock/no lead time

Upon initial observation, it is apparent that both TDK InvenSense IMUs consume much less power than their Bosch counterparts, with lower input currents drawn by the devices. To derive these values for approximate average power consumption, a standard operating VDD value of 3.3V was assumed for all four devices and was multiplied by the average supply current listed for each on their respective data sheets. Worth noting is that these supply current values correspond to the full operating capacity of each IMU, meaning that the accelerometer, gyroscope, magnetometer, and processor of each device are fully powered and operational in these states. Each of these four IMUs are capable of running under low power modes or suspended operation modes that further reduce power consumption for each. Compared to the BNO055, the ICM-20948 and MPU-9250 consume a quarter of the power that the BNO consumes during normal operation with 9 DOF enabled; however, the BNO055's low power mode allows typical power consumption to fall as low as 9.0mW [19]. Taking this into consideration, we came to the conclusion that the power consumption of each IMU should not be a major deciding factor in device selection; even at full operation, the power consumption of both Bosch devices is not so high that it can't be mitigated by a strong enough battery.

One area in which these TDK InvenSense and Bosch Sensortec IMUs differ is in the sensitivity of their integrated MEMS accelerometers. On Table XX sensitivity of the accelerometers are displayed in terms of LSB/g, which essentially translates into how many quantizations of each g can be represented by the available bit count. Because the MEMS accelerometers produce an analog output within each device, a digital output is obtained through the use of integrated ADCs whose bit counts differ between devices: both TDK InvenSense IMUs sport 16-bit accelerometer ADCs while the BMX055 and BNO055 utilize a 12-bit and 14-bit ADC, respectively. To calculate LSB/g, the following formula is utilized:

$$LSB/g = 2^{(bit\,count)/(range\,of\,accel.\,measurement)}$$

For instance, to find the sensitivity of the ICM-20948 measuring across a full-scale range of +/-2g the equation above becomes 2exp(16)/4g and yields a sensitivity of 16,384 LSB/g. This number decreases by factors of 2 as the measurement range increases by factors of 2. Comparing the four IMUs, it is immediately apparent that the ICM-20948 and MPU-9250 support greater sensitivity as a result of the 16-bit ADC utilized in converting the analog output of their MEMS accelerometers to a digital format. The BMX055 suffers a decrease in sensitivity by a factor of 1/16th due to its use of a 12-bit ADC and the BNO055 offers a quarter of the sensitivity of its TDK counterparts due to its 14-bit accelerometer ADC. At this point in the project it is difficult to determine whether or not the sensitivity of each device will have a meaningful impact on the ability of Smart Gloves to accurately measure linear acceleration, but the argument can be made that both TDK devices outperform the Bosch devices in this regard. The gyroscopic sensitivity of each

device, measured in terms of LSB/dps, is the same across each due to the use of 16-bit ADCs for their MEMS gyroscopes.

One area of difference between all four IMUs that is not displayed on Table XX is the motion processing capabilities of each device. Both TDK InvenSense IMUs feature very similar motion processing capabilities, with an onboard DMP that offloads computation of motion processing algorithms from the host processor by processing data from the accelerometer, gyroscope, magnetometer, and other third party sensors. This DMP also allows for the generation of interrupts at external pins and manages power consumption at low power modes. Both Bosch IMUs, on the other hand, feature integrated 32-bit ARM cortex M0+ microcontrollers that run Bosch Sensortec 9-axis FusionLib software, which intelligently fuses sensor data across 9 axes to provide cohesive output data that gives the absolute orientation of the device in the form of quaternions, Euler angles, or raw sensor data if desired by the user.

In addition to fusing raw sensor data, FusionLib software also manages background calibration of all integrated sensors and minimizes drift/offset by applying drift cancellation algorithms such as kalman filters to the digital output of data. FusionLib software even manages the generation of user-programmable interrupts that are detected certain ranges or types of movement specified by the user. The use of this powerful 32-bit microcontroller and FusionLib software is arguably what drives the higher power consumption of both Bosch devices, even moreso with the BNO055 which employs a slightly more powerful integrated microcontroller than the BMX055. For reasons described in the previous paragraph, this increase in power consumption is completely warranted given the superior motion processing capabilities of the BNO055 compared to both TDK InvenSense devices.

The final considerations taken into account regarding these four potential IMUs were price and availability. The lowest priced device is the Bosch Sensortec BMX055, with a DigiKey price of $7.99 per unit, and the highest priced device is the Bosch BNO055 with a price per unit of $12.07. With a range in price of $4, our group determined that these prices would not be a major deciding factor in choosing the implementation of any particular device for Smart Gloves. A $4 price range for potential components might carry more weight for a company that were to produce a product like Smart Gloves on a larger scale; however, with the BNO055 arguably being the most powerful motion tracking/processing device out of the four reviewed, this jump of $4 would likely be considered worth the cost. Regarding the availability of each device, the only IMU that isn't readily available from online retailers would be the Bosch BMX055. This essentially takes the BMX055 out of final considerations, but this isn't a complete loss for our Smart Gloves project implementation given the shortcomings of the device in accelerometer sensitivity and the slightly decreased motion processing power compared to its counterpart, the BNO055.

Given the considerations and comparisons made between all four IMU contenders, we decided that the Bosch Sensortec BNO055 Intelligent 9-axis Absolute Orientation Sensor would best fit our Smart Gloves implementation with its robust motion tracking/processing capabilities. Though it has the highest average power consumption of all four devices, this can be considered a worthwhile trade-off given the inclusion of a powerful 32-bit ARM cortex M0+ microcontroller running Bosch Sensortec FusionLib software. Besides increased power consumption, the only other area that the BNO055 falls short in compared to its TDK InvenSense counterparts is in accelerometer sensitivity: utilizing a 14-bit ADC compared to a 16-bit ADC cuts the sensitivity by a factor of 1/4. Despite this, 14 bits of data is likely to be plenty given the range of measurements that would be taken with Smart Gloves, and again is a worthwhile tradeoff for the BNO055's complex motion processing capabilities. Additionally worth noting is the fact that a member of our group has previous experience working with the BNO055 in the form of a BNO055 breakout board produced by Adafruit Industries. This prior experience will help to streamline the initial integration of this device into our project.

3.5.2 Microcontroller Selection

After comparing the chips above the decision of microcontroller selection has come about. First off is consideration of the ATmega32U4. While this microcontroller is fairly commonly used and compatible with the arduino ide, it isn't an ideal selection for us. First off this unit does not have a Floating point unit on chip which means the majority of the imu calculation will either have to be calculated through emulated instruction which we might have to create ourselves or the calculations would have to be sources off-chip. If we decided to utilize this approach then we would more then likely need more data throughput on our communication channels which would throw Bluetooth low energy out the window. Other notable qualities that stand out negatively are the clock rate compared to power consumption. Looking at the other chips we have analyzed the power consumption is fairly low for the amount of processing power given. The real superior quality here is the price point. These chips cost about 5 dollars a pop, but we won't be considering this as an option going forward for the other reasons stated above. [20]

Next up on the list is the nRF52832. This unique little powerhouse pack a bluetooth low energy module onboard that in combination with the built in single precision floating point unit packs a powerful punch. With the floating point unit on-board we will no longer have to emulate instructions, and can save a lot of development time getting to market and prototyping phase. Plus since the bluetooth module is built in, that's one less thing to worry about because it's integrated with the CPU. Only caveat here is that it might be difficult to get the environment up and running, and having Bluetooth LE built into the chip might complicate the instruction set. Its maximum clock rate is also fairly strong considering the extremely low power

consumption (12.2mW at 64MHz). With High on board memory, and a variable voltage range this is a strong contender for our microcontroller.[21]

After the nRF52832 comes the TM4C123GH6PM. This little buddy is packed to the brim featuring the fastest clock rate we've seen yet. It's essentially a beefed up version of the nRF52832 due the consideration that they both run on the same 32-bit ARM Cortex-M4 architecture. This also means that this microcontroller, like the previous, has a single precision Floating Point Unit on board. This definitely simplifies the IMU calculations as previously mentioned and opens us up technology wise on the wireless communication side. Although this power does come with significant negatives. The power consumption of this chips clocks in at around 33 - 140 mW depending on processing speed when considers a 3.3 voltage input. Also the voltage supply range isn't very malleable either with it only ranging from 3.15 to 3.6 volts. Having a little bit of leg room when it comes to voltage range would be ideal since we are going to be running off of battery power at some point and we don't want our microprocessor shutting down early if there isn't enough tolerance for minor fluctuations. The real ideal situation for this microcontroller would be if you were communicating with many devices at once as it spouts 15 interfaces for you to communicate over not with protocols such as I2C, SPI, and UART. For our purposes we don't need all this additional fluff when considering microcontrollers as our interfaces will be rather simple and the power consumption of this guy is far too much to justify the negatives. [22]

Finally we can consider the last microcontroller mentioned above, the MSP432P401R. Like the previous two microcontrollers above this guy features the same architecture, the ARM 32-Bit Cortex - M4F. Meaning, you guessed it, we get a Floating Point Unit on board. I won't bother mentioning the advantage of this again but in addition to this aspect the MSP432P401R features a lower power consumption of 27.7 mw (max) at full use. With a with a wide variable range of voltage inputs this microcontroller would be a good fit for consideration with a battery attached storage since i would be able to withstand any minor voltage fluctuations. The true advantage this microcontroller has over the nRF52832 would have to the be the peripheral driver libraries that Texas instruments has included onboard. These drivers are basically software built on chip that allows a higher level programming language. Speeding up development and promoting readability within the software environment. Plus this software would then be cross compatible with the MSP430 and other MSP432 platforms furthering our flexibility in the future when considering to-market approaches.[23]

Table 3.10: nRF52832 and MSP432P401R comparison

Specification	**nRF52832**	**MSP432P401R**
Clock speed	**64MHz**	48MHz
Memory	**512 kB flash** **64 kB RAM**	256 kB flash 64 kB RAM
Floating Point Support	Single Precision FPU	Single Precision FPU
Supply Voltage Range (VDD)	1.7 - 3.6V	1.62 - 3.6V
Digital I/O Supply Voltage (VDDIO)	High: VDD - 0.4 Low: VSS	High: VDD - 0.499 Low: VSS
Power Consumption	**12.2mW** **(at full use)**	27.7 mW (at full use)
Digital Communication and I/O Formats	2xI2C(100-400KHz) 3xSPI(125-8MHz) UART($\leq$ 1MHz)	4xI2C(100-400KHz) 8xSPI(125-16MHz) 4xUART($\leq$16MHz)
Unique positive aspects	Bluetooth Low Energy Support built on chip	Peripheral driver libraries
Price per Unit*	$8.68	$10.63
Availability	In stock no lead time	In stock no lead time

**Prices sources from Digikey (10/29/17)*

After considering all of the previous microcontrollers above it's come down to two options. The MSP432P401R and the nRF52832, compared above. While the MSP432 has advantages with respect to ease of use with the Peripheral driver libraries and the fact that all of us have utilized the MSP430 development environment at some point in our own educational pursuits it fails to compensate in the other areas. The nRF52832 has a higher availability of both physical Flash memory and SRAM. This means with regards to program space we have more flexibility with how we decide to utilize it. Featuring both higher processing speed and lower

power consumption paired with Bluetooth Low Energy support on-chip there is a clear winner when it comes to power for price. The only way the MSP432P401R comes out on top is if we decided to utilize another communication profile besides Bluetooth Low Energy which, at this point we are not considering, although it is worth noting for a future consideration.

3.5.3 Accessory Component Selection

After picking our main two components that will be doing the most of the leg work, the processor and IMU, we need find way to interface these components together and combine them with the necessary interfaces such as battery, UART bridge and Linear voltage regulator. For picking these components we have decided to utilize a reference to the adafruit breakout boards for these corresponding units, the BNO055[24], and the nRF52832[25] Adafruit designs are licensed under the ShareAlike 3.0 Unported which states that we are free to share copy and adapt these materials for any purposes, even commercially as long as we give credit and release our designs under the same licenses, which we intend to.

Table 3.11: Additional component selections for PCB

Component	Part number	Price per unit*
Linear Voltage Regulator	AP2112	0.48
LIPO Charger	MCP73831/2	0.59
PNP Filtering MOSFET	DMG3415U	0.92
Schottky Diode	AP2112	0.44
USB-UART Bridge	CP2104	1.65

**Prices sources from Digikey (10/29/17)*

Since we are basing our PCB design off of an adaption of these breakout boards we are also going to purchase them at first for initial testing and prototyping. After finalizing our PCB layout adapted from these designs we intend to utilize the breakout boards to ensure working functionality. It should be noted that these PCB layouts are planned to be extremely small. Thus adequate access to proper facilities for soldering small components will be absolutely necessary. Given that our school has research laboratories dedicated to these types of things, finding this

equipment and utilizing it should not be an issue. We also require a few additional components to get properly set up. These are quality soldering iron for soldering components and wires together. A handful of jumper wires for easy interfacing and a bluetooth device for desktop computer interfacing for quick prototyping. Other things such as resistors, capacitors and the like we already have access to so we neglected them in our purchases. The breakout board prices as well as other components purchased for testing reasons are listed below and have already been ordered. Initial testing and prototyping is expected to begin on 11/4/17.

Table 3.12: Component selections for prototyping

Component	**Part number**	**Price per unit***
nRF52832 breakout board	Adafruit Feather nRF52 Bluefruit LE - nRF52832	29.95
BNO055 breakout board	Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055	30.27
Soldering kit	Tabiger Soldering Iron Kit 60W 110V-Adjustable Temperature Welding Soldering Iron with Tool Carry Case	18.99
Wires for interfacing	Z&T Solderless Flexible Breadboard Jumper Wires M/M 100pcs	7.29
Bluetooth USB Device	Plugable USB Bluetooth 4.0 Low Energy Micro Adapter	13.95

**Prices sources from Amazon (10/29/17)*

3.6 Related Software Search

Wearable devices have become very popular in most recent years. It has allowed to leaps in human health and understanding of the human body in ways that have not been possible in the past. Taking the Apple Watch as an example, apart from

telling time, Apple Watch users are able to pay using their debit or credit cards directly from the convenience of their wrist. Apple Watch users are also able to send messages, make calls, get GPS data, look at pictures, and plenty of more things all from the comfort of their wrist. Apart from all the conveniences, users also have access to health data such as heart rate, calories burnt, hours ative, numbers of steps taken, number of lights taken, the list goes on. Taking advantage of the technologies available for these health tracking activities will be essential to the design and implementation of the smart gloves.[26]

Perhaps one of the most important hardware components for motion tracking is the inertial measurement unit. Along with accelerometers and gyroscopes, electronics can provide the motion tracking data to make a product like smart gloves possible. This field of technology has been of much interest that plenty of studies and papers have been done. It is these sensing and motion measurement technologies that have paved the way the monitoring and measurement of human status and performance. Perhaps one of the biggest challenges that arise in motion tracking and measurement is having accurate readings and samples without a large number of sensors and vast environmental factors. With modern technology, the development of accurate hardware devices have been made possible.

In recent years, IMUs (inertial measurement units) have shown to adapt and be useful under many environmental factors. Optical motion has shown these type of restrictions that do not make great products in this market. Most importantly all these IMU based components are affordable while also being less intrusive. IMU measurements have been used in the past for many years to calculate the height of aerial vehicles. The advancements in commercial IMUs for wearables is mainly accredited to companies that sell and manufacture IMUs and IMU based systems.

Due to lack of experience, figuring out ways to efficiently collect data from IMUs may be a difficult task. Due to the low level application of printed circuit boards and their components, the programming language to be used will be C. C has served as a great tool for manipulating data as well as programming hardware devices at a low level such as directly interacting with components on the PCB.

As mentioned in previous sections, the Bosch BNO055 IMU will be used. This inertial measurement unit contains three triple-axis sensors to measure simultaneously tangential acceleration. This measurement is facilitated through the accelerometer. The gyroscope assists in rotational acceleration measurements while the strength of magnetic field is measured by the magnetometer. This IMU allows for two ways of interacting with its output data. Data can be sent to an external microprocessor or it can be analyzed inside the sensor. The Bosch BNO055 contains a microprocessor for running a proprietary fusion algorithm. The ability is then given to request the data in many different ways from the microprocessor. The internal chip contain interrupts that can help record certain motion that has occurred. This feature can be extremely helpful in our implementation as certain movements will

need to be tracked in order to alert or notify the user that incorrect form is being used during a particular exercise.

Before use, the IMU must be calibrated. Once calibrated the read register that holds the current calibration status will be overwritten and will be able to be used immediately the next time it is powered on. In order to successfully calibrate the IMU, a compass and a level are needed. Bosch provides a visualization tool on their website to help on the programming of this component. The gyroscope can reach a calibrated state when the device is stable without much noise. In order to successfully calibrate the accelerometer, the device containing the IMU must be rotated in 45 degrees in at least one access. Once full calibration is reached, the calibration meter on bosch's software will be filled. In order to verify that the accelerometer has been calibrated, readings on pitch, roll, and heading should be changing when the device is moved. In order to calibrate the magnetometer, a figure eight must be drawn in space. After a few rotation, the calibration meter will show the magnetometer as calibrated. The IMU has to be calibrated for ninety degrees of freedom by following the same steps as before. [27]

Once calibrating the IMU, the host microcontroller can request data for the three sensors in non-fusion mode or also request absolute and relative orientation in fusion mode. The units of measurement in which the sensor returns acceleration is in meters second squared. While returning magnetic field strength in metric ton (mT) and finally gyroscope date is return in degrees or radians per second. Also, Euler angles can be return in either degrees, radian, or quaternions while temperature is return in celsius or fahrenheit. All these unit of measurements are set on the unit selection register.

To design for versatility and so the smart gloves do not have a limited range of motion. Quaternions can be used. While Euler angles are also an option, quaternions should be able to provide the data that is necessary to accurately calculate the range of motion from the smart glove or other parts that the sensor is located. This will allow for versatility to use the smart glove sensor in different body parts for different types of workouts. The Bosch BNO055 gives all the possibilities to be able to achieve such versatility. Quaternions provides a way to multiply and divide three numbers. After being invented in the 1800s, quaternions fell stopped by use until it revival in the nuclear age and the recent developments in modern computer graphics programming. Quaternions consists of four different numbers: a scalar and three component vectors. In this case w, x, y, and z are real numbers while i, j, and k are quaternions units. However, all these numbers are less than or equal to one.

4 Standards/Design Constraints

4.1 Standards

The American National Standard Institute (ANSI) is a coordinator of the United States private sector for voluntary standardization systems. ANSI has been operating for over 90 years ”enhancing both the global competitiveness of U.S. business and the U.S. quality of life by promoting and facilitating voluntary consensus standards and conformity assessment systems, and safeguarding their integrity”. ANSI is responsible for enforcing both standards and conformity assessments. ANSI coordinates the U.S. voluntary consensus standards systems which providrd a neutral forum for the development of policies on standards issues. It addition, it serves as a watchdog for standards development and conformity assessment programs and processes within the United States and its interests.

The federation also accredits qualified organizations. These organizations are committed to developing standards in which the process meets all of ANSI's requirements. Curiously enough, ANSI itself does not develop standards. The organization simply oversees all of the standards development. In addition, ANSI represents U.S. interests in regional and international standardization activities while overseeing conformity assessment activities that promote the global acceptance of U.S. products, services, systems, and personnel. Before we can stress the importance of this organization, we must first specify some definitions.

A **standard** is a document, established by consensus that provides rules, guidelines, or characteristics for activities or their results. (As defined in ISO/IEC Guide 2:2004)

A **conformity assessment** is defined as any activity concerned with determining directly or indirectly that relevant requirements are fulfilled. (As defined in ISO/IEC Guide 2:2004)

While a standard is a technical expression of how to make a product safe, efficient, and compatible with others, a standard alone cannot guarantee performance. However, conformity assessment provides consumer assurance by increasing consumer confidence. This confidence is achieved when personnel, products, systems, processes or services are evaluated against the requirements of a voluntary standard. The large body of conformity helps establish trust with the consumer base.

Software Quality Assurance (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality. The methods by which

this is accomplished are many and varied, and may include ensuring conformance to one or more standards, such as ISO 9000 or a model such as CMMI.

SQA encompasses the entire software development process, which includes processes such as requirements definition, software design, coding, source code control, code review, software configuration management, testing, release management, and product integration. SQA is further broken down and organized into goals, commitments, abilities activities, measurements, and verifications.

4.1.1 Standard SystemC Language Reference Manual

IEEE Std 1666 2011 is a standard which defines SystemC. SystemC is an ANSI standard C++ class library for system and hardware design. This standard is made available for use by designers and architects who need to address complex systems that are a hybrid between hardware and software. The SystemC Standard provides a precise and complete definition of the SystemC class library so that a SystemC implementation can be developed with reference to this standard alone. The primary audiences for this standard are the implementers of the SystemC class library, the implementers of tools supporting the class library, and the users of the class library.

4.1.2 Objective-C

Objective-C is a general purpose, object oriented programming language that adds Smalltalk-style messaging to the C programming language. It was the main programming language used by Apple for the OS X and iOS operating systems. Objective-C also collaborates with the respective application programming interfaces (APIs) Cocoa and Cocoa Touch prior to the introduction of Swift. Objective-C's features often allow for flexible, and often easy, solutions to programming issues.

These solutions are achieved by delegating methods to other objects. Also remote invocation can be easily implemented using categories and message forwarding. Other intricacies of Objective-C is allowing for swizzling of the isa pointer to allow for classes to change at runtime. The swizzling of the isa pointer is typically used for debugging where freed objects are swizzled into zombie objects whose only purpose is to report an error when someone calls them. Swizzling was also used in Enterprise Objects Framework (EOF) to create database faults. Swizzling is used today by Apple's Foundation Framework to implement Key Value Observing (KVO).

The Objective-C environment is a growing collection of tools and reusable components (Software ICs) for large-scale production system-building is discussed. Its goal is to make it possible for its users to build software systems in the way that hardware engineers build theirs. This can be done by reusing Software ICs supplied by a marketplace in generic components rather than by building everything from scratch. The Objective-C environment is based on conventional technology (C and Unix-style operating systems), which it includes and extends. The extensions presently include a compiled and an interpreted implementation of Objective-C (an object-oriented programming language based on C) and several libraries of reusable components (ICpaks).

4.1.3 Objective-C 2.0

At the 2006 Worldwide Developers Conference, Apple announced the release of "Objective-C 2.0". Objective-C 2.0 is meant to be a revision of the Objective-C language to include "modern garbage collection, syntax enhancements, runtime performance improvements, and 64-bit support". Mac OS X v10.5, released in October 2007, included an Objective-C 2.0 compiler. GCC 4.6 supports many new Objective-C features, such as declared and synthesized properties, as well as dot syntax, fast enumeration, and optional protocol methods. Additionally, Objective-C 2.0 features method/protocol/class attributes, class extensions and a new GNU Objective-C runtime API.

4.1.4 Swift

Apple released Swift, a modern programming language to be the successor of Objective-C. In less than a year and a half after its first release, Swift became one of the most popular programming languages in the world, considering different popularity measures. A significant part of this success is due to Apple's strict control over its ecosystem, and the clear message that it will replace Objective-C in a near future.

Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. for iOS, macOS, watchOS, tvOS, and Linux. Swift is meant to be a powerful and intuitive programming language. Writing Swift code is presented by Apple as being interactive and fun, with the syntax being both concise yet expressive. The ease of writing Swift code is mostly due to its design. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks as well as the large body of existing Objective-C code written for Apple products. It is built with the open source Low Level Virtual Machine (LLVM) compiler framework and has been included in Xcode since version 6. On platforms other than Linux, it uses the

Objective C runtime library which allows C, Objective C, C++ and Swift code to run within one program.

Swift is similar to C in various ways. Both have the benefit of generic programming with class methods being inherited. In addition, functions are considered first class objects. More importantly, several notoriously error-prone behaviors of earlier C-family languages have been changed. A short list of these improvements are pointers are not exposed by default, there is no need to use ”break” statements in ”switch” blocks, and variables and constants are always initialized and array bounds are always checked. In addition to basic syntax and initialization there are improvements to integer overflows. These overflows result in undefined behavior for signed integers in C, are trapped as a runtime error in Swift. Programmers can choose to allow overflows by using the special arithmetical operators “&+”, “&-”, “&*”, “&/”, and “&%”. The properties “min” and “max” are defined in Swift for all integers types and can be used to safely check for potential overflows, versus relying on constants defined for each type in external libraries.

4.1.5 Java Standards

Coding conventions are one of the most important aspects of software development. Roughly 80% of the lifetime cost of a piece of software goes to maintenance. This staggering cost is due to the fact that software is rarely maintained by the original author for its entire life. As new people get introduced to the software, there is a learning curve which costs both time and money. Unfortunately, few resources are dedicated to documentation. This only further complicates the learning curve problem. [28]

Fortunately, standards such as coding conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly. Skills and understanding of one code base may transfer easily to another. Standardization also lends way to presentation. If developers ship source code as a product, professional presentation goes a long way. The software must be well packaged and clean; just like any physical product that ships.

4.1.6 PCB Design Standards

IPC, the Institute for Printed Circuits, is a trade association whose aim is to standardize the assembly and production requirements of electronic equipment and assemblies. Its name was later changed to the Institute for Interconnecting and Packaging Electronic Circuits to highlight the expansion from bare boards to packaging and electronic assemblies. IPC is accredited by the American National Standards Institute (ANSI) as a standards developing organization and is known glob-

ally for its standards. It publishes the most widely used acceptability standards in the electronics industry. IPC standards are used extensively by the electronics manufacturing industry. IPC-A-610, Acceptability of Electronic Assemblies, is used worldwide by original equipment manufacturers and EMS companies.

The following are standards that the Smart Gloves project must be in compliance with. This is for both mandatory regulation and for the benefit of the consumer base. The standards outlined below help to allow for easier design and manufacturability of the product. In addition to the design and manufacturing benefits, the standards below will allow for both a safer and more user friendly product.

IPC-2612 Sectional Requirements for Electronic Diagramming Document

The IPC-2612 standard establishes the requirements for the documentation of electronic diagrams. This standard will be used as the foundation for defining the electrical interconnectivity of electronic parts. The description pertains to either schematic diagrams, logic diagrams or Boolean truth tables and includes methodology for defining circuit flow, electrical or functional restrictions, as well as maintenance test procedures used to design or maintain the electronic product. The requirements encompass everything such as hard copies, electronic copies or electronic data descriptions. [29]

The purpose of the IPC-2612 standard is to establish a consistent set of naming conventions, schematic and logic attributes, and documentation standards. The standard should ensure that all schematics and logic descriptions contain the information required for inspection, hardware realization, software development, and design reuse. The initial schematic logic diagram designates the electrical functions and interconnectivity to be provided by the printed board and its assembly. The standard also specifies that the schematic should define, when applicable, critical circuit layout areas, shielding requirements, grounding and power distribution requirements, as well as the allocation of test points, and any pre-assigned input/output connector locations. Schematic information may be generated as hard copy or computer data (manually or automated).

IPC-2223 Sectional Design Standard for Flexible Printed Boards

This standard establishes the specific requirements for the design of flexible and rigid-flexible printed board applications and its forms of component mounting and interconnecting structures. The flexible materials used in the structures are comprised of insulating films, reinforced and/or non-reinforced, dielectric in combination with metallic materials. These interconnecting boards may contain single, double, multilayer, or multiple conductive layers and can be comprised wholly of flex or a combination of both flex and rigid. [30]

IPC-7351B Generic Requirements for Surface Mount Design and Land Pattern Standards

This document provides information on land pattern geometries used for the surface attachment of electronic components. The intent of the information presented herein is to provide the appropriate size, shape and tolerance of surface mount land patterns to insure sufficient area for the appropriate solder fillet to meet the requirements of IPC/EIA J-STD-001, and also to allow for inspection, testing, and rework of those solder joints. [31]

Although, in many instances, the land pattern geometries can be different based on the type of soldering used to attach the electronic part, wherever possible, land patterns are defined with consideration to the attachment process being used. Designers can use the information contained herein to establish standard configurations not only for manual designs but also for computer-aided design systems. Whether parts are mounted on one or both sides of the board, subjected to wave, reflow, or other type of soldering, the land pattern and part dimensions should be optimized to insure proper solder joint and inspection criteria. Land patterns are dimensionally defined and are a part of the printed board circuitry geometry, as they are subject to the producibility levels and tolerances associated with plating, etching, assembly or other conditions. The producibility aspects also pertain to the use of solder mask and the registration required between the solder mask and the conductor patterns.

IPC-2615 Printed Board Dimensions and Tolerances

The purpose of this Standard is to establish acceptable principals and practices for dimensioning and tolerances used to define end-product requirements for printed boards and printed board assemblies. This Standard covers dimensioning and tolerances of electronic packaging as it relates to printed boards and the assembly of printed boards. The concepts defined in this Standard are derived from ASME Y14.5M- 1994. Printed boards have such wide applications that there may be times where this standard does not address a specific case. In those cases, the user is referred to ASME Y14.5M 1994 for use of additional dimensioning and tolerances concepts. This Standard covers dimensioning, tolerances, and related practices for use on printed board drawings and in related documents. Uniform practices for stating and interpreting these requirements are established herein. [32]

IPC-D-325 Documentation Requirement for Printed Boards

This standard establishes requirements and other considerations for the documentation of printed boards and printed board assemblies. The purpose of this stan-

dard is to establish the general requirements for the preparation of drawings necessary to fully describe end product printed boards, printed board assemblies and related support drawings. Special emphasis is given to the technical requirements necessary to fully describe the fabrication and assembly of various types of printed boards. Regardless of material, construction, layer count, special fabrication requirements, or end product usage, the documentation package may include, but not be limited to the following:

- Master Drawing Requirements
- Specifications
- Board Definition
- Artwork/Phototooling
- Soldermask Requirements
- Master Pattern Drawing
- Production Master
- Assembly Drawing and Parts List
- Electrical Test Requirements
- Final Schematic/Logic Diagram
- Related Support Drawings
- Artwork Plot Data
- Excellon Drill Data

Refer to IPC-D-275, 'Design Standard for Rigid Printed Boards and Rigid Printed Board Assemblies', regarding all subjects pertaining directly to design. This standard may be used for both commercial and military applications. Printed boards and printed board assemblies intended for military usage shall be fabricated and/or assembled by a manufacturer that has been qualified to the appropriate military specification, unless otherwise agreed to contractually. [33]

IPC-6011 Generic Performance Specification for Printed Boards

This specification establishes the general requirements for printed boards and the quality and reliability assurance requirements that must be met for their acquisition. The intent of this specification is to allow the Printed Board user and supplier flexibility to develop optimum procedures for the Manufacture and Procurement of Printed Boards. [34]

4.1.7 Performance Classes

Three general classes have been established to reflect progressive increases in sophistication, functional performance requirements and testing/ inspection fre-

quency. It should be recognized that there may be an overlap of equipment categories in different classes. The user has the responsibility to specify in the contract or purchase order the performance class required for each product and shall indicate any exceptions to specific parameters, where appropriate.

Class 1

General Electronic Products — Includes consumer products, some computer and computer peripherals suitable for applications where cosmetic imperfections are not important and the major requirement is function of the completed printed board.

Class 2

Dedicated Service Electronic Products — Includes communications equipment, sophisticated business machines, instruments where high performance and extended life is required and for which uninterrupted service is desired but not critical. Certain cosmetic imperfections are allowed.

Class 3

High Reliability Electronic Products — Includes the equipment and products where continued performance or performance on demand is critical. Equipment downtime cannot be tolerated and must function when required such as in life support items or flight control systems. Printed boards in this class are suitable for applications where high levels of assurance are required and service is essential.

IPC-6012 Qualification and Performance Specification for Rigid Printed Boards

This specification covers qualification and performance of rigid printed boards. [35]

- The printed board may be single-sided, double-sided, with or without plated-through holes.
- The printed board may be multilayer with plated-through holes and with or without buried/blind vias.
- The printed board may be multilayer containing build up HDI layers conforming to IPC-6016.
- The printed board may contain active embedded passive circuitry with distributive capacitive planes, capacitive or resistive components.
- The printed board may contain a metal core or external metal heat frame, which may be active or nonactive.

4.2 Realistic Design Constraints

The Smart Gloves project includes various hardware and software design constraints that needed to be addressed while designing the system. It is important to keep in mind that this will be a device that users will wear on their body. With this in mind, ethical, health, and safety constraints. As well as environmental, social, and political constraints will be discussed in detail to provide insight in constraints during the brainstorming and design process.

Keeping cost and time constraints in mind are important when choosing the parts to make up this Smart Gloves sensor. During the initial phases of brainstorming, it was discussed to dig deep into hardware components that would facilitate the efforts implementing a sensor that would provide feedback while a user is performing various types of exercises. It was quickly figured out that understanding every minute detail of all hardware and software tools and components would take a large amount of time that would not be beneficial to the progress of this project. Through guidance of project advisors, the green light was given to model research and schematics completed by manufactures. As a result, time has been spent in the variety of components available to accomplish this project. It was also very important to keep the budget of this project at a minimal. Not only would this push for understanding tradeoffs and benefits that come with different design but it also allows for the interest of people of who would be interested in investing in this product if we so choose to continue with mass production. As a good rule of thumb, it is good to keep cost a minimum while maximizing features and quality at an overall high. Wearables have become common while also being a fashion statement. Keeping these in mind, allows to design a product that is attractive to use without attracting unneeded or unwanted attention. Lastly, this product will use a number of sensors. Understanding the health risks of the signals these sensors emit is important to ensure that the user is not exposed to a harmful environment. These topics will be discussed in the subsections to come.

4.2.1 Time and Economic Constraints

Time Constraints

The Smart Gloves project is an idea created by the students of the University of Central Florida. Because the idea was created by the students of the university, it was not presented as a fully fleshed out project. A company or corporation did not hash out all the details of the design and the implementation. That being said, the Smart Gloves team must dedicate a significant amount of time and resources into research and development. The hardware decisions for the project must be carefully considered as the project develops further.

The time constraints of this project are mostly influenced by the development of the PCB (Printed Circuit Board) as well as the development of the software for the motion tracking. The timeline for the fabrication of the PCB can be rather large. In addition to the PCB fabrication, the design and 3D printing of the plastic housing can take some time. The research, design, and documentation of the Smart Gloves project must be completed by December 4th, 2017. The software deliverables, fabrication, and assembly of the hardware must be finished by May 2018. It is important to have certain checkpoints along a timeline for this entire process. At each checkpoint, the Smart Gloves team can evaluate the progress and the teams velocity to adjust the schedule accordingly.

Economic Constraints

The economic constraints of this project are the budgets of the team members of this project. The Smart Gloves project does not have any University of Central Florida sponsors or general funding from companies. Due to this restriction, certain components and parts that we would like to use may not be accessible to the Smart Gloves team for economic reasons.

4.2.2 Environmental, Social, and Political Constraints

Environmental Constraints

There are a number of different fitness trackers that incorporate motion tracking technology. While having the latest and greatest electronic devices can be fun, necessary, or both, our addiction to having them comes at a high cost to ourselves and our planet. Enormous amounts of raw, often limited, resources are used to make these devices. In order to create these devices, there are intensive levels of energy and work required to both design and assemble, as well as package, ship, and deliver them around the world to consumers. In addition to the creation cost for that initial device for a consumer, it is known that millions of devices are replaced, once, twice, maybe even three times a year by customers eager for something better. This stems from replacing a broken device or simply wanted the latest and greatest. Doing so results in a large amount of hazardous household waste that is often irresponsibly discarded in the trash. When electronics are not recycled properly, the raw materials inside them leech toxic chemicals into the ground and the surrounding area, potentially spoiling both our water and food supply for decades. For this reason, it is extremely important to recycle electronic devices in a proper manner. The EPA estimates that over 438 million electronic devices were sold in 2009, which is twice the amount sold in 1997. At this rate, that number will jump to over a billion devices per year in no time. That being the case, it's essential to recognize the damage we can do to the environment and our children and grandchildren by simply throwing an old device in the trash. [36] [37]

The explosive growth of the electronics industry has led to a rapidly escalating issue end-of-life (EOL) electronics and e-waste. Internationally, only 10-15 percent of the gold in e-waste is successfully recovered while the rest is lost. Ironically, the electronic waste contains deposits of precious metal estimated to be between 40 and 50 times richer than ores mined from the earth, according to the United Nations. Because of the explosive growth in the electronics industry, combined with short product life cycle has led to a rapid escalation in the generation of solid waste. Old electronic devices contain toxic substances such as lead, mercury, cadmium, and chromium, and proper processing is essential to ensure that these dangerous materials are not released into the surrounding environment. These electronic wastes may also contain other heavy metals and potentially toxic chemical flame retardants. The uncontrolled movement of e-waste to countries where cheap labor and primitive approaches to recycling have resulted in health risks to local residents exposed to the release of toxins continues to an issue of concern.

The Smart Gloves team must source our parts and components ethically. Another facet for the Smart Gloves team to look at is the ethical fabrication of our plastic housing. The fabrication parts and materials must be able to be recycled. Each step of the manufacturing process and life cycle of the devices must be carefully considered when thinking about the environment. The choice to recycle must also be made simple for the consumer to access.

Social Contraints

Much less is known about the social impact of sport and physical recreation. However, in recent years, there has been an increasing focus on and interest in identifying such impacts.

Political Constraints

Within the medical community, physical therapy doesn't seem to carry the same amount of respect as the other disciplines. Physical therapists are often looked upon as a lesser kind, and liken them to being a slightly more intelligent personal trainer. Helping to create a product to replace the use a physical therapist might further those negative stereotypes. If a product can replace that service, the service that a physical therapist offers may come into question. [38]

4.2.3 Ethical, Health, and Safety Constraints

Ethical Constraints

The underlying precept that should guide all business-related endeavors by health and fitness products is honesty. Health and fitness products and their companies

have a fundamental responsibility to be both truthful and law-abiding. Their words and behavior should be totally free from deceit, fraud, and dishonesty. Honesty involves more than merely avoiding criminal or illegal acts. Rather, it is the cornerstone of having character and respect for others. The consumers of such products are not to me preyed upon or manipulated for financial gain, or gain of any kind. [39]

Health Constraints

Safety is a priority. From designing a car, computers, bridges, to something as small as a microcontroller, if it's not safe then people will not use the product. It is that plain and simple. Keeping safety first and incorporating it into the design process will allow for implementations that are also safe. In the case of the smart gloves, utilizing components like batteries that won't burn or cause a fire or cables that get in the way of the user while exercising or something as minimal as ensuring that the companion application does not drain the battery of the phone are safety factors to keep in mind. Following design practices from wearables out in the market are a great start as they provide a design that works with standards that users are accustomed to.

When designing a product that a person must attach to their body, it is important to design a product that is as least invasive as possible. While maintaining a simple and not invasive design will attract users and promote intuitive use, it is also important to understand health, ethic, and safety constraints. The technologies being used in this product and project are technologies that have been used in wearables by the most popular companies. Taking the Apple Watch for example, it contains a gyroscope, accelerometer, and magnetometer. These are popular hardware components that provide the data needed to track user's movements. With the backing and use of these hardware technologies by a large company like Apple, the use of these technologies can be assumed to be safe and also are components that people are comfortable wearing.

The positive impact of participation in sport and active recreation on physical health is now well accepted. Research has identified a wide range of sport-induced health benefits including improving cardiovascular health and assisting in the development of strength and balance. In light of this, governments at all levels have become increasingly active in encouraging people to adopt physical activities as a regular part of their lifestyle.

Research has show strong correlations between exercise and psychological benefits for participants. The majority (90%) of studies examined both the anti-depressive properties of exercise and the effect of exercise in combating anxiety. In addition, the studies generally substantiate the claim that improved mood is associated with exercise. A non-aerobic activity (weight training) was found to have equally positive

effects on the alleviation of depression as an aerobic activity (running) suggesting that positive impacts on mood resulting from exercise may not be dependent on an increase in aerobic capacity. Exercise and weight-training release endorphins. Endorphins are neurotransmitters that prevent pain, improve mood, and fight depression. An increased in endorphins naturally reduces stress and anxiety. Endorphins also stimulate the mind, improving alertness and boosting energy. Weight-training can brighten your entire day or help you combat a bad one. [40]

As you age, it is very important to protect your bones, joints and muscles. Studies have shown that doing aerobic, muscle-strengthening and bone-strengthening physical activity of at least a moderately-intense level can slow the loss of bone density that comes with age. This can help ward off things such as osteoporosis. Muscle-strengthening activities can help you increase or maintain your muscle mass and strength. Slowly increasing the amount of weight and number of repetitions you do will give you even more benefits, no matter your age. Weight training also increases strength in connective tissues and joints. Strong joints, ligaments, and tendons are important to prevent injury and can relieve pain from osteoarthritis. Strengthening muscles and connective tissue will make injury from daily tasks and routine exercise less likely, and can even improve sports performance. [41] [42]

The smart gloves provide highly beneficial health benefits to the user. Exercise should be a behavior people find fun in. Unfortunately injury does happen. This is where the fun is taken away. In order to prevent injury, these smart gloves will help the user maintain exercising techniques that are safe to the body and reduce movements that can lead to injury. The goal is to create a feedback system that is not distracting to the user. This feedback system will inform the user when bad forms or movements are happening. Bad health consequences are not anticipated but are not ruled out at this moment.

Physical therapy and exercise can be used as a preventative measure for injury. A customized physical therapy program can help individuals return to their prior level of functioning, and encourage activities and lifestyle changes that can help prevent further injury and improve overall health and well being. Primary care doctors often refer patients to physical therapy at the first sign of a problem, since it is considered a conservative approach to managing problems. When a patient begins physical therapy, a patient will get screened for fall risk. If a patient is at high risk for falls, therapists will provide exercises that safely and carefully challenges balancing skills as a way to mimic real-life situations. These physical therapy exercises improve coordination and aid with safer walking. [43]

4.2.4 Manufacturability and Sustainability Constraints

The manufacturability of the Smart Gloves will be relatively simple. The manufacturing constraints of the product will be limited by time, money, and volume of parts needed.

4.2.5 Tooling

The Smart Glove product will most likely use injection molding to create the housing for all the sensors and other electrical components. Injection molding uses highly pressurized melted plastic that is forced into a metal mold in the shape of the desired housing. These molds hold the plastic until the plastic has cooled and hardened. The time to shape and create these molds depends on the complexity of the housing.

4.2.6 Engineering Validation Test (EVT)

The first batch of plastic components are called first shots. These are typically in small batches of less than 100 units. These pieces will be assembled and checked for defects in both the design and the manufacturing process. There are cosmetic things to think about such as colors as well as lines, creases, and other deformations in the plastics. [44]

Electrically, there can be issues. There can be bugs from the parts sourced by various vendors. Electrical components such as LED's can vary in brightness, connections on the boards may be incomplete, and some components may have variance that isn't listed in the specifications.

The Engineering Validation Test is mostly comprised of manufacturing a few units, trying to fix everything that is broken, rinse, and repeat. EVT is the first time the engineers will try to build units using manufactured parts, not prototyped parts.

At the end of the EVT, engineers should be able to consistently manufacture units that work like and look like the prototypes. All the tweaks, modifications, changes, and retesting can take 6 weeks.

4.2.7 Design Validation Test (DVT)

The Design Validation Test consists of finalizing the cosmetics, performing final testing, and setting up the first assembly line. The parts of this step are mostly

tiny details. This is where the fine details are looked at for the finishing touches on the product enclosure. The molds can be textured to have the right look and feel (glossy, matte, sanded, stippled, etc.) The parts may be treated even further after injection molding with things like powder coating, anodizing metal parts, etc.

While these steps are taking place, there are critical tests to determine if the product will survive the real world such as drop testing, thermal and environmental testing, shock and vibration testing, etc. This is where the engineering team can be performing pre-scans and evaluations for any certifications the team wants to have their product such as Federal Communications Commission (FCC), UL, Conformite Europeene (CE), Norma Oficial Mexicana (NOM), etc.

The assembly line is a living process. The line consists of the people who work at the factory assembling the electronic devices. It is the job of the engineers to design an assembly process that organizes the workers in a way that is simple, intuitive, reasonably error-proof, and accomplishes the goal of assembling the product. Generally, the organization takes the form of a set of stations and checkpoints on a factory floor that transforms your device from a bucket of parts to an assembled and tested product. The production line may need to be threaded as to avoid bottlenecks.

A properly designed line will check each component separately, assemble the components, and then do an end-to-end functional and cosmetic test of the product. It may also program the firmware and/or serialize each unit (give it a unique identifier). Above all, a line will flow smoothly from start to finish and detect any problems with the unit.

You don't exit the DVT stage until you are consistently manufacturing and smoothly assembling units that pass all tests; cosmetic, compliance, reliability, etc. This will require several rounds of iterating on both your engineering design and your assembly line setup.

4.2.8 Production Validation Test (PVT)

After the Engineering Validation Tests and the Design Validation Tests, you can consistently manufacture and assemble functional and cosmetically polished units. From here, the assembly lines need scaled up to meet full production.

All of the challenges you faced setting up your first assembly line during DVT will be amplified. Lines will need to branch and intersect, test fixtures will get overloaded, weaknesses in your documentation will become more apparent. You'll need to add more infrastructure to the process - more tests, checkpoints, test fixtures, etc. All of the bumps and kinks will need to be smoothed over, until all of your assembly lines are running as smoothly as your single line did in DVT.

4.2.9 Test Fixtures

As an early stage hardware setup, you'll want to test every board that is manufactured. You want to verify that, as the manufacturing process progresses, you're not getting any defects in manufacturing or programming your boards. An error loading your pick-and-place machines can result in the wrong resistor value getting loaded onto a board.

To test every PCB quickly and effectively, we use test fixtures. These fixtures interface with your board and verify some aspect of its performance. Test fixtures come in many different flavors. The most well known is the In Circuit Testing (ICT) fixture, also known as a "bed of nails". An ICT fixture consists of a grid of metal probes designed to touch test points on your PCB. The fixture will then measure the voltages at those test points, and can flash an alert if any values are unexpected. It's this kind of test that could detect an incorrect resistor on your board. A test fixture can measure anything a sensor can. Test fixtures can have microphones, cameras, magnetometers, temperature sensors, the list goes on. These fixtures measure any relevant parameter to your PCB's operation and assure you that the board is working properly.

A good manufacturing effort involves building two products; your company's actual product, and the test fixtures that validate its functionality. It helps that many good CM's have the capability to develop test fixtures for you in-house. But they'll need support from your electrical team (test point layout, an electrical test plan), firmware team (custom firmware that cycles the device through its possible states), and possibly even your software team (custom LabView and/or Python scripts to run the tests).

5 Project Hardware and Software Design Details

With all relevant project research having been completed, all components for implementation in our final design selected, and all relevant standards and design constraints reviewed, this following section will break down the preliminary hardware and software design of our Smart Gloves project. This section of the report will focus primarily on the proposed design of Smart Gloves; all component or software testing will be discussed in Section 6.0. Worth emphasizing is the fact that all design details to be discussed in this section are part of our proposed project design: minor details will be subject to change during the construction of our final project prototype during the following semester. Any changes made then that differ from the design specified in this section will be recognized and updated. Figure 5.1 provides a generalized outline of the hardware side of our project through the form of a simplified block diagram.

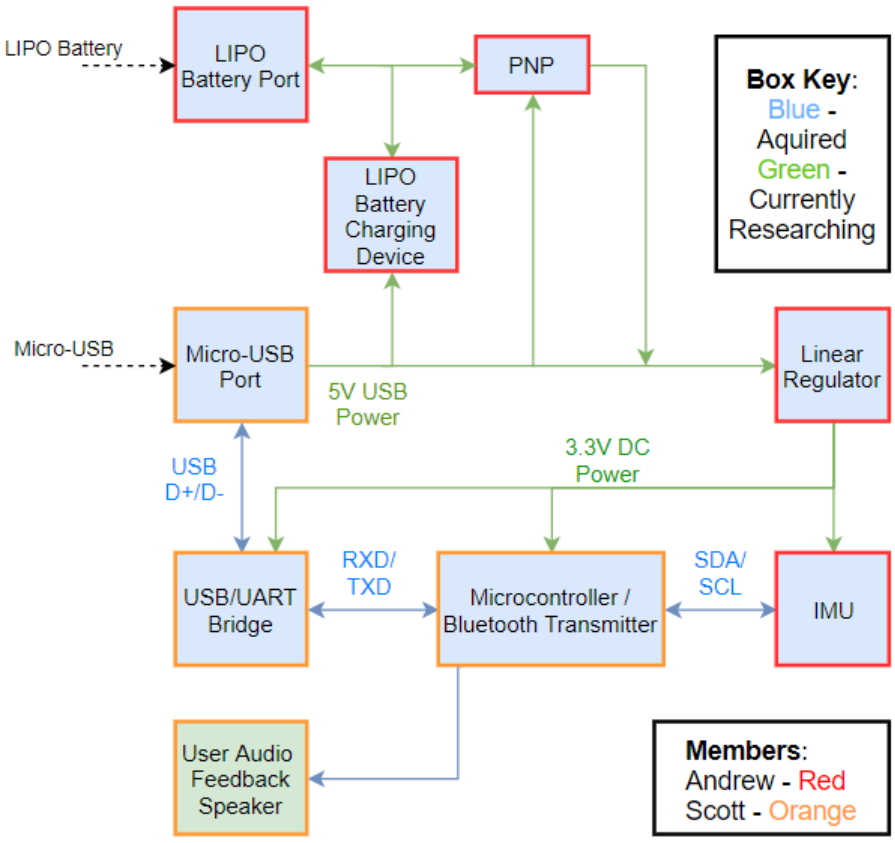


Figure 5.1: Hardware Block Diagram

This section will first begin with an analysis of the proposed hardware design for Smart Gloves. Starting with a brief and mostly conceptual review of our design, the specifics will be discussed as each sub-system is individually reviewed. There are three main sub-systems that will constitute our Smart Gloves design:

- Power regulation circuitry, including USB and LIPO battery power
- Microcontroller integration and connection to micro-USB hub with USB/UART bridge device
- Inertial Measurement Unit integration

These three sub-systems will be individually analyzed, with all relevant schematics included to supplement the provided descriptions of sub-system design. The schematics for overall system design will be provided following the conclusion of all three sub-system design reviews.

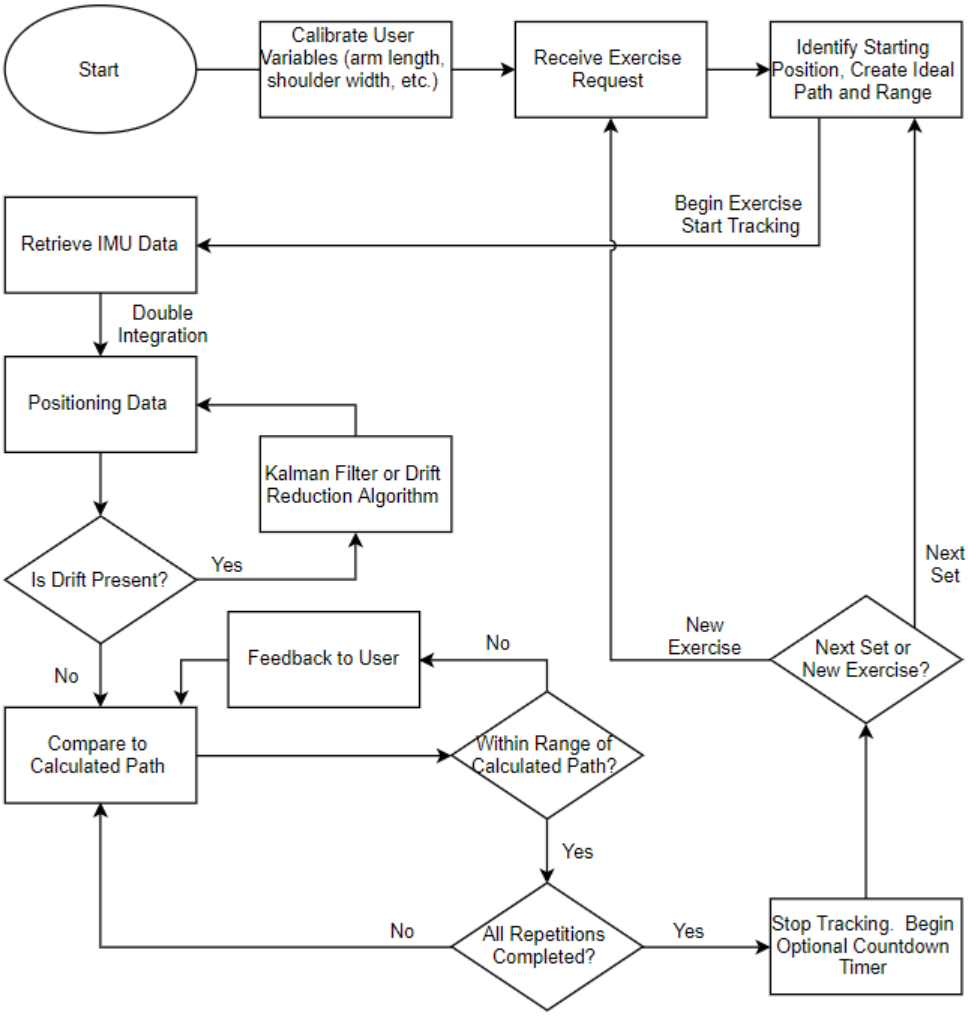


Figure 5.2: Software Block Diagram

With intelligent software design being highly critical to the operational functionality of Smart Gloves, Section 5.2 will analyze the proposed software design aspects of our project. Figure 5.2 provides a generalized conceptualization of how the software flow of Smart Gloves will perform during standard operation. Like with the review of our proposed hardware design, this section will begin by giving a brief and generalized review of the function of Smart Gloves software both within the glove-mounted electronics and user-operated smartphone application. A few of the key aspects of software design that will be individually analyzed are as follows:

- Software development environment within embedded system and user smartphone application
- Processing of output information from IMU in form of either quaternions or Euler angles
- Mathematical modeling of weightlifting motions and comparison to IMU data

These three topics will be the highlight of all included software design discussions, block diagrams, and schematics, but additional information will be discussed as necessary provided its relevancy to the overall software design of Smart Gloves.

5.1 Hardware Design

This section of the report will provide the specific design details of our Smart Gloves project concerning the three major sub-systems that will compose the bulk of our hardware design: power regulation circuitry, the microcontroller and USB/UART bridge, and the IMU. Figure 5.1 displays a generalized view of our proposed hardware design in the form of a simple block diagram.

The top half of this block diagram describes the connection between components used in the power regulation sub-system of our proposed Smart Gloves hardware design (the green connecting arrows show the flow of power in this device). There are two main power sources that will be able to power the device; while a micro-USB plug is connected, the 5V power supply will run through a 3.3V linear regulator and provide power to all embedded components. While plugged in, the micro-USB power supply will provide current to the attached LIPO battery through a LIPO charging module that passes current through until the battery has been fully charged. While the 5V micro-USB power supply is attached, a PNP transistor will prevent current from the LIPO battery from passing through to the linear regulator. This will allow the device to adapt to use either power source depending on whether the micro-USB connection is attached. The micro-USB port will not be used during regular user operation of Smart Gloves however, as a wired connection would prevent users from achieving a full range of motion. Instead, this micro-USB port

will be used to upload embedded code to the microcontroller through a USB/UART bridge and to charge the LIPO battery that will be the primary source of power during standard user operation. When not attached to a micro-USB plug, the PNP transistor will allow current to flow from the LIPO battery through the device. The 3.3V linear regulator will provide a constant 3.3V DC voltage that will power all embedded devices.

The bottom half of the block diagram shown in Figure 5.1 visualizes the connection and integration of all data transmitting components, with datapaths represented by blue arrows. The micro-USB port included will not only provide a current through which the LIPO battery can charge, but will also be used to upload all embedded software to be used by the microcontroller/Bluetooth transmitter. Because the direct interface of USB and microcontrollers can be challenging, a USB-to-UART bridge will be utilized to bridge the gap between the two. Both USB D+ and D- connections will be funneled through the bridge, which will be interfaced with the microcontroller via UART communication protocols. This microcontroller, which supports full Bluetooth transmission capabilities, will be connected to the IMU using I2C communication protocols. The data from the IMU will be transmitted to the microcontroller, which will in turn process this information and transmit it through Bluetooth to a user's smartphone where an application will further analyze the data. An auxiliary device such as a small speaker will be attached to and powered by the microcontroller, functioning as a feedback device that will be activated to notify users when it is necessary for them to change their form during an exercise.

While Figure 5.1 gives a generic overview of how our Smart Gloves hardware will be set up, the schematic diagram featured in Figure 5.3 provides a detailed analysis of the proposed implementation of all electronic components to be featured in our design. All power regulation components and sub-circuits are displayed in the top half of this diagram while the bottom half shows all processing components, including the microcontroller/Bluetooth transmitter, USB-to-UART bridge, and IMU. This design is based heavily off two open-source breakout board designs provided on Github by Adafruit Industries, the first being their Adafruit Feather NRF52 Bluefruit LE development board and the second their Adafruit BNO055 Absolute Orientation Sensor. Both designs are open-source and free for the public to use and modify, and the schematics for each can be found in Appendix A of this report. Sections 5.1.1, 5.1.2, and 5.1.3 of this report will further break down each included sub-system, providing an in-depth analysis of the components used and their connections to other components in the system.

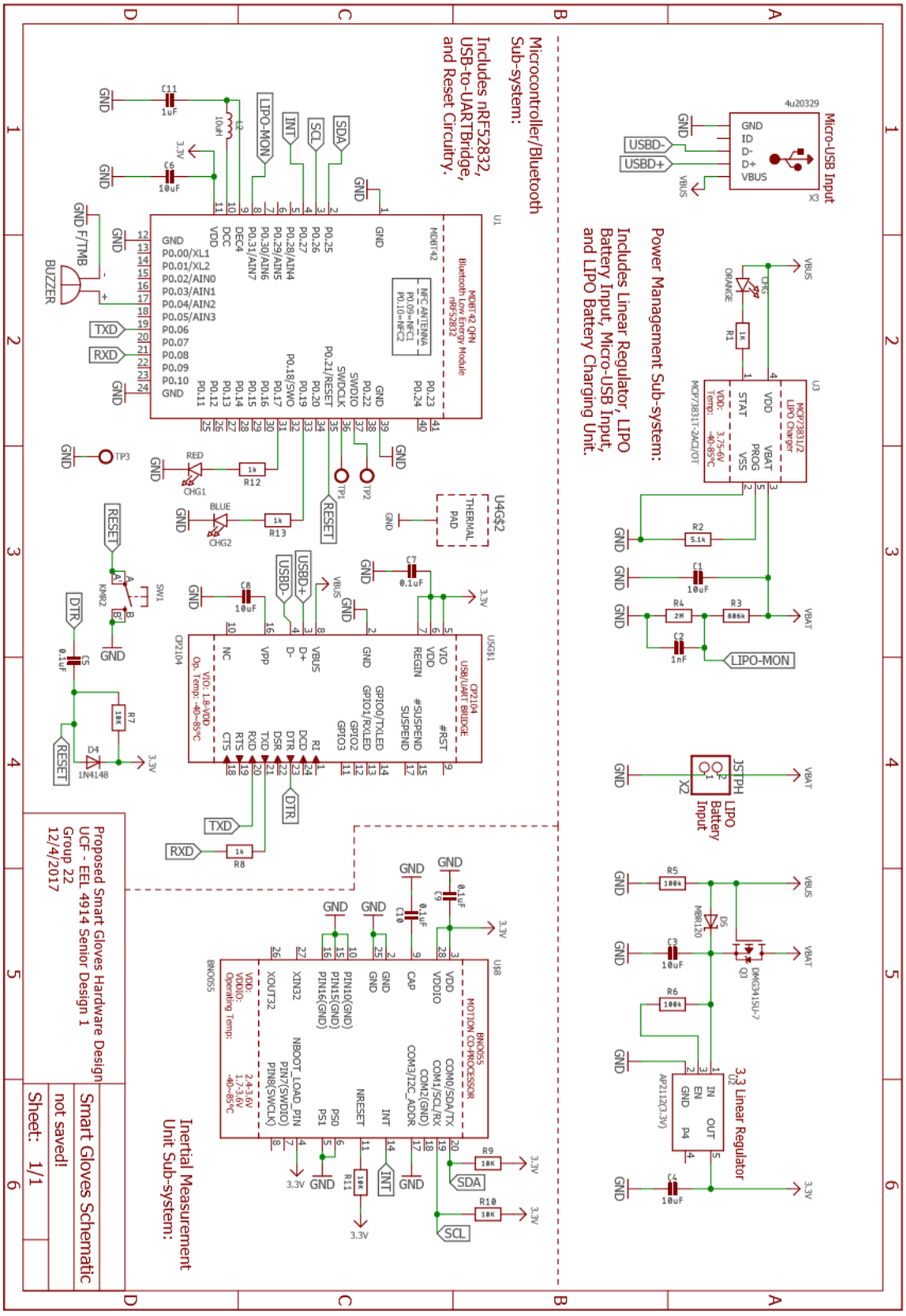


Figure 5.3: Smart Gloves Design Schematics

Tables 5.1 - 5.2 represent the full parts list for our Smart Gloves design, including all major components and secondary components like resistors and capacitors. All secondary components will be of the standard 0805 device footprint.

Table 5.1: Major Component List

Part Name	**Function**	**Price**	**Quantity**
BNO055	Inertial Measurement Unit	$12.07	2
MDBT42Q NRF52832	Microcontroller/ Bluetooth LE	$10.10	2
CP2104	USB-UART Bridge	$1.65	2
AP2112K	3.3 Linear Voltage Regulator	$0.48	2
MCP73831T	LIPO Charging Module	$0.59	2
DMG3415	P-channel MOSFET	$0.46	2
MBR120	Schottky Diode	$0.44	2
1N4148	General purpose diode	$0.14	2
PKCELL LP503562	LIPO Battery, 3.7V 1200mAh	$14.02	2
KMR231NG	Push-button Switch	$0.48	2
WM17143CT	Micro-USB Connector	$0.74	2
JST-PH	2-pin LIPO Connector	$0.75	2
ELEDIY	Piezo Buzzer for User Feedback	$1.10	2

Table 5.2: Secondary Component List

Part Name	**Price**	**Quantity**
1k Resistor	$0.10	4
5.1k Resistor	$0.10	1
10k Resistor	$0.10	4
100k Resistor	$0.10	2
806k Resistor	$0.10	1
2M Resistor	$0.10	1
0.1uF Capacitor	$0.10	4
1uF Capacitor	$0.11	1
10uF Capacitor	$0.12	5
1nF Capacitor	$0.10	1
10uH Inductor	$0.17	1
Orange LED	$0.31	1
Red LED	$0.31	1
Blue LED	$0.31	1

5.1.1 Power Sub-System

As explained previously, Smart Gloves electronics will draw power from two different sources, one being the 5V micro-USB power bus provided during connection with a powered micro-USB plug and the other being a LIPO battery. There are several major components that will be included in the power regulation circuit; they will all be listed below with a brief description of their function and major characteristics.

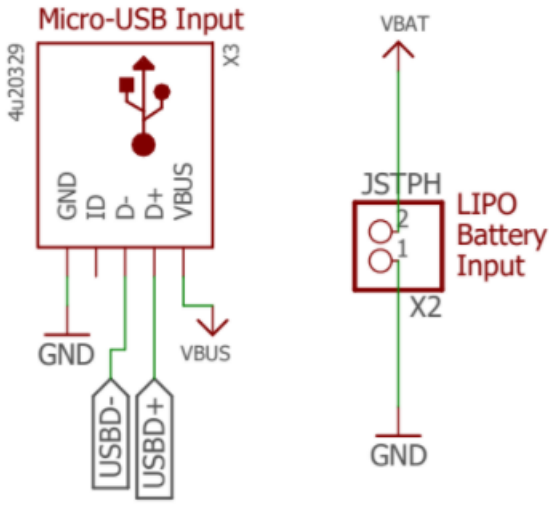


Figure 5.4: Power Source Inputs

Power Source Inputs - The Smart Gloves system design will support an input for each power source to be used, one for the micro-USB input and one for the LIPO battery input. As seen in Figure 5.4, the micro-USB input will connect to ground and provide a supply of 5V through VBUS. Additionally, the USBD+ and USBD- connections will serve to transmit and receive data via USB communications protocols and will connect to their respective inputs on the USB/UART bridge. This connection will be discussed further in Section 5.1.2. The LIPO battery will be connected via two pins, one establishing connection with ground and the other providing current and voltage to the VBAT terminal. Each of these connections will be sourced and ordered during construction of our final device prototype.

Linear Voltage Regulator, 3.3V - Because a constant DC voltage of 3.3 volts was chosen to operate all embedded hardware, we decided to implement a linear voltage regulator rated for a 3.3V output. This particular linear voltage regulator that we chose to incorporate in our design is a low dropout linear regulator, allowing a constant output voltage of 3.3V even when the supply voltage is close in value. With an input voltage range of 1.5V to 6V, this device can support the use of a wide range of LIPO batteries. Additionally, with a typical load regulation value of 0.2 %/A and a line regulation value of 0.02 %/V this linear regulator will reliably maintain between a 98.5% and 101.5% Vout of 3.3V. This particular regulator is a surface-mount device with 5 pins: one connects to the supply voltage, one provides 3.3V output voltage, one connects to ground, and another serves as an active high chip enabler. The 5th pin is not to be connected and serves no purpose. The exact model number of this regulator, along with manufacturer information, price, and package size, can be seen below.

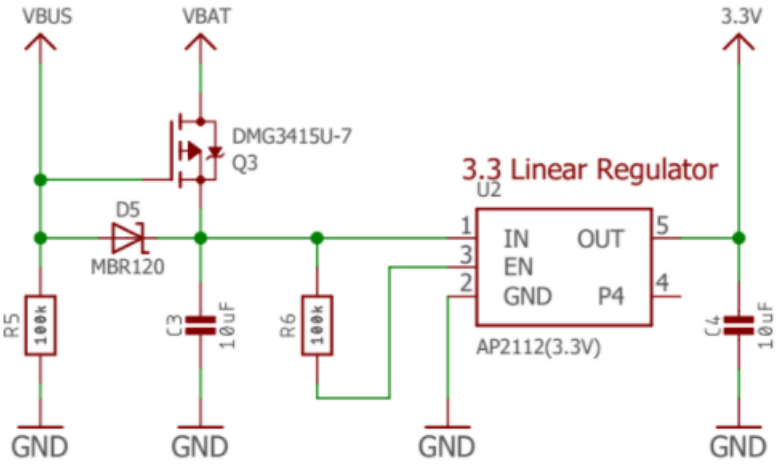


Figure 5.5: Linear Voltage Regulator

As seen in Figure 5.5, both VBUS and VBAT sources will be connected to the input pin on this linear regulator; however, both sources will be separated from each other via a PNP transistor and a Schottky diode that will prevent both sources from providing power to this pin simultaneously. The use of these components will be further discussed later in this section. Pin 3, which is the active high enable pin, will be connected to the input pin via a 100k resistor to ensure this regulator remains active while either power source is plugged in. The main 3.3V voltage supply terminal that will directly provide power to all processing components is connected to the regulator via pin 5, while pin 2 connects directly to ground. Pin 4 will remain unconnected as it serves no purpose for the device.

- Manufacturer: Diodes Incorporated
- Part Number: AP211K SOT25
- Price: $0.48 per unit
- Package Size: 3.00 x 2.80 x 1.15 mm

LIPO Charge Management Controller - The use of a LIPO battery in our power system design requires the employment of a module that controls the charging of the battery when a micro-USB connection is established. We chose to implement a single-cell LIPO charge management controller with a regulated output voltage of 4.2V and fully programmable charge currents ranging from 15 mA to 500 mA given the use of a single external resistor. This device will essentially pass current from the 5V micro-USB supply voltage through to the LIPO battery and, upon detection of a full battery charge, will automatically power down and cease passing current. To set the charge current value, the user places a resistor (RPROG) across the PROG input of the device to the 0V reference pin (VSS). The following equation

determines the value of the achieved charge current, where IREG is in terms of mA and RPROG is in terms of kOhms:

$$I_{REG} = \frac{1000V}{R_{PROG}}$$

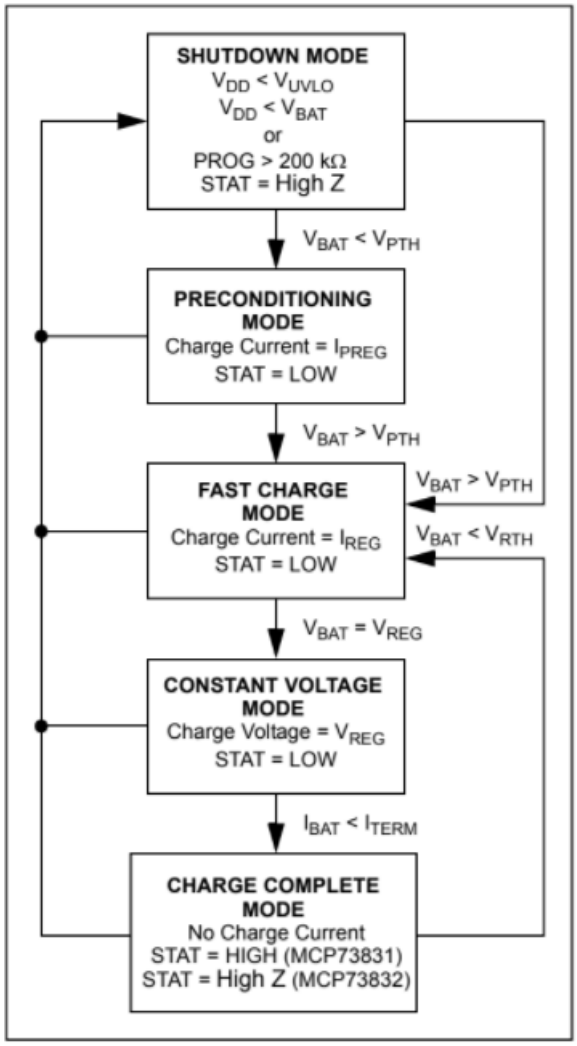


Figure 5.6: LIPO Charger Flowchart (reprinted with permission from Microchip)

An undervoltage lockout (UVLO) circuit constantly monitors the input voltage of this device and keeps the charger in shutdown mode until the input supply voltage rises to a level of 150 mV above battery voltage, resuming shutdown mode if the supply voltage falls to within 50 mV of the battery voltage. A 6 micro-A current sourced to the battery output charge pin allows for the detection of a battery; if a battery is detected and all UVLO conditions are met a preconditioning charge mode is activated. In this mode, a fraction of IREG is passed through to the battery until voltage

at the VBAT pin exceeds the preconditioning threshold. From there, the device enters fast charge constant-current mode, where the charge current supplied to the battery is equal to IREG, and upon VBAT reaching the regulation voltage value of 4.2V, the current is adjusted to maintain regulation voltage across the VBAT terminal. Upon detection of a drop in current, more specifically below a percentage of IREG, the charge cycle is terminated with a filter time of 1 millisecond to ensure premature termination. The voltage at the VBAT terminal is constantly monitored, and new charging cycles will begin if it drops below the recharge threshold [13].

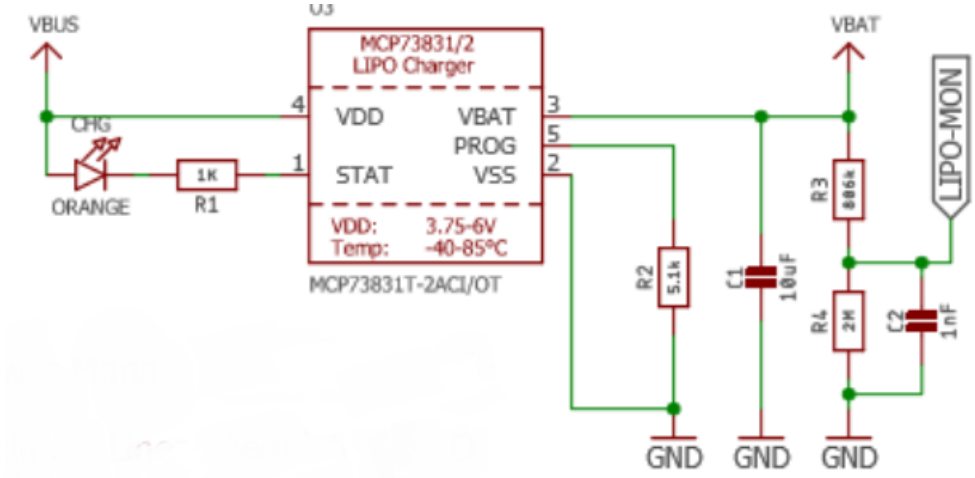


Figure 5.7: LIPO Charge Management Controller

There are five connections made to all 5 pins on this device, as shown in the schematics provided in Figure 5.3. The 5V VBUS terminal is connected directly to pin 4, which serves as the VDD input for this LIPO charger. The STAT pin, which is pin 1, is connected to VBUS via a 1k resistor and an LED: as the charger draws in current to charge the battery, this LED will be powered on to show charging status. Once fully charged, current will cease to pass through this terminal and the LED will be powered off. Pins 2 and 5 are connected via a 5.1k resistor, which serves to set a charging current of approximately 200mA to the VBAT terminal. Pin 3 connects to VBAT, the battery input terminal. Resistors R3, R4, and capacitor C2 serve as auxiliary circuitry to provide a reference point for the microcontroller to monitor the charge of the LIPO battery. The node between R3 and R4 will be connected to a general purpose analog input port on the microcontroller so that this charge can be monitored during operation of the device.

- Manufacturer: Microchip Technology Inc.
- Part Number: MCP73831T
- Price: $0.59 per unit
- Package Size: 2.90 x 2.80 x 1.45 mm

PNP MOSFET and Schottky Barrier Rectifier - Our Smart Gloves hardware design features two potential power sources for the device, one being the micro-USB 5V power bus and the other a rechargeable LIPO battery. These two sources can't be powering the device simultaneously however, and in order to ensure only one of these sources is powering the device at a time we will utilize a PNP MOSFET transistor to act as a switching device controlling the flow of current from the LIPO battery. As seen in Figure 5.5, the 5V micro-USB power supply is connected to the gate of the PNP transistor, while the LIPO battery is connected to the drain and the source is connected to the node that serves as the input voltage source for the 3.3V linear voltage regulator. When a powered micro-USB cable is plugged into the USB port on our Smart Gloves device, the 5V USB power bus will turn the PNP transistor on and prevent current from flowing through the drain to the source. When there is no device plugged into the micro-USB port, the PNP transistor will be turned off and current will flow from the LIPO battery through the device and into the supply voltage terminal of the 3.3V linear voltage regulator. With a maximum gate voltage rating of 8V, low on-resistance of 71 mOhms max, and a low turn-on delay of 71 ns, this device will function exactly as desired in our design.

- Manufacturer: Diodes Incorporated
- Part Number: DMG3415U
- Price: $0.46 per unit
- Package Size: 3.00 x 2.50 x 1.10 mm

To prevent the flow of current from the source of the PNP transistor into the gate, we will employ the use of a Schottky Barrier Rectifier, also known as a Schottky diode. This diode will be placed between the gate and source of the transistor, facing the direction of the input voltage terminal of the linear voltage regulator. With a peak DC blocking voltage value of 20V, this diode will function exactly as we expect it to in our final hardware implementation.

- Manufacturer: ON Semiconductor
- Part Number: MBR120ESF
- Price: $0.44 per unit
- Package Size: 3.80 x 1.80 x 0.96 mm

Power Calculations and Battery Selection - As per the requirement specifications discussed in Section 2.3 of this report, a key goal of our Smart Gloves design is for the glove-mounted electronics to have a battery life greater than 2 hours. This is crucial to the function of our eventual prototype, as a short battery life would get in the way of all necessary testing if the device was constantly in need of recharging. An extremely short battery life would also be detrimental towards any users of Smart Gloves: what would the use be in using this device if it constantly died in

the middle of workout sessions? Additionally, the battery will undoubtedly be the largest component implemented in our Smart Gloves design; however, we can't choose a battery too large or it could potentially interfere with a user's range of movement and serve as an annoyance. Battery capacity and size share a direct correlation, and as such a balance must be found in battery size and charge capacity.

It has been previously determined in this document that we will utilize a lithium-polymer battery in our design given their rechargeability and widespread use in integrated electronic designs. LIPO batteries are typically rated in terms of output voltage and charge capacity, which is classified as milli-Amps per hour (mAh). For example, a battery rated at 3000 mAh can supply 3000 mA of current for one hour on a full charge. The components of our design will be powered by 3.3V from a linear voltage regulator, so any battery that supplies more than 3.3V will be acceptable due to the regulator's low droupout capabilities. Given the mAh rating of a battery, the following equation can be used to calculate the life of a fully-charged LIPO battery:

$$Battery\,Life\,(hours) = \frac{Battery\,Capacity\,(mAh)}{Total\,Device\,Current\,Draw\,(mA)}$$

This calculation describes the battery life that would be obtained from operating a device at 100% efficiency; however, this is often not the case due to external factors that could potentially limit the efficiency of power consumption. To account for this, the value of battery life obtained in the equation above can be multiplied by a factor of 0.7 to account for these external factors and represent 70% efficiency. This is likely much lower than the efficiency that will be obtained in our final hardware implementation, but it is better to calculate/estimate the worst-case battery life scenario to avoid any last-minute surprises. Table 5.3 shows the absolute maximum current draw of all major components in our proposed Smart Gloves design that will draw non-negligible current when powered by the LIPO battery as opposed to the micro-USB power supply.

Table 5.3: Maximum Current Draw of Major Components

Component	**Maximum Current Draw (mA)**
Bosch BNO055	13.7
NRF52832 Microcontroller	13
USB-to-UART Bridge	17 .0

Adding the maximum current values up produces a total maximum current draw of 43.7 mA. To account for any other current draw that we might be unable to identify at this stage of the design process, let's assume that the Smart Gloves glove-mounted electronics draw a maximum 50 mA of current. With this value for current consumption in mind, Table 5.4 displays the expected battery life of several common LIPO battery capacities at 70% power consumption efficiency.

Table 5.4: Expected Batterly Life at 70% Efficiency

Battery Capacity (mAh)	**Battery Life (hours)**
500	7
1200	16.8
2000	28

As demonstrated in the table above, Smart Gloves electronics draw such little current that even a weaker battery capacity of 1200 mAh can theoretically provide approximately 16.8 hours of battery life at a worst-case estimation of 70% efficiency. This more than meets our requirement of ¿2 hours battery life and allows us to focus more on actual battery size as a limiting factor for battery selection. Having reviewed the expected battery life given different capacity values, two LIPO batteries currently under review are fit for use in our Smart Gloves project. The characteristics of these two batteries, both sold by Adafruit, are listed in Table 5.5.

It's worth noting that each battery's C rating is C5, which corresponds to the number of hours their respective power capacities can be spread across from a full charge. For example, in the case of Battery 1, 500 mAh of power is spread out over 5 hours with a maximum continuous discharge current of 100 mA. The standard continuous discharge current is only 20% of the maximum discharge current, which would correspond to longer battery life; however, our design would use approximately 40-50% of the max discharge current. By comparison, Battery 2 supports a maximum discharge current of 240 mA of which our design would be using an even smaller fraction of, leading to a much longer battery life. Taking this and all other characteristics displayed in Table 5.5 into account, Battery 2 would seem to be the best option for use in our Smart Gloves design due to its more than double power rating and battery life in comparison to Battery 1. Despite being more than twice as long as Battery 1, the length of Battery 2 is still short enough to fit on the back of a user's hand with the rest of our Smart Gloves electronics. The manufacturer, part number, and price of Battery 2 can be seen below.

Table 5.5: LIPO Battery Consideration Comparisons

Specification	Battery 1	Battery 2
Voltage Range	4.2V-3.7V	4.2V-3.7V
C5 Power Capacity Rating	500 mAh	1200 mAh
Max Charge Current	100 mA	240 mA
Standard Continuous Discharge Current	20 mA	120 mA
Max Discharge Current	100 mA	240 mA
Cycle Life	Greater than 500 cycles	Greater than 400 cycles
Battery Size	35.0 x 30.0 x 5.0 mm	62.2 x 35.5 x 5.2 mm

- Manufacturer: Hunan Sounddon New Energy Co.
- Model Number: 503562 1200mAh
- Price per Unit: $9.95

5.1.2 Microcontroller

This section will explain the various microcontroller interfaces we have attached in our hardware design and give a brief explanation as for reasoning and further planned development. Certain software implementation caveats and limitations of the nrf52's architecture will be mentioned as needed to aid in understanding our choice of implementation. In Figure 5.8 below we've made reference to the current microcontroller hardware schematics for your convenience when attempting to understand the schematics and various interfaces.

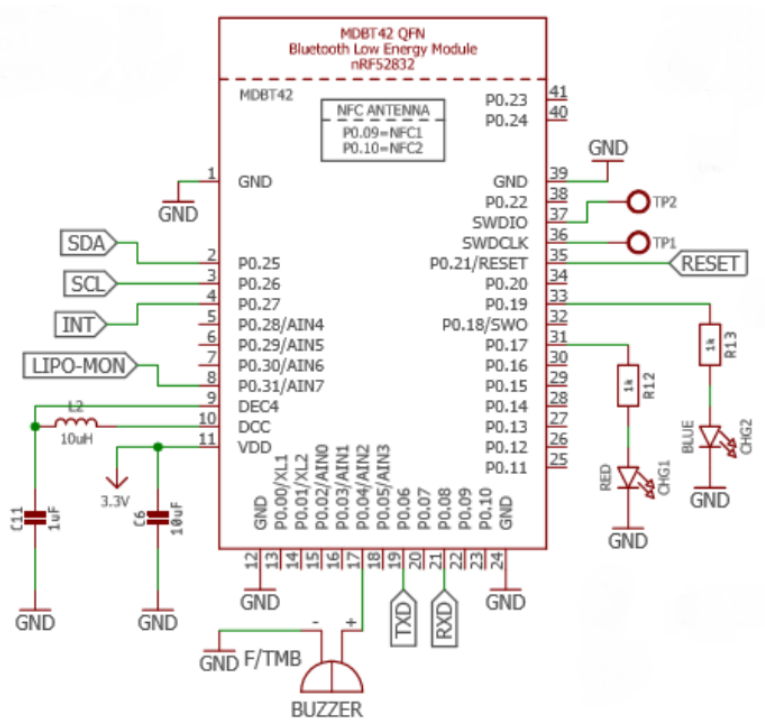


Figure 5.8: Microcontroller Layout with Pin-outs

IMU Interface: SDA, SCL and INT

The Microcontroller hardware layout has a few notable features and interfaces that are worth mentioning. First would be that of the I2C interface which utilizes the SDA and SCL line from the microcontroller which meet up with the BNO. We dedicated pins 25 and 26 for this case since a similar layout was utilized in the Adafruit breakout board. Doing so we can ensure that whatever legwork we do on the prototype can be mirrored in the actual product development, saving time. Another interface with the BNO from the microcontroller is the INT line. The INT connection on the BNO is basically a programmable interrupt we can utilize to send a notification from the BNO to the microcontroller. Unfortunately the BNO is not robust in its implementation of this and only simple situations can be accommodated but this will be discussed more in the BNO section.[21]

SWD interface: SWDIO and SWDCLK

Two other pins we are planning to utilize either on the back of the microcontroller as solder pads or through an SWD pitch connector are the SWDIO and SWDCLK connections. These connections are utilized as a simplified version of JTAG that only requires a 3 pin connection: Ground, SWDIO and SWDCLK. This interface will

allow us to flash the bootloader on to the microcontroller by simply soldering wires to these connections. The SWDCLK and SWDIO lines both have internal pull up resistors on the microcontroller so it's not necessary for us to implement this on our side. It's worth noting here that flashing the microcontroller will require additional hardware in order to connect this interface to a computer and flash the bootloader in the form of a mini stlink and accompanied the nrfjprog software environment that will let us push our developed code to the microcontroller.[21]

RESET

To the RESET pin on 28 we have it lined up with the RESET node. On this two we have two interfaces for resetting the device. One is a push button that will reset the device when pushed down and the other is connected to the DTR connection on the UART to USB bridge. This connection will be further expanded upon later but for now just know that the DTR pin on the UART to USB bridge has the ability to go to low which would cause the RESET node to go to low, resetting the microcontroller program to the start. Reasoning for this and specific interactions with the UART to USB connections will be discussed later.[21]

Battery Charge Monitoring: LIPO-MON

On pin 31 we linked the microcontroller up with LIPO-MON node. This connection basically allows for us to monitor power levels from the battery by reading the voltage divided across a resistor. To do this we utilize the microcontroller's ADC or Analog to digital converter and pin 31 on the microcontroller is one of the 8 ADC capable pins on the microcontroller. The ADC on the nrf52 is a SAADC or Successive approximation analog-to-digital converter.

This SAADC allows for 8, 10, 12 bit resolution and 14-bit resolution with oversampling. We can read full scale input range of 0 to VDD voltage and sampling can be triggered as a interrupt from timer connections. A 12 bit clarity would mean that there would be 4096 steps from us to read the voltage on the SAADC. This would mean that voltage level would be able to be read from increments of 0.00073 volts from 0-VDD where VDD is assumed to be 3V as per voltage supply from the voltage regulator.[21]

Table 5.6: Properties of the nrf52 Analog to Digital Converter

nrf52 Analog to Digital converter	Value
Input Range	0 - VDD (VDD 3̄V)
Expected battery read range	2.64V(low charge) - 3V(full charge)
Resolution	8/10/12-bit resolution 14-bit resolution with oversampling
Voltage clarity at 12-bit resolution	VDD(2$\hat{8}$) = 0.00073V increments

We also have the ability to utilize up to 8 input channels on the SAADC, however this would more than compensate for what would require of it. Support is also given for direct transfer from the SAADC to RAM through EasyDMA (Direct Memory Access) this would free up processor time for other activities by simply allowing the voltage sample to be written at regular intervals to RAM. Sample acquisition time utilizing the 806K Ohm resistor in series with VBAT will be about 40us to charge up the ADC's internal capacitor. This is not problematic since we expect our sample rate to be much less than this. Even a 1Hz sampling rate would be fine for our purposes since battery level fluctuation will happen over long periods of time.[21]

USB/UART bridge

For the connections on the microcontroller to the USB to UART bridge we have two connections hooked up. The TXD node is for asynchronous data output from the microcontroller to the USB to UART bridge. The RXD node is the Asynchronous data input line to the microcontroller from the USB to UART bridge. These two pins, 19 and 21, are configured software side to allow for a continuous UART connection with a computer through a USB interface. As a side note this connection will only be available for full programmability once we initially push the bootloader onto the microcontroller through the SWD interface as described above.

Piezo Buzzer: PWM

For connections with the piezo buzzer we are going to utilize PWM or pulse width modulation to control the frequency received. This modulation would allow for specific tonal sounds to be created as the frequency of the pulse width modulation

is adjusted, enabling us to create simple tonal sounds. The PWM unit provided by the nrf52 provides three separate modules with individual frequency control in groups of up to four channels. Each module has a fixed PWM base frequency with a programmable clock divider.

LED Indicators

Status indicators for battery charge will also be connected to the microcontroller on one of the GP/IO pins along with a resistor in series. This will allow us to relay to the user the current power levels of the battery as we've read from the LIPO-MON node mentioned above. This implementation of this would be as simple as setting the LED to high when at full battery and toggling the LED when the battery drops below a certain threshold, say 30%. Another status LED will be connected to determine Bluetooth statuses such as searching, and connected to display to the user what mode the Bluetooth connection is in.

5.1.3 USB-UART Bridge

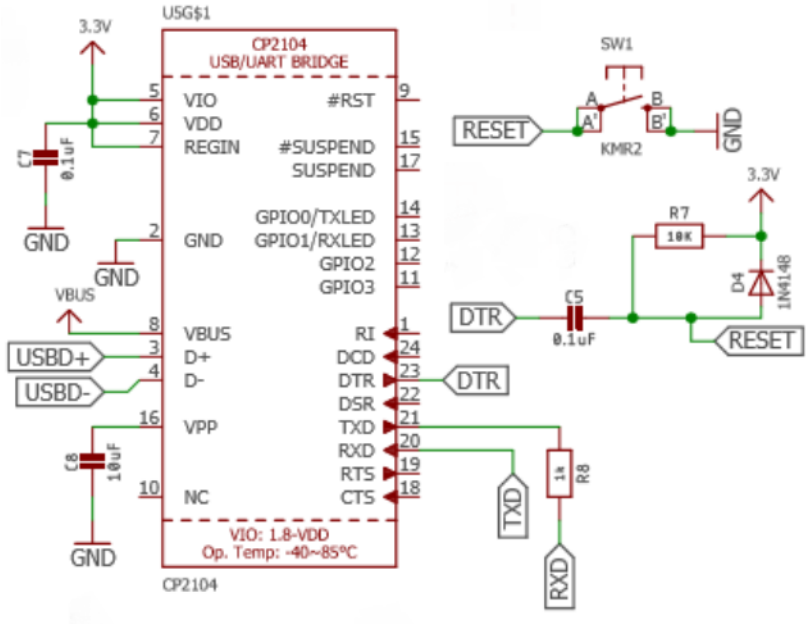


Figure 5.9: USB/UART Bridge with DTR reset connection

The USB to UART bridge is going to be our main connection from the microcontroller to the computer. This interface is what is going to allow us to push software from the computer to the microcontroller once the bootloader is initialized. This

connection will also allow back and forth communication so the microcontroller can send information to the computer which can then be viewed in a serial monitor. This section will outline some of the major connections and interfaces that we are going to be utilizing. The major or obvious connections will be excluded from discussion.

USB+ and USB-

These nodes are connected to the USB/UART bridge and provide data transfer from USB to the UART bridge. The USBD+ and USBD- connections are USB 2.0 full speed function controller enabled. The UART interface supports a variety of data formats and baud rates that are abstracted from the user when interfacing with the device through virtual COM Ports.[45]

RXD and TXD

These nodes are connected to their corresponding pairs on the USB to UART bridge. This RXD and TXD nodes are connected to the TXD and RXD pins respectively on the bridge. These are essentially asynchronous transmission lines connected from the the microcontroller to the bridge with RXD being the receiving side and TXD being the transmitting side. A 1 kOhm resistor is placed in series with the TXD pin to protect the microcontroller from high current values as there is no pullup resistor the nrf52.[45]

DTR

The DTR node is connected to the DTR or Data Terminal Ready pin on the USB to UART bridge. This pin is active low and whenever the computer wants to talk with the microcontroller it will bring this pin to the active low position. We've interfaced this pin with the RESET node buffered by a capacitor meaning whenever this is initially set to low, the RESET node will briefly be connected to ground while the capacitor discharges. This will cause the microcontroller to reset, setting the program counter its initial location which in our case would be the bootloader once flashed. The bootloader is programmed to query the UART connection upon startup which allows this communication to be performed and code to be uploaded. If a user chooses to not program the microcontroller they can still set DTR to utilize the serial monitor to inspect code for proper functionality and debugging purposes.[45]

5.1.4 Inertial Measurement Unit

This section will explain the hardware implementation of our IMU or inertial measurement unit. The BNO055 is going to be our IMU and we are interfacing it with the microcontroller to relay the accelerometer, gyroscopic and magnetometer data to then transfer this data over bluetooth to the phone environment. Calculations and comparisons will then be performed to ensure the exercise is being performed correctly.

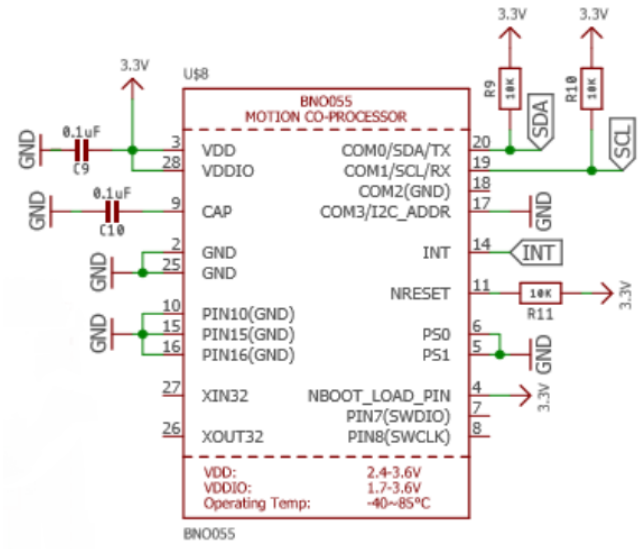


Figure 5.10: BNO055 PCB layout

Microcontroller Interface: SDA, SCL and INT

The SDA and SCL lines, already mentioned in the microcontroller section, facilitate the I2C communication. The SCL line is the clock signal generated by the microcontroller and the SDA line for data signal. SDA is basically a two way data communication line that either side can pull low to initiate transfer. We will be using this line to send IMU data to the microcontroller and manage mode configurations. Both communication lines are connected to the 3.3V power source via 10k pull-up resistors. [19]

The INT pin is configured for interrupt signals to the host or in this case the microcontroller. We can configure several types of interrupts such as slow motion, any motion, or a accelerometer high g. All of these interrupts are fairly customizable and will raise the interrupt pin to high until reset by the controller. As for our purposes we are planning to utilize the INT pin for high acceleration since this is a good indicator that you are performing an exercise improperly. [19]

5.2 Designing and 3D Printing Sensor Case

In order to keep a clean design and ease of use, a case is to be considered to be design for the sensor. In order to keep a custom-made design, the case is to be designed and printed with 3D printing technology. A good 3D print requires a well-built model. It is important to keep in mind that while desiging a print with 3D modeling software, neatness matters. Understanding which geometric shapes go well together will go a long way with creating a design that will transform into a well-built model. Though building a case for the printed circuit board should be a simple task, getting the sizing right may come from multiple trials of modeling. This project is focused on designing and implementing electrical and software components. However, knowledge in designing 3D printed models can be helpful in this project and future projects that require cases for electrical components.

When designing a model, there are material and size limitations. There are some modeling programs that do not allow to build in a particular unit. Though size can be expanded indefinitely in software, it is important to keep in mind that there may be some designs that require wall thickness. These units become relevant when dealing with small units of measurement. In the case of this project size and small details will matter as we are trying to keep the housing as small and least intrusive as possible. The innovation lab at the University of Central Florida will be used to 3D print the parts needed. Utilizing this lab will keep costs at a minimum as those resources can be used free of cost.

5.3 Software Design

The success of this project relies heavily on great design and the utilization of design patterns. From a software standpoint. Write clean and well-organized code is significant. It can help reduce debugging time tremendously as well as making it easier to add new features to the current code base. From a user interface standpoint, Apple provides guidelines in their Human Interface Guidelines documentation to provide details and tips on creating a great customer experience. Though the scope of this project is only for this senior design project, practicing and keeping these UI design patterns in mind will help in future project and perhaps even in our careers.

In its documentation, Apple touches on design principles to maximize the impact and reach of an application. In total, Apple mentions that there are six things to keep in mind when developing and design a user interface. The first one is aesthetic integrity. The importance of this is to ensure that an app's appearance and behavior integrate with its function. When capturing or displaying serious or important information, keeping the user focused is important. This can be done by using

subtle, unobtrusive graphics, standard controls, and predictable behaviors. In the companion app for the Smart Gloves, we plan to display important and potentially sensitive data relating to health. Keeping this design principle is very important. It is also important to remain consistent in the way that data is displayed by using system provided interface elements, common easy to recognize icons, and standard text fonts and styles. Within consistency it is important to display features and behaviors that the user expects. Throughout the design of the UI, it will also be important to keep in mind that people, not robots, are in control. This means that the app must suggest a course of action or warn about dangerous consequence of potential interactions. The goal for the companion app is to keep the user engaged and interested to continue to use both the hardware and the software products.[46]

In industry, many people like to talk about the different design patterns that they use and how they are helpful in different situations. It is inarguable that design patterns makes coding safer and easier. Software design patterns represent the best practices used by experienced object-oriented software developers. They also provide solutions to general problems that are faced during software development. Since the companion app will contain a user interface and backend code, the Model-View-Controller (MVC) design pattern will be used. This design pattern is used to separate the concerns of the application. The model represents an object that contains data. It can also contain the logic to update the controller if data is changed. The view represents the visualization of data that the model contains. Finally, the controller acts on both the model and view. It serves as the middleware between the two. Controlling the data flow into model objects and updating the view whenever data is changed. In the case of the smart gloves, the view will be the user interface, the model can be the data that is to be retrieved from the smart glove sensor or the user's information such as their name, weight, height, and age. The controller will be the link between the two and contain the logic necessary to provide the information the user must see. The figure below display Xcode's Interface Builder. The figure highlights sections of Interface Builder that give the ability to design user interfaces in Xcode. When using the Model-View-Controller design pattern, this is the section of Xcode where the view will be designed Displayed is also a simple UI with five buttons and a title. These components can later be attached or configured to code so give them use and purpose.

There are two options to choose from when deciding on which platform to develop a mobile application that is available to the masses. While Android holds a bigger market share globally, in the United States the split is down the middle between Android and iOS. There were internal debates within the team as to which platform we should code the companion application. For the moment, we have chosen to write the app on iOS but options are being kept open to potentially also create an Android if more appropriate.

5.3.1 Xcode

In order to develop a successful application on iOS, in particularly an iPhone, one must become familiar with the tools that Apple provided. The Integrated Development Environment (IDE) used for development in this ecosystem is Xcode. Xcode provides plenty of tools to develop, test, and deploy beautiful, easy to use applications that can help market and bring in multitudes of money. Xcode allows to keep code organized while also provides an intuitive experience for new or seasoned developers. Xcode integrates to version control systems like Git and makes it easier to maintain a history of the code base by using GitHub. There have been instances in other projects where code has been lost or sharing code was a challenge due to not using version control. Not only does it provide peace of mind that the code that is being developed is safe but having the ability to revert changes can be crucial in the instance that issues or bugs arise. Xcode also provides a graphical debugger that helps with troubleshooting bugs. It has the ability to connect to continuous integration systems that allow for frequently testing additions to software.

Perhaps one of the most useful tools in application development in Xcode are the vast majority of iOS simulators it provides. In the use case of this project, we will only be focusing on developing apps for iPhones. Xcode provides multiple simulators for different iPhone devices and their different screen sizes. This provides to get a clearer picture of what the app will look like in a real device and also make testing much quicker. As expected, it also provides the ability to deploy code to a physical device to accomplish the same tasks mentioned before. Xcode also provides an useful tool called Interface Builder. With Interface Builder, user interfaces can be created to fit the needs of the application. It is a way to arrange and organize UI components and linked them to variables and objects. Since there are different iPhones with different display sizes, the use of Autolayout in Interface Builder make it easier to code for all the different screen sizes.

Also, interface builder contains storyboards. Storyboards show a complete view of the application's flow. There are also a large number of storyboard controllers that provide multiple capabilities in UI design. These controllers include table view, collection view, navigation view, tab bar controllers, and many more. At the moment that UI for the application has not be determined but the use of these tools will be essential once that point is reached. It is important to not that Xcode will only run on Apple computers running Mac OS. This is an important limitation to consider and not every developer uses a Mac computer.

In order to design and develop iOS applications, a Mac with Xcode is required. Along with Xcode, one must be familiar with Objective-C and Swift. Swift is a recent addition to that technology stack. Since the release of the iPhone and perhaps years before that, Objective-C was the main programming language used to develop iOS application. Apple and its developers, however, started to realize that Objective-C was starting to show its age. So Apple developed and released Swift

in 2014. Since its released, it has become a trending topic among the tech community. Developers embrace its general purpose, modern design to develop iOS applications.

As mentioned on Apple's documentation on Swift, Swift is a powerful and intuitive programming language for macOS, iOS, watchOS, and tvOS. Writing Swift code is interactive and fun, the syntax is concise yet expressive, and Swift includes modern features developers love. Swift code is safe by design, yet also produced software that runs lightning-fast.[47] Perhaps, much of the previously mentioned statement is a great marketing tactic by Apple, but the tech community and reviews support these statements. Swift has quickly become a popular programming language that even kids have started to learn at a young age. It has been started to replace Java in the enterprise world for back end systems and programming. Apple has seen the potential of Swift that they have open sourced it in order to take it to another level.

Swift is now on its fourth iteration with more features added. For this project, the companion app will be coded in Swift. Doing so will allow the use of modern libraries and learn skills that can be applied in industry. So far, it has been easy to learn and get started with Swift through Playgrounds. Apple has developed Playgrounds to make it easy and quick to get started coding in Swift. It allows to get writing code faster, providing output without needing to wait for compile and runtime. Apple provides libraries for that can be coded in Swift in order to interact with bluetooth devices. This is a huge feature as this is how we plan to communicate with the printed circuit board. Much more knowledge is needed to sample and collect data efficiently to display accurate, precise, and useful data to the user.

Swift supports basic operations like most programming languages. These operations are addition, subtraction, division, multiplication, and modulo. Variables are assigned with the keyword var while constants are assigned with the let keyword. Swift also contains Unary and Ternary operators to make code cleaner and easier to write. Perhaps one of the most useful features of Swift is tuples. Though they exist in other programming languages, tuples in Swift make code be a lot more versatile and easier to read. Unlike Java, tuples allow to return multiple values instead of just one.

Swift also allows for useful string manipulation with its built in methods. To make it easier to strings and other data types such as numbers can be concatenated together with string concatenation. While making it easier to print strings and other data types with string interpolation. For collection types, Swift offers arrays, sets, and dictionaries. Each which are helpful and have their own syntax. These are standard data structures among programming languages. Perhaps what makes the better in Swift is their syntax. There is a lot less typing needed to do to declare an array or a map compared to Java.

Code in Swift can be organized in classes or structs. Each with its own unique features and limitations. Structs are a great way to define objects that don't change very much, while classes are good for objects whose member variables may change constantly. There are also other features like structs have built in constructors while classes don't. But for the purpose of explaining and providing a bit of information of Swift, these details will be omitted. Swift also provides enumerations, protocols, methods and functions, generics, type casting, error handling, among many other features. But perhaps a modern feature called optionals.

As described in Apple's documentation, optionals are useful in situations where a value may be absent. An optional represents two possibilities. Either there is a value and it can be unwrapped to access the value, or there isn't a value at all. It is important to note that the concept of optionals do not exist in C or Objective-C. The closest thing that Objective-C provides is being able to return nil from a method that would otherwise return an object. Apple provides an example of how optionals are useful while developing iOS applications. Swift's int type has an initializer that tries to convert a string value into an int value. But note every string can be converted into an integer. In the case that the initializer fails, an optional int is returned. In this case the optional would be empty but it can be safely unwrapped and does not lead to errors in code. Optionals are written with a question mark at the end of the data type. The question mark indicates that the value it contains is optional, meaning that it might contain some value or it might not contain a value at all.

As described in recent paragraphs Swift is a versatile, modern programming language that makes iOS development a lot easier than when using Objective-C. Like mentioned before, Swift will be used to develop the companion app for the smart gloves sensor. With the help of Xcode, Swift should provide all the tools needed to communicate, gather, and display data from the smart glove sensor.

With Android holding 86.2 percent of the worldwide smartphone operating system market in 2Q16, the Smart Glove team made the decision to develop for Android. Developing for Android would be the best use of our time and resources as it captures the largest portion of the market. [48]

Java is the official language of Android development and is supported by Android Studio. With that in mind, it is naturally the language of choice for our Android based application. Java is an Object Oriented Programming (OOP) language with all of the pros and cons.

Object Oriented Programming is a programming language model organized around objects rather than a logical procedure that takes input data, processes it, and produces output data. In all Object Oriented Programming languages, a class defines an object. An object is a self contained item that interacts with other objects.

5.3.2 iOS Bluetooth Libraries

Bluetooth is a core technology to be used in the communication of the Smart Gloves sensor and the companion iOS application. Apple provides a Core Bluetooth framework with classes that assist in the development of iOS and Mac apps. The classes in the Core Bluetooth framework provides the ability to communicate with devices that are equipped with Bluetooth low energy technology. As stated on Apple's documentation, in Bluetooth low energy communication there are two key players, the central and the peripheral. A peripheral typically has data that is needed by other devices. A central typically uses the information served up by a peripheral to accomplish some task. In this project, the peripheral will be the Smart Gloves sensor and the central will be the companion iOS application.[49]

The central and the peripherals are considered a player in the system. They each perform a different set of tasks within their role. Apple mentions that peripherals make their presence known by advertising the data it contains over the air. Centrals scan for nearby peripherals that might have data they're interested in. Once that connection is made, the central will then begin to interact with the peripheral and exchange the data desired. In this case, the peripheral is responsible of responding to the commands the central sends. It is also important to note that while the iOS app is in the background or in a suspended state, its Bluetooth related capabilities are affected. Apple has designed their devices so that by default, applications are unable to perform Bluetooth low energy tasks on the background or in a suspended state. This is important to note as the app may enter one of these states while the user is exercising. However, this default can be changed and allow the app to perform Bluetooth low energy tasks in the background. The tasks that are allowed differ from app state. As more documentation and design constraints are explored, it is important to keep note that even apps that support background processing may be terminated by the system in order to free up memory and resources.

Apple provides more in-depth information how to implement the central and peripheral roles. It goes on to provide code snippets to get started with receiving information from a peripheral as well as best practices on manipulating this data. Apple's documentation will be used in the implementation of the companion iOS application.

5.3.3 Adafruit BNO005 Software Library

Adrafruit provides drivers for the BNO055 to retrieve relevant information from the sensor. According to Adadruit documentation on Github, the interface defines some basic information about the sensor and returns SI units of specific type and scale for each supported sensor type.[50] Highlighted in the documentation is the simplicity and usability of the library, giving the ability to gather significant amount

of information from just a few lines of code. It also highlights how the data can be easily and quickly used since it is already converted to SI units that is easy to understand and compare. With the standardization of SI units, it is also so compare results with benchmarks or other sensors. This library will make the development with the BNO055 sensor easier while also allowing easy learning from the BNO055 breakout board. The library consists of header files written in C with built in functions that make gathering data easy.

5.3.4 iOS Companion App

The companion application will contain code that will make use of the data collected by the BNO055 IMU. This data can be meaningless to users as the output data is somewhat cryptic. The iOS application will take this cryptic like data and make it readable to the user and also help provide warnings if the user is exercising in a way that can be harmful to the their body. With that being said, organizing software is important for readability and maintainability. The screenshot below displays a sample of the types of files and hierarchy that will be used to organize the software for this application. There will be files with the swift extension that will serve as controller code for the view portion of the application. These files can contain Swift classes, structs, variables, functions, and methods. Swift files that contain Controller as a suffix contains code that interacts with the view and model. As discussed before controllers act as a middle man and contains logic to make sense of data models can process efficiently and display human readable information.

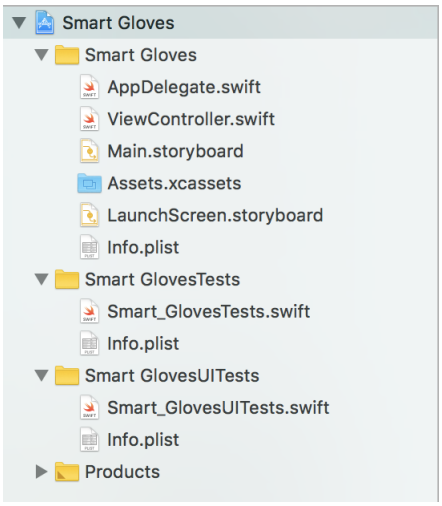


Figure 5.11: XCode Project File Hierarchy

The sample picture above shows the different project files that make up a sample iOS application. Under the Smart Gloves directory, the project files that make up the UI and backend code are stored here. There are also two other directories

that that contains tests. One of these tests is solely for UI testing while the other is for the backend code. Plist files are used to store user settings. These user settings can be used in tests to pass in data needed. The same can be done for project files. As noticed from the picture, the files containing code will have the swift extension. This means that the code will be written in the Swift programming language.

Another important file in the iOS and Xcode file hierarchy is the storyboard file. In this case, main.storyboard will contain the UI components needed for the app. The screenshot below displays what the main storyboard looks like without any UI components. This is the canvas that will be used to add the necessary UI components and like that to the code in the controller classes. The screenshot also displays a View Controller Scene. This is where all the UI components are shown in a listview. Having the ability to see all UI components in a list allows to quickly select, edit, and delete components. As the screen starts to get cluttered and contain multiple components, using the View Controller Scene will become very useful.

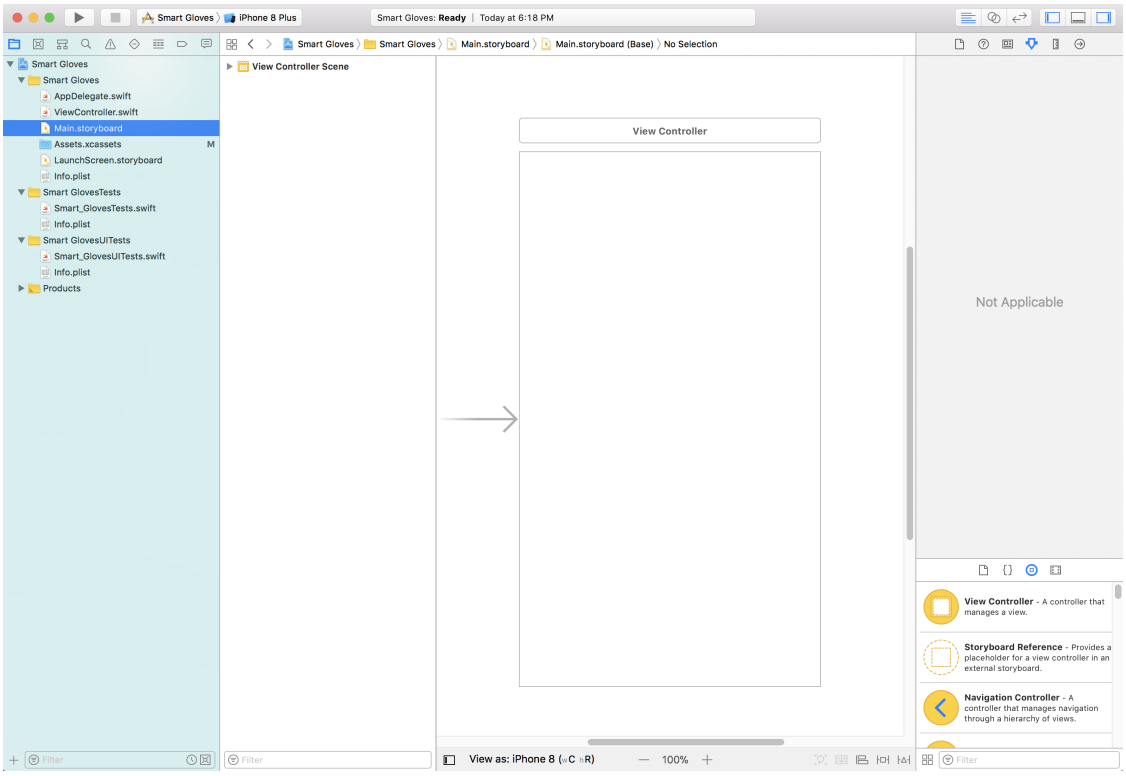


Figure 5.12: Main Storyboard View

With the help of the assistant editor in Xcode, the controller code for UI components can be placed size by size with the UI components to have both in a single view. This allows to neatly arrange code and link components to the backend driving

code. The screenshot below displays this setup. As components are added to the storyboard, those components can be added to the ViewController.swift file. Member variables can be assigned. In this controller Swift file, methods and functions can be written to create actions when a button is pressed or change the text of a text field. Having the ability to directly link UI components to variables and methods is extremely powerful and provide clarity to ensure that the correct interactions and connections are being made.

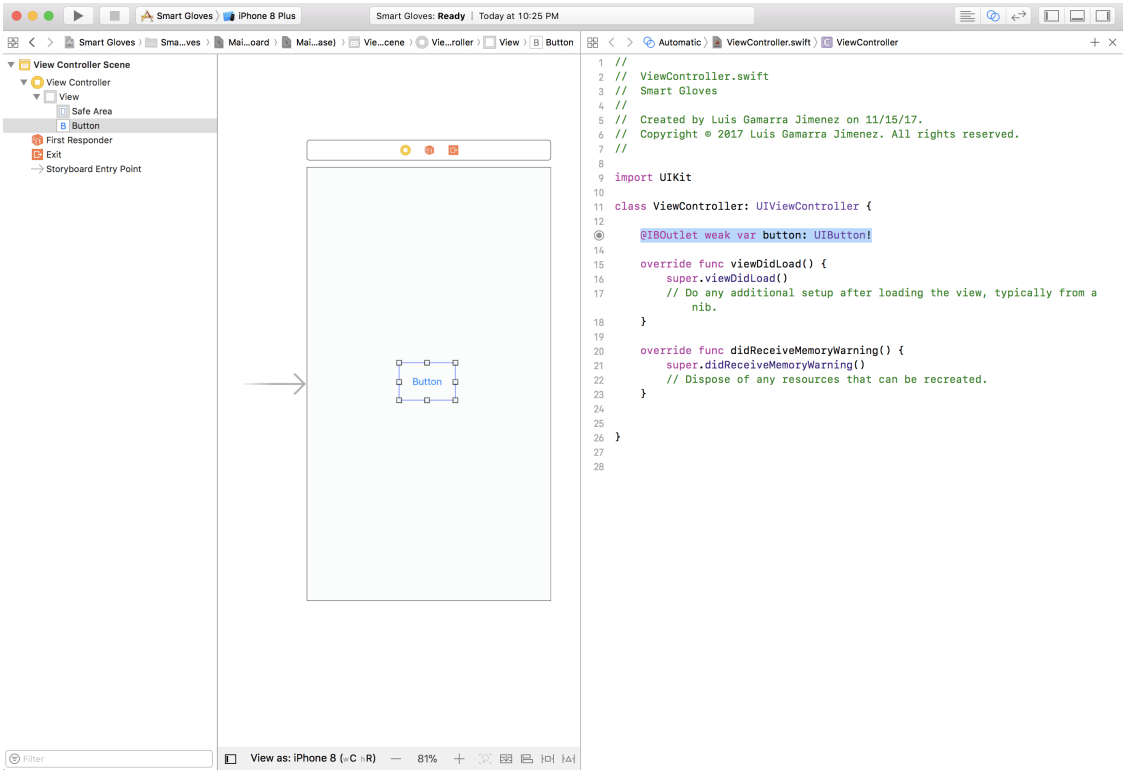


Figure 5.13: Main Storyboard with Backing View Controller

Moving on to a plist file. As mentioned before plist store user settings that can be used in application code or tests. Plists are also key-value pairs. In the world of programming, examples key-value pair data structures are dictionaries and hash maps. Dictionaries can be embedded within each other as commonly seen in JSON files. The screenshot to the left shows a plist file with project settings.

Key	Type	Value
▼ Information Property List	Dictionary	(14 items)
Localization native development region	String	$(DEVELOPMENT_LANGUAGE)
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
▼ Required device capabilities	Array	(1 item)
Item 0	String	armv7
▼ Supported interface orientations	Array	(3 items)
Item 0	String	Portrait (bottom home button)
Item 1	String	Landscape (left home button)
Item 2	String	Landscape (right home button)
▼ Supported interface orientations (iPad)	Array	(4 items)
Item 0	String	Portrait (bottom home button)
Item 1	String	Portrait (top home button)
Item 2	String	Landscape (left home button)
Item 3	String	Landscape (right home button)

Figure 5.14: XCode Plist File

5.3.5 iOS Companion App Design

In order to facilitate the design of the application and brainstorming of controllers needed, UML diagrams are created. UML stands for Unified Modeling Language. UML is a way to visualize a software program using a collection of diagrams. UML diagrams were designed to be used for object-oriented designs but has over time be developed to use with a wider variety of software engineering projects. The purpose of making UML diagrams is to connect shapes that represent an object or class with other shapes to illustrate relationships and the flow of information and data.

There are different types of UML diagrams. The variety of diagrams can be categorized under two distinct groups; structural and behavioral diagrams. Structural diagrams contain diagrams that define the individual components of the application. These include class, package, object, and component diagrams. As the name suggest, the make up the main structure of the application as without these the application does not have state or behavior. Behavioral diagrams inform about activity, sequence, state, and use case.

As mentioned in Smart Draws documentation on UML diagrams, class diagrams are the backbone of almost every object-oriented method. They describe the static structure of a system. These diagrams will be mainly used in this project to describe the classes, class members, and methods that will be used. Along with class diagrams, state diagrams will also be used to describe the sequence and behavior of the application. State diagrams are useful to model reactive objects

whose states are triggered by specific events.[51] In the case of this project, certain events will be triggered by interrupts or certain events in which the user may be exercising the wrong way or the user begins a workout. In this case there will be certain steps that will be need to be executed to ensure that the proper methods get initialized and the expected outcome is provided. A valid question to ask would be, why go through the trouble of creating a diagram that can be very cumbersome to create. But in reality creating UML diagrams can have huge benefits. As mentioned in Smart Draws documentation about UML diagrams, visualizing user interactions, processes, and the structure of the system being built will help save time down the line and make sure everyone on the team is on the same page. I couldn't agree more with this statement. There is a big time constraint both for this paper and implementation of the Smart Gloves. Having a diagram that will help steer back into the right direction will be helpful in times of doubt and question throughout the implementation process. However, at this point of the design process, it may be a bit premature to call a class diagram as finalized. Due to the nature of the project for which learning is done in real-time, documentation and design decisions are bound to change. For the time being a UML diagram will serve as a good brainstorming tool to help identify strengths and opportunities in the design choices.

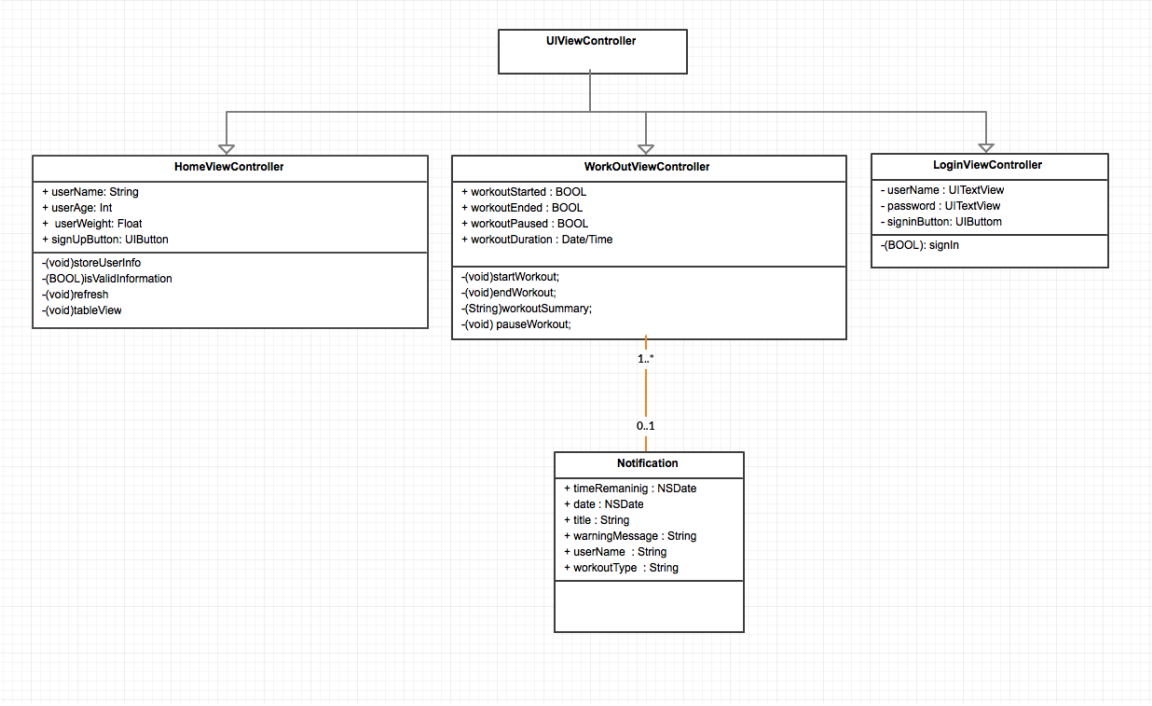


Figure 5.15: Class Diagram of iOS Companion App

The class diagram above provides description for the classes and class members to be used in the view controllers for the iOS companion application for the Smart

Gloves. This class diagram is a work in progress as it is early on in the design process to fully decide what the classes will contain.

5.3.6 Android Studio

Android Studio is a piece of software called an Integrated Development Environment (IDE). Along with Android Studio comes the Android Software Development Kit (SDK). The Android SDK is a set of tools to facilitate Android development. This will give us everything that we need to get an Android application up and running.

Android Studio allows allows a developer to gain access to the unique features of the Android operating system. Android Studio also allows a developer to use an emulator to test apps that are built, monitor a device, and do a host of other things.

The Android SDK typically comes bundled with Android Studio. The Android SDK can be broken down into several components: Platform-tools, build-tools, SDk-tools, the Android Debug Bridge (ADB), and the Android emulator. To start writing Java programs, we need a way to compile source code and turn it into an executable for the Java runtime. In order to do this, we can utilize the Java Development Kit (JDK).

5.3.7 History of Java

Java was released by Sun Microsystems back in 1995 and is used for a wide range of programming applications. Java code is run by a virtual machine, which runs on Android devices and interprets the code.

It was designed to be an easy to learn language for programmers who already knew C and C++. During 2006 and 2007, Sun Microsystems released Java as a free and open-source software under the terms of the GNU General Public License (GPL).

Java is architecture-independent which means that a .java file isn't compiled for a specific processor on a specific OS, like Windows on an Intel x86 chip, or Android on an ARM Cortex-A processor, but rather it is turned into Java bytecode. Java bytecode is the instruction set of the Java virtual machine (JVM). The job of the Java virtual machine is to run that byte code on the specific platform.

5.3.8 Embedded Systems Application

Mobile Application Design

Upon initially opening the mobile application, the user will be met with a splash screen and options for logging in or creating an account. The first time the application is opened, the user will be guided through a tutorial on how to set up and use the device. At this point, the various sensors on the Smart Gloves will be tested for connectivity and communication. The user will be walked through various steps on how to pair the device with a smart phone. The user will also be guided on how to attach the sensors to their body and set them up for calibration. Pictures will show the user what the various ports, buttons, and lights on the device signify.

On the main screen of the application, there will be a variety of tabs. The tabs will include things such as Workouts, Exercise Progress, Sensor Calibration, Tutorials, Account Information and Settings. Under the Workouts tab, a variety of exercises will be available to choose from. Once an exercise is chosen, the sensors will start looking for motion that matches the motion associated with that exercise. Under the Account Information and Settings tab, the user can find information for Support, and About Section with version identification, Frequently Asked Questions (FAQ), and Log Out.

5.4 Algorithm Description

The following subsections will outline the various algorithms in place to interpret the sensor data.

5.4.1 Trajectory Optimization

Trajectory Optimization is the process of designing a trajectory that minimizes (or maximizes) some measure of performance while satisfying a set of constraints. Generally speaking, trajectory optimization is a technique for computing an open-loop solution is either impossible or impractical.

5.4.2 Kalman Filter

Kalman filtering is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates

of unknown variables that tend to be more accurate than those based on a single measurement, by using Bayesian inference and estimating a joint probability distribution over the variables for each timeframe.

The Kalman filter is a widely applied concept in time series analysis used in fields such as signal processing. Kalman filters also are one of the main topics in the field of robotic motion planning and control, and they are sometimes included in trajectory optimization.

The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no addition past information is required.

The Kalman filter does not make any assumption that the errors are Gaussian. However, the filter yields the exact conditional probability estimate in the special case that all errors are Gaussian distributed.

6 Project Prototype Construction and Coding

6.1 Initial Development Environment

In order to finalize our pcb design it was necessary to utilize some simple prototyping. To accomplish this we are utilizing the components we purchased, as mentioned above in the additional component selection. Namely these components are the two breakout boards that we are basing our pcb design on, the Adafruit BNO055 Absolute Orientation Sensor, and the Adafruit Feather nRF52 Bluefruit LE. Fortunately the nRF52 supports the Arduino IDE which simplifies the development process by abstracting the low level environment from us.

With this environment we will be able to test and confirm basic qualities of the components that we are interfacing with as well as open us up to new experimentation. One such example of planned experimentation is to interface a battery charge reader that will use the microcontrollers ADC (Analog to Digital Converter) to read the voltage value directly from the battery as the voltage value will drop with respect to remaining charge. Another planned prototype is to interface the microcontroller to a Piezo Buzzer. A Piezo Buzzer is basically a small cheap little speaker that can utilize the PWM (Pulse Width Modulation) from the microcontroller in order to output simple tones. This would enable us to provide feedback to the user when the exercise goes out of range or even play a little tune at bootup.

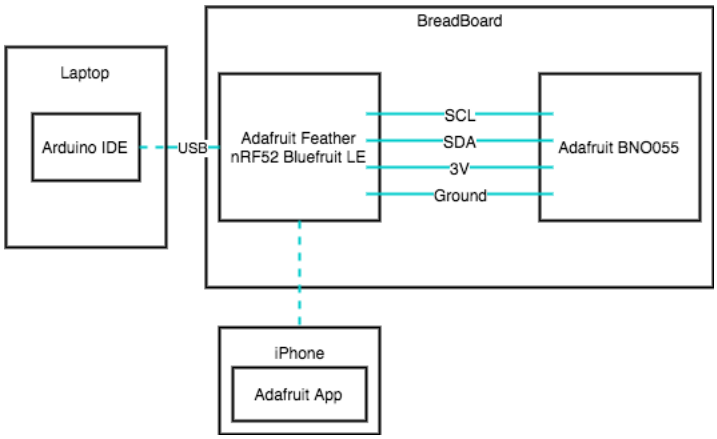


Figure 6.1: Initial environment for Prototyping

In the remainder of this section simple software sketches will be run in the Arduino IDE with out respective breakout boards to verify and expand component selec-

tion. As components are acquired we will first runs tests within the development environment before including it in the final PCB design.

6.1.1 Bluetooth Phone Communication

A simple test was run in order to see if we could transmit data back and forth over Bluetooth to the phone. In order to do this we hooked up our Adafruit Feather nRF52 to a computer and pushed a simple arduino sketch premade by adafruit that allowed us to interface to the Bluetooth devices on the Adafruit application. This setup basically lets us set strings back and forth between the connected devices. This transmission is done over UART.

After performing the test, we were able to confirm that we were able to transmit data back and forth from the microcontrollers without any hiccups. Simple strings along with more complex data from the iPhone to the microcontroller were able to be transmitted such as the iPhone's accelerometer, gyroscope, accelerometer data, and strings inputted by users. From the microcontroller we were able to transmit ASCII strings to the iPhone through a UART protocol.

6.1.2 Bluetooth LE Throughput

In order to test Bluetooth transmission throughput we utilized another Adafruit sketch called throughput that attempted to send 20480 bytes over bluetooth to the phone application and returns the time. In previous estimations we expected the iPhone to be able to support 5333 bytes per second since the smallest connection interval we can obtain is 15ms and each connection interval sends 4 packets of 20 bytes per packet. Assuming the IMU data is able to fit in each connection interval (which we expect it to be able to) this yields a refresh rate of 66Hz on the IMU data. This should be a good target since the IMU data only refreshes at a rate of 100Hz anyways.

After running the test through the Arduino IDE the following results were achieved. 20480 bytes were able to be received in about 3.7 seconds by the iPhone. This yields a rate of 5319 bytes per second which would suggest about 66 connection intervals were obtained per second. This means our refresh rate could conform as expected to 66Hz with each connection interval delivering 80 bytes each. These results are extremely encouraging as it proves that our theoretical assumptions with respect to the iOS environment and Bluetooth LE were correct therefore we can expect minimal issues with respect to data throughput although another test should be run once two devices are up and running to ensure throughput is maintained when two connections are sustained.

Table 6.1: iOS Bluetooth Throughput: Theoretical vs Actual

	Theoretical	**Actual**
Bytes per second	5333	5319
Connections intervals per second	66	66

6.1.3 IMU Communication

We connected the microcontroller to the IMU through the I2C interface and similarly utilized a prebuilt communication test supplied to us through Adafruit that enabled us to quickly receive data from the IMU. This test was successful and were able to receive all the expected IMU data, including the both the gyroscope and magnetometer readings and print them out to the serial monitor. In this test specifically we did not choose to receive accelerometer data since it was not necessary however in the future we will require this so will have to modify the configuration of the IMU in order to do so.

All that would be required now that we have utilized both Bluetooth communication and IMU is thread them together so we pool for the IMU data while we preparing packets for the Bluetooth connection interval. The next step from here would be to weave in commands and responding from different events that need to be responded to such as a audio beep request from the iPhone in response to improper form or LED color changes in response to battery level changes. This contact pooling will be researched and discussed later in the software sections.

6.1.4 Piezo Buzzer

In order give audio feedback to the user in the occurrence of a deviation from the expected exercise we are going to make a noise utilizing the piezo buzzer. To do this we utilize Pulse Width Modulation across the piezo buzzer from the microcontroller at a certain frequency. This will in effect create a tone which we can specify by way of the frequency passed. Pulse width modulation basically just uses timed interrupts to change the signal from high to low so we can simulate a square waveform at a certain frequency. The piezo buzzers we purchased were from Amazon and rather cheap at about 10 dollars for 10 (1 dollar per).

As for actual interfacing this component with nrf52 it was rather simple. We hooked up the speaker to PIN A3 on the breakout board with the positive terminal to the pin and the negative terminal to ground. Once actually connected Adafruit has some samples that use the nrf52 pulse width modulation unit which we adapted to be output to our speaker configuration. Once interfaced and the code uploaded we were able to confirm that we could hear audio from the speaker. After this we utilized a simple for loop which would lower the voltage gradually to confirm we could create seperate tones depending on the frequency passed. This all went through without a hitch and the speaker successfully went through a series of tones.

As for planned final implementation we will most likely utilize a wrapper function which we will create that will contain preconfigured tones at short bursts. We will call these functions from our main program to create short little tunes for bootup and an alert for the user when the exercise deviates. Testing will be done at this point as to whatever tone will be most comfortable for the user. Unfortunately due to the implementation of the piezo buzzer there will be no option for us to be able to modulate how loud the volume is so it will be a one size fits all sort of deal. We will however manage to have a mute option in the phone app if you do not want to deal with it anymore.

6.2 PCB Vendor and Assembly

FermiTron Inc.

FermiTron Inc., established in 2010, is a full service electronics products design corporation which provides support to companies ranging from promising entrepreneurs to Fortune 500 firms. FermiTron Inc. has the capability to provide full turnkey product development or targeted assistance on a project depending on our needs. Some of those services include, electronic circuit design, RF/wireless, PC board layout, firmware development, prototyping and test services, production and production support. Luckily for the Smart Gloves team, FermiTron is located in Orlando. [52]

Develotech

Develotech provides services related to intellectual property (IP) creation, prototype and concept development through final production of printed circuit boards (PCBs) and software/firmware. Develotech concentrates on Electrical Engineering tasks and the software development required to create electrical hardware to the the changing requirements of modern technology. Develotech Inc. provides services related to circuit prototypes, analog and digital sensor development, testing,

design, and low volume PCB manufacturing. Through these processes, Patents, Intellectual property, engineering and business related documentation and certifications are supported in their development. [53]

7 Testing Plans

An important cycle of the software and hardware development cycle is testing. This part of the cycle is where the software/hardware being developed is tested for correct functionality. The software that will companion the smart gloves sensor will contain many features and parts that will be integrated together to become a part of the solution to offer a good user experience. To ensure such good experience, both the hardware and software but work as expected. It must also be able to recover from mistakes or exceptions gracefully without crashing the application or losing user data. Moreover, the algorithms and software features at play must also be able to perform simple and complex mathematical calculations that provide optimum output. To be able to guarantee this level of quality, the software must be tested. Following in this sections are tools provided by the Integrated Development Environment (IDE) that will be used to develop the companion iOS application for the smart gloves. There will be different types of testing that will be executed at different phases of the development cycle. For details are in the paragraphs to come.

7.1 Hardware Test Environment

Our hardware test environment will be extraordinarily similar to our prototype environment setup and running similar tests to ensure that individual components will operate as expected. The basic layout of the testing setup will by the post bootloaded circuit wired up through a usb interface to a computer. Within the computer tormentor we will compile and run arduino sketches that will test a multitude of component wise functionality like Bluetooth communication, BNO communication, power measurement among others. For communication with the phone we will of course have to have the software application as a companion for this. However in the beginning it will be of no consequence to simply utilize the adafruit application to perform initial testing.

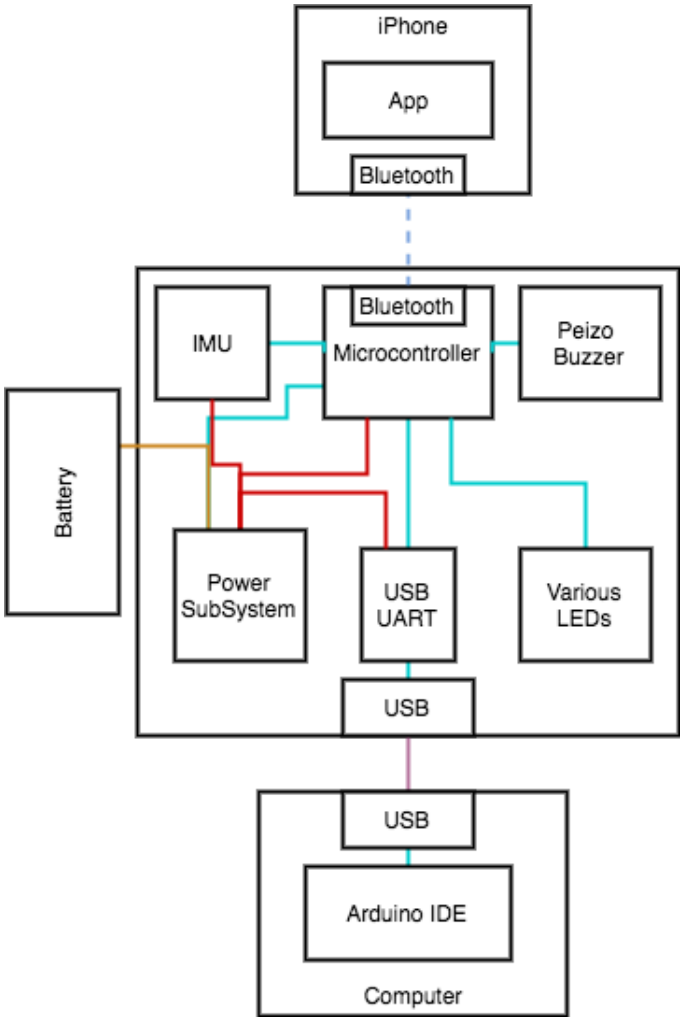


Figure 7.1: Hardware Test Environment

7.1.1 Arduino IDE

The Arduino IDE simplifies our development environment by providing a communication channel to program and develop on our Microcontroller. Adafruit thankfully allows us to utilize their custom board interface that they created to interface the nrf52 with Arduino IDE. This package also contains high level wrapper functions for things like pulse width modulation (PWM), direct to memory access (DMA) , analog to digital conversion (ADC), Bluetooth communication and other basic microcontroller functionality. This abstraction from the low level assembly level environment will not only kickstart our development environment on the nrf52, but also allow less mistakes as we go since programming at an assembly level can be an extremely difficult task.

7.2 Hardware Test Strategy and Implementation

This section is going to outline the testing strategy and planned implementation of basic component wise tests such as the power subsystem, battery, IMU, and microcontroller. The objective of the hardware tests are simply to ensure proper working functionality of all the components before moving on to full on software development/testing. These tests will have somewhat nested requirements upon each other as the IMU needs the microcontroller to work which in turn needs the power subsystem to work. The majority of these tests will require the USB to UART and microcontroller functioning properly as that is the only way to read and output to certain components like the piezo buzzer.

7.2.1 Hardware Test Strategy

The test strategy going forward is going to be to a form of integration testing. We will attempt to perform a bottom up test plan when possible. That is we will test individual components when possible then test their integration together, utilizing component drivers when necessary to simulate a normal operating environment. Due to the nature of the component wise testing the all of the testing work will be blackbox testing as we can not modify the nature of any of the components therein. Evaluation of the test results will be of a pass/fail caliber with comparisons to the requirement specification when applicable.

7.2.2 Power Subsystem Tests

The power subsystem is the part of our circuit that provides output power to all parts of our design. Thus it is the first subsystem to be analyzed for proper functionality since the further components have a strict reliance on this systems proper operation. The setup of this test will involve the circuit layout with usb power being received from the computer to the usb-uart bridge. Its not necessary for the microcontroller to be fully functional or bootloaded at this time as we will not need to interface with it. This test will have several stages to ensure proper functionality of the power subsystem such as connecting/disconnecting the battery, and measuring voltage across VBUS, 3.3V, and VBAT.

Table 7.1: Power Subsystem Test 1

Test Identifier	hw-pw1
Objective	USB-UART battery power being recieved
Description	Ensures that USB connection is supplying power to the power subsytem by measuring voltage accross VBUS and 3.3V
Test conditions	USB power being recieved to USB to UART bridge, battery disconnected
Expected result	Between a 3.3 and 5 volt reading from mutlimeter at VBUS and a minimum voltage of 3 volts at 3.3V
Dependencies	none

Table 7.2: Power Subsystem Test 2

Test Identifier	hw-pw2
Objective	Power Subsystem able to charge battery
Description	Ensures the battery is able to charge by way of the power subsystem by reading the voltage at VBAT and ensuring the orange charging LED is powered from the LIPO Charger
Test conditions	USB power being received to power subsystem
Expected result	Between a 3.3 and 5 volt reading from mutlimeter at VBAT and the Orange LED attached to the LIPO Charger is on.
Dependencies	hw-pw1

Table 7.3: Power Subsystem Test 3

Test Identifier	hw-pw3
Objective	Power for linear regulator being sourced from USB power when avalible
Description	Ensures the power preference to the linear regulator is being sourced from the USB line or VBUS when available and VBATs transistor is functioning as intended by measuring voltages with a multimeter at both VBAT and the linear regulators input terminal.
Test conditions	USB power is disconnected and VBAT is solely powering the circuit.
Expected result	Read voltage value at VBAT is different then the voltage value at the linear regulators input
Dependencies	hw-pw1

Table 7.4: Power Subsystem Test 4

Test Identifier	hw-pw4
Objective	Linear regulator is able to source power from VBAT when VBUS is unavailable
Description	Voltage is measured at both VBAT and 3.3V nodes.
Test conditions	USB power being recieved to power subsystem and charged battery is connected
Expected result	Between a 3.3 and 5 volt reading from mutlimeter at VBAT and a minimum voltage of 3 volts at 3.3V
Dependencies	none

7.2.3 Microcontroller Tests

The microcontroller tests will be utilized to ensure that we can properly push custom programs to the microcontroller through the Arduino ide. These test will also cover a basic Bluetooth communication test as the Bluetooth element is baked into the nrf52. Note that these tests will assume that the steps to bootload the microcontroller have already been taken as we will need this step to be able to utilize the USB to UART bridge. Due to the nature of this test already checking for USB to UART functionality there will be no independent section for tests the USB to UART bridge.

Table 7.5: Microcontroller Test 1

Test Identifier	hw-mc1
Objective	Able to push programs to microcontroller through Arduino IDE
Description	A simple program will be compiled and pushed to the microcontroller where we output strings to the serial monitor
Test conditions	USB connection with computer
Expected result	Sketch able to run and successfully output to serial monitor on computer
Dependancies	hw-pw1

Table 7.6: Microcontroller Test 2

Test Identifier	hw-mc2
Objective	Bluetooth communication with phone
Description	Simple sketches similar to the ones we have initial used in prototyping will be pushed to the microcontroller to communicate with the iPhone. A two way serial communication will be used to ensure the phone is able to talk with the microcontroller and vice versa.
Test conditions	USB connection with computer and either adafruit or custom phone app running.
Expected result	Serial monitor on computer showing communication between the iPhone and microcontroller
Dependencies	hw-pw1, hw-mc1

Table 7.7: Microcontroller Test 3

Test Identifier	hw-mc3
Objective	LIPO battery charge reading
Description	Ensures we are able to read battery levels by pushing a sketch to the microcontroller that will use the ADC to output voltage readings of VBAT to serial monitor.
Test conditions	USB connection with computer and battery connected.
Expected result	Serial monitor on computer printing voltage readings from ADC
Dependencies	hw-pw1, hw-pw2, hw-mc1

7.2.4 Inertial Measurement Unit Tests (IMU)

The tests for the Inertial Measurement Unit (IMU) will require proper microcontroller functionality as we will need to communicate with the IMU through the I2C interface. To ensure proper functionality of the IMU we will not only have to check that the I2C connection is working properly but that the IMU's data for positional tracking is updating correctly as we reorient it. Thankfully Adafruit has a few sketches that interface with the BNO which we will be using as a baseline for our tests.

Table 7.8: IMU Test 1

Test Identifier	hw-imu1
Objective	IMU communication and orientation data retrieval
Description	A sketch will be pushed to the microcontroller which will communicate over I2C with the IMU. Once the data has been retrieved the IMU data will be continously printed to the serial monitor. The IMU will be rotated and moved to ensure the data changes accordingly.
Test conditions	USB connection to computer
Expected result	Serial monitor prints formated positional data and data updates as IMU is handled
Dependencies	hw-pw1, hw-mc1

Table 7.9: IMU Test 2

Test Identifier	hw-imu2
Objective	IMU communication latency test
Description	We will print the IMU orientation data along with a time stamp while reorienting IMU.
Test conditions	USB connection to computer
Expected result	IMU data is updating with a latency of less then 0.5s as per specification requirement
Dependencies	hw-pw1, hw-mc1, hw-imu1

7.2.5 Bluetooth Communication Tests

In this section we will cover the tests relating to Bluetooth communication. Two such tests are the throughput test that we performed in the prototyping section as well as a latency test to check the duration it takes to send a message from the microcontroller to iPhone and vice versa.

Table 7.10: Bluetooth Communication Test 1

Test Identifier	hw-bt1
Objective	Bluetooth data throughput
Description	20480 bytes will be sent over Bluetooth the iPhone, once this process has competed the microcontroller will print out how long this process took.
Test conditions	USB connection with computer and either adafruit or custom phone app running.
Expected result	About 3.7s, yielding about 5300 Bytes a second.
Dependencies	hw-pw1, hw-mc1, hw-mc2

Table 7.11: Bluetooth Communication Test 2

Test Identifier	hw-bt2
Objective	Bluetooth latency
Description	A timestamp will be sent to the phone application which will then be returned and compared against and output to the serial monitor.
Test conditions	USB connection with computer and either adafruit or custom phone app running.
Expected result	$<$100ms (40ms anticipated)
Dependencies	hw-pw1, hw-mc1, hw-mc2

Table 7.12: Bluetooth Communication Test 3

Test Identifier	hw-bt3
Objective	Bluetooth distance test
Description	The phone will run the serial communication test from hw-mc2 and we will check the maximum distance communication can be performed at.
Test conditions	USB connection with computer and either adafruit or custom phone app running.
Expected result	minimum 2m as per specification requirement
Dependencies	hw-pw1, hw-mc1, hw-mc2

7.2.6 Piezo Buzzer Test

With the Piezo Buzzer connected we will attempt to use pulse width modulation to push a valid frequency to it (10 kHz $\rightarrow$ 100 kHz). The piezo buzzer is reliant

on the microcontroller being able to receive sketches so we will have to be able to interface with the microcontroller via UART bridge before we can perform this test.

Table 7.13: Piezo Buzzer Test 1

Test Identifier	hw-aud1
Objective	Audio output test
Description	A sketch will be pushed to the microcontroller that will utilize PWM to push a valid frequency to the piezo buzzer.
Test conditions	USB connection with computer
Expected result	Sound emitted from buzzer
Dependencies	hw-pw1, hw-mc1

7.2.7 Battery Life Test

This test will catalog the life a fully charged battery at constant Bluetooth communication and ensure that this meets our requirement as defined in the requirements specification. We will do this by running the test for Bluetooth throughput on a loop to ensure that the maximum expected power consumption is being utilized.

Table 7.14: Battery Life Test 1

Test Identifier	hw-bat1
Objective	Battery life test
Description	The device will be started up with a sketch to run hw-bt1 in a loop thereby consuming max power. The power level read from the ADC and a timestamp will be printed to the serial monitor until the battery runs out.
Test conditions	USB connection with computer and either adafruit or custom phone app running.
Expected result	2 hours min as per requirement specification
Dependencies	hw-pw1, hw-mc1, hw-mc2, hw-mc3, hw-bt1

7.3 Software Test Environment

Android Studio is Android's office Integrated Development Environment (IDE). It is built for Android to accelerate your development and help build apps for any Android device. The tools within Android Studio are custom-tailored for Android developers. These features include rich code editing, debugging, testing, and profiling tools. [54]

Android Studio has a unique feature called Instant Run. Instant Run allows the IDE to push code and resource changes to the running application. Instant Run intelligently understands the changes and can deliver them to the application without restarting the app or rebuilding the APK. This results in effects that can be seen immediately and greatly increases development speed.

The Android emulator installs and starts applications faster than a real device and can allow a developer to prototype and test an app on various Android configurations. The emulator can emulate a variety of devices such as: phones, tablets, Android Wear, and Android TV devices. Additionally, Android Emulator can simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

Android Studio offers build automation, customization build configurations, and dependency management. Developers can configure a project to include local and hosted libraries, define build variants from different code and resources, and apply different code and app shrinking configurations. Android Studio provides a unified environment for developers to build apps for various devices. Structured code modules allow you to divide your project into units of functionality that developers can independently build, test, and debug.

Android Studio integrations with version control tools, such as GitHub and Subversion, so developers can keep their teams in sync with the project and build changes. The open source Gradle build system allows developers to tailor the build to the proper environment and run on a continuous integration server such as Jenkins.

Android Studio provides extensive tools to help developers test and Android applications with JUnit 4 and functional UIO test frameworks. With Espresso Test Recorder, developers can generate UI test code by recording interactions with the app on a device or an emulator. Developers can also choose to run tests in Firebase Test Lab.

Android Studio Lintelligence provides a robust static analysis framework and includes over 280 different link checks across the entirety of an application. Additionally, it provides several quick fixes that help developers address issues in various categories such as performance, security, and correctness, with a single click.

In Apple's ecosystem, Xcode is the primary Integrated Development Environment (IDE) used to develop iOS applications for the iPhone, iPad, iPod Touch, Apple Watch, Mac, and Apple TV. In order to create robust applications that have minimal bugs and are also quick to market, Xcode provides capabilities for extensive software testing. Testing software is a behavior that is practiced among all software and electrical engineers. Developing a well tested application helps to create a product that performs as expected and increases user satisfaction. As stated in Apple's documentation, testing can also help develop applications faster and further, with less wasted effort, and can be used to help multi-person development efforts stay coordinated. [55] On the following paragraphs, testing iOS applications will be discussed in details. These include measuring performance from our test. This will help to understand whether different implementations or code improve or degrade performance. The UI will also be tested to ensure that elements render properly with various datasets.

As the different forms of testing are discussed further, it is important to note that these tests are written in either Objective-C or Swift. These are the two programming languages used to develop iOS applications. The recent release of Swift makes it an attractive programming language to learn as it will eventually replace Objective-C for all development done on Apple's ecosystem. All the available test

for a specific application are shown on the Test Navigator pane. The Test Navigator pane shows a list of all the test bundles with their corresponding classes and methods. It also shows whether a test has passed by displaying a green checkmark or that a test has failed by displaying a red x. In Java, unit tests can be written either in JUnit or other unit testing frameworks such as TestNG. However, in Xcode and iOS mobile development, the XCTest framework is used to run these unit tests, performance tests, and UI tests; all within Xcode. Assertions are used to ensure that the desired conditions and results are satisfied during code execution. Test failures are recorded with optional messages if those conditions aren't satisfied. Before going into detail on each type of test, a brief overview of the capabilities of each test type will be discussed.

For starters, a test is code that is written to exercise an app and code library that lead to either a pass or fail result. These results are measured against a set of expectations. This means that a test could check the state of an object's instance variables after performing some operations. It can also be used to ensure that the application throws a particular and correct exception when subjected to a boundary condition. All of this are examples of potential unit tests. For performance tests, the maximum amount of time is calculated to record the expected time versus the actual time for a set of routines to run to completion. For the purposes of this Smart Gloves project, a unit test example would be ensuring that retrieving and displaying a various set of data is done accurately without changing the state of objects that may affect the application's input or output at a later point. Another example is in the case that the user is exercising in a form that is damaging to their body. These cases can be considered as edge or boundary cases as each user may performance unique types of unexpected motion. Testing the application on these edge/boundary cases is essential to the success of the application and to the health of the user. In the case of performance testing, there could be an instance in which a user is entering a lot of data or that the sensor is collecting a vast amount of data from a user's workout. Responsiveness and app availability is crucial since the user's health is in play. To avoid injury or life threatening situations it is important to compute and provide feedback to the user as quickly as possible in real time. The XCTest testing framework provides the capability to do so.

In order to cover tests of all possible units, it is important to note the software design principle of composition. All software is built using it. This means small components are arranged together to form larger, higher level components with greater functionality until the final product goal is met. It is of good testing practice to have tests that cover functionality at all levels of this composition. A great and important feature of XCTest framework is that is allows to writes tests for components at any level. Splitting each software component is an important design pattern. One that will be followed during the development of the companion mobile application for the Smart Gloves.

Test-driven development is a style of writing code in which the test logic is written before writing the code to be tested. This is a standard design pattern used in industry that has a track record to facilitate testing and leads to positive results. Writing the tests first provide a foundation for which once the tests pass, improvements to the code in test can be made and therefore decreasing the amount of bugs in the application. To emphasize more on Unit Testing, tests help reduce the introduction of bugs in code as it is modified to enhance features and functionality. In the context of this project, tests can be incorporated in the any working or prototyped state to ensure that future changes don't modify the app's existing behavior other than in the planned ways. Software bugs can be inevitable. There are many edge and boundary cases that could happen and could be a challenge to come up with. As fixes for bugs are written, new tests can help ensure that these bugs have been fixed.

The XCTest framework provides APIs (Application Programming Interface) to measure time based performance. This enable to track performance improvements and regressions. As stated in Apple's documentation, in order to provide a success or failure result when measuring performance, a test must have a baseline to evaluate against. XCTest calculates a baseline from a combination of the average time performance in ten runs of the test method with a measure of the standard deviation of each run. XCTest evaluates these test runs and marks a failure when test drop below the time baseline or vary too much from run to run.

The third and final potential type of test needed for to test the companion application for the Smart Gloves would be UI (User Interface) testing. As mentioned in Apple's documentation on user interface testing, unit testing is primarily concerned with forming good components that behave as expected and interact with other components as expected. From a design perspective, unit testing approaches development from its inside. It is important to note that while testing these internal components is essential for the success of the iOS application, the app must also be tested in a way that the user will interact with it. All users will interact with the internals of the app through the user interface. It is difficult to write unit tests to exercise the interactions that users experience though the UI. UI tests approach testing apps from the external surface. Just as the user experience it. This type of testing enables to write code that send simulated events to both system and custom UI objects. The response is then captured and tested for correctness or performance much like the internally oriented unit tests.

Just like normal code can have bugs, test code can also have bugs. Xcode provides the capabilities to debug tests. Debugging can provide visibility to an object's state and behavior while also expose unwanted values and interactions. One of the first things to determine when facing a problem is to determine if the failure is originating from a bug in the code that is being tested or a bug in the test method that is executing. As mentioned on Apple's documentation, test failures could point to several different kinds of issues. It could be that incorrect assumptions are be-

ing made therefore leading to wrongly assertions. Debugging tests can span to several different workflows. Tests are typically short in length and also straightforward. Apple advises to first examine what the test is intended to do and how it is being implemented. In order to trigger a debug point, breakpoints are set in code on a desired line. Xcode will then show all the available properties for the object's created and the ability to step through the code line by line is provided for further investigation and understanding of behavior.

Another important metric to consider while testing is code coverage. Xcode 7 enables the visualization and measurement of how much code is being exercised by tests. With the code coverage provided, it can be determined whether tests are doing the intended job. In iOS development, code coverage in Xcode is a testing option that is supported by LLVM. LLVM is compiler used to compile Swift and Objective-C code. When this is enabled, LLVM instruments the code to gather coverage data based on the frequence that methods and functions are called. In industry, code coverage metrics is a way to measure the value and quality of code and tests. It is a way to determine which lines of code are being tested and perhaps also which lines of code are dormant and not necessary. When talking about high scalable systems, every line of code matters. Therefore is important to practice writing code that is actually being used and to factor in areas that could use improvement. Small refactoring of code can lead to significant performance and functional improvements and so this is worth noting. Code coverage will be kept in mind and will be at the heart of throughout the development process.

This senior design project will be treated with the quality and care as production systems in the wild. After executing tests, Xcode takes the LLVM coverage data and uses it to create a coverage report in the Reports navigator. A list of source files and functions within the files are showing as well as the coverage percentage for each. The source editor shows the count for each line of code in the file and highlights code that was not executed. This is a great way to quickly see code that is could be refactored or otherwise improve testing strategies. Also within the right pane of the code editor, a count for how many times a particular part of code was hit during the test is shown. This is extremely helpful to quickly discern critical code and workflows. Like mentioned before, only sections that were not called or covered are highlighted in red with a zero (0) coverage count. Code coverage report data suggests an opportunity to write a test that includes unexpected or invalid characters to be sure that error handling works as intended or that passed in parameters are processed as desired.

All the of the testing methods discussed above are useful but can be cumbersome to execute manually. Fortunately, Xcode provides ways to execute these tests in an automated way. This is greatly beneficial and a time saver. In the grand scheme of things, this can help reduce development time in a project with a short deadline such as this Smart Gloves senior design project.

The goal for this project is to develop and build a robust hardware and software solution. One that keeps the user at its core functionalities and design. Keeping this in mind, the companion iOS application to be created for the Smart Gloves will go through in depth testing like the methods discussed above. Xcode also provides a useful application called Instruments. As stated in Apple's documentation, Instruments is a powerful and flexible performance-analysis and testing tool that's part of the Xcode tool set. It is designed to help profile Mac OS and iOS apps, processes, and devices in order to better understand and optimize their behavior and performance. Like unit, performance, and UI tests, incorporating Instruments into the testing workflow from the beginning of the development process can help save time late by helping expose issues early in the development cycle. [56]

Apple markets Instruments well on its documentation as they mention that unlike other performance and debugging tools, Instruments allows to gather widely disparate types of data and view them side by side. It is their argument that this makes it easier to identify trends that might be otherwise overlooked. By integrating instruments to the development cycle of the companion app the behavior of the apps and its processes can be examined. As well as device specific features like WiFi and Bluetooth. In this case, it will be important to get in depth analysis of how the Bluetooth connection and performance is behaving between the iPhone and the Smart Gloves sensor. The testing methods discussed above do a great job at testing the software components. Instruments provide great insight to how the application and the Apple device, in this case the iPhone, are behaving when communicating and interacting with the sensor. In the case that issues or unexpected behavior arise, Instruments will help debug those issues.

Another great thing about Instruments is that profiling can be done on a simulator, meaning that the application would be running on a simulated environment on the Mac. Or it can also be done on a physical device. This allows for versatility in testing and can help lead to great understanding of how the app is behaving with the hardware. Another common issue that can arise with software applications are resource leaks such as memory leaks. Also abandoned memory and zombies fall under this category. Instruments will help facilitate the understanding of these potential issues if they every come up. Lastly, the application will be running on a mobile device that uses battery for its power functionalities. It is important to develop an app that is power efficient so that the user's device battery is not completely drained during a workout. Instruments provides insights on power managements and provides ways to optimize the app for greater power efficiency.

7.4 Software Test Strategy and Implementation

As discussed before, testing is an important component of the software development cycle. Without testing, there is one way to ensure that the application will

provide expected outputs with various data sets and various scenarios. The testing strategy for this project is very simple, always test. As mentioned in the introduction of this section, there are various types of tests that can be performed. Under the functional testing umbrella fall unit testing, integration testing, system testing, and acceptance testing. Apart from functional testing, there is also non-functional testing. Within it lies performance testing, security testing, usability testing, and compatibility testing. All these forms of testing fall under testing methodologies. Testing methodologies are techniques or forms of testing used to ensure the functionality of a software application. Imagine a new vaccine being released to the public without proper testing or a new vehicle engine released on new cars without proper testing. There is form of wrongful feeling when discussing about the release of products without the proper testing. In certain applications, the consequences of not thoroughly testing a product can be detrimental and lead to death. In the case of the smart gloves, the user can experience life long injuries by incorrect output. Such detrimental and fatal accidents are fully avoidable and in the following sections different testing strategies will be discussed to ensure the proper development and deployment of the software for the smart gloves.[57]

7.4.1 Unit Testing

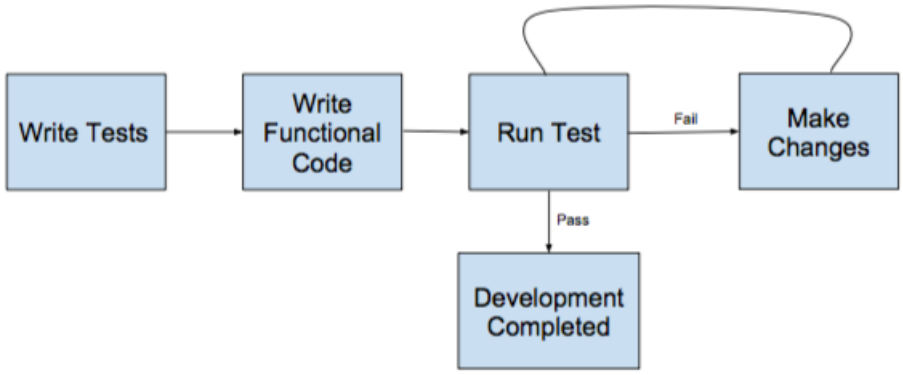


Figure 7.2: Test Driven Development Process

Unit testing has become a standard testing strategy that is practiced in industry. Throughout the development cycle, applications must reach a certain level of unit testing code coverage in order to pass accepting testing. While unit testing can be performed manually, these tests are often automated. Automating these test will increase productivity, time management, and provide a level of continuous integration. With automated unit tests, the user defines what units are to be tested. These units are often defined as the smallest piece of code that contains enough

functionality that would provide significant functionality. Since defining a unit for testing is left at the developer's discretion, it can be both beneficial and harmful. Scenarios in which the developer chooses the right units to test can occur while selecting the wrong units to test or no selecting any at all would be detrimental to the application's health.

A huge benefit of unit testing is that errors can be identified earlier in the development cycle and therefore provide the opportunity to find a quick fix before integrating with out vital sections of the application. In order to facilitate unit testing, software driven development is practiced. Like unit testing, software driven development is a best practice that many use industry. It is often found to be a required skill in job posts and it is a skill that has proven to work and provide great results. In test driven development, or TDD, the tests to be performed are written first. These test contain the desired functionality wanted to be achieved when adding new features or refactoring code. Test driven development is also helpful to discern the requirements of a design before writing functional code. Another great benefit from test driven development is that it leads to written software that works.

As mentioned, tests are written first before any functional code. This means that the tests will first fail. This is the first step of test driven development. Once enough tests have been written to test each unit of the code to be written, the functional code is written. While writing the functional code, there will be checkpoints in which the tests will be executed. The goal is to see some tests to begin passing. This process is to be repeated until all test produce a passing output. The figure above shows a diagram how this process. Throughout the development of this project, test driven development will be used to test the functional code that is written and a code coverage of 80 percent will be a requirement.

7.4.2 Integration Testing

After all the different software components that make up the smart gloves are developed and tested, integration testing will be executed. The goal of integration testing is to test all the modules and components to ensure that they work together as expected. It is expected for this project to contain multiple components. For example, a component that sends and receives data from the inertial measurement unit, or IMU, must be able to also be able to integrate and pass on the data received to a component dedicated to display data to the user. Integration test is planned to be performed once all unit tests and components have been completed. It is crucial for all unit tests to be passing at this point to ensure a smooth transition into this point.

Like unit tests, integration tests can be completed in an automated or manual manner. Both of these method provide its own benefits. Tests can be performed quicker when completed in an automated way. This allows for data to be mocked and allow

for various datasets to be tested. With automated testing and data mocking, data sets that can only be achieved in extreme situations can be mocked and therefore a deeper understanding of how the application behaves and integrates can be learned. This provides powerful testing by giving the ability to test with large amounts of expected and unexpected data. This leads to a great point, testing with data that could lead the application to crash. Users expect to use software that will not crash and continue to run even when errors occur. To accomplish this functionality, exception handling must be incorporated to the application. Exception handling is a technique that catches errors when they occur in code. Errors can be resolved in various way and it will depend on the use case. In the case where the user enters incorrect data, a message can be displayed notifying the user that the data entered is not correct and provide a hint that leads to a successful entry. Ensuring that the application is able to handle and recover from errors is essential and will lead to the success of this project.

7.4.3 Physical Testing

Since the use case of this project is to be used while the user is performance physical activity, the sensor and companion application must be tested while performing a vast array of exercises. This can become extremely difficult to test because the inputs and movements are unpredictable. Users will be of different height, weight and age. Each user will also have different physical abilities and techniques. Taking all these things into consideration lead to an infinity number of possible inputs and outputs. As expected, physical testing can only be done in a manual manner. There are possibilities of automating this form of testing as well but due to budget constraints, manual testing will be the only form of testing used. With that being said, physical testing will consist of placing the smart gloves sensor on a subject and activating the companion mobile application. The subject is then to perform various types of weight lifting exercises. Some of these exercises are to be performed in a way that would could intentional failure while other exercises are to be performed in a way that the correct output is expected. These activities will lead to the understanding of limitations as well as areas in which the system works well. Like unit testing, when performing physical testing, tweaks and changes will be made to help resolve bugs and develop the desired product.

7.4.4 Performance Testing

Performance testing an application can go overlooked as applications are mainly tested for functionality. In the case of this project, the smart gloves and the companion app must be able to manipulate and display data fast and efficiently. As mentioned multiple times throughout this paper, user experience is important, there-

fore it is important for users to interact with the devices in a manner for which response times don't affect the user experience in a negative way. The performance testing is not expected to be extensive but there will be forms of testing to ensure that the application's response time is kept as low as possible to enhance user experience.

8 Project Operation

The aim of the Smart Gloves device is to help individuals maximize the efficiency of their workouts and avoid serious injury. The user manual is designed to help the user successfully exercise safely and with proper form. Below is all the necessary information to safely use Smart Gloves to its fullest extent.

8.1 Safety Precautions

Exercise equipment is inherently dangerous. Various types of home gym devices are typically large and have moving parts. The majority of exercise injuries are due to excessive weights or quick movements. The Smart Gloves can help to provide safe guidance for a user. Smart Gloves are designed to provide a structured boost to a user's workouts while avoiding any sort of interference with a range of motion.

8.2 Troubleshooting Tips

Table 8.1: Troubleshooting Tips

Problem	Solution
The device will not turn on.	Make sure the battery is charged and that everything is connected properly.
The device will not connect the mobile application.	Turn the device off, wait 15 seconds, and turn the device back on.
The device is not calibrated correctly.	Repeat the calibration and orientation steps found within the companion application.
The device is not tracking the desired exercise.	Go into the companion application and reselect the desired application.

9 Administrative Content

9.1 Milestone Discussion

This section will break down the various accomplishments and milestones of the Smart Gloves project. Table 9.1 will show all of the milestones from the start of Senior Design 1 (August 2017) all the way to the final product presentation of Senior Design 2 (May 2018). The first section was Senior Design 1, where the majority of the research and development was done. The second section is Senior Design 2 where most of the integrating, testing, and redesign takes place. Most of the dates are estimated and the non-professor assigned due dates were decided by the team. The Project Milestones table below goes into much more detail.

Table 9.1: Project Milestones

Number	Task	Start Date	Due Date
Senior Design 1	-	-	-
	Idea	8/21/17	9/4/17
	Project Selection and Role Assignment	9/4/17	9/18/17
	Divide and Conquer	9/18/17	9/23/17
	Research, Documentation, and Design	8/21/17	9/23/17
	60 Page Draft	9/23/17	11/03/17
	100 Page Draft	11/03/17	11/17/17
	120 Page Final Submission	11/17/17	12/04/17
Senior Design 2	-	-	-
	Assemble Prototype	11/06/17	12/16/17
	Testing and Redesign	11/06/17	12/03/17
	Finalizing Prototype	12/04/17	TBD
	Peer Report	TBD	TBD
	Final Document	TBD	TBD
	Final Presentation	TBD	TBD

9.2 Budget and Finance Discussion

Table 9.2: Cost of Project (12/1/17)

Item	Supplier	Quantity	Unit Price	Final Price
IMU	Digikey	2	12.07	24.14
Microcontroller (nRF52832)	Digikey	2	10.1	20.2
Linear Voltage Regulator	Digikey	2	0.48	0.96
LIPO Charger	Digikey	2	0.59	1.18
PNP Filtering MOSFET	Digikey	2	0.92	1.84
Schottky Diode	Digikey	2	0.44	0.88
USB-UART Bridge	Digikey	2	1.65	3.3
Piezo Buzzer (10 pack)	Amazon	1	10.99	10.99
nRF52832 breakout board	Amazon	1	29.95	29.95
BNO055 breakout board	Amazon	1	30.27	30.27
Soldering kit	Amazon	1	18.99	18.99
Wires for interfacing	Amazon	1	7.29	7.29
Bluetooth USB Device	Amazon	1	13.95	13.95
			SUM	163.94

9.3 Logo and Branding

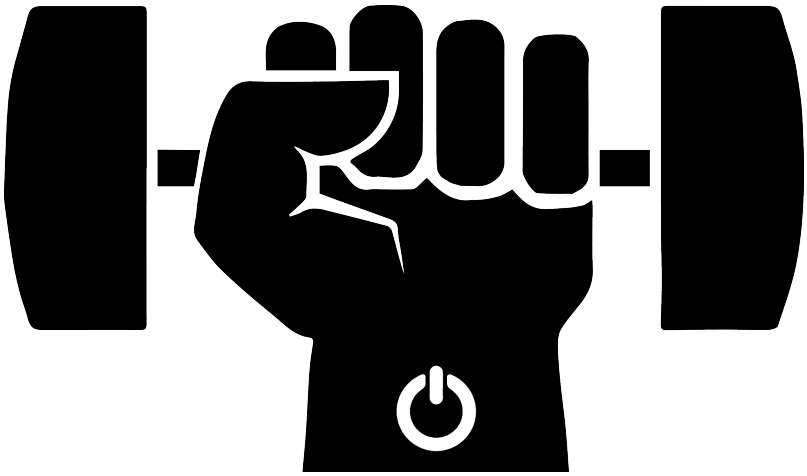


Figure 9.1: Smart Glove Logo

9.3.1 Symbols and Imagery

Gloves

The symbol of a glove often represents the hand itself. Gloves can signify high status and clean hands. However, gloves can also conceal. Gloves embody power and protection, as well as nobility.

Dumbbell

The symbol of a dumbbell represents exercise and strength. Dumbbells are associated the the gym, weightlifting, and working out. It is a symbol of an active lifestyle. One that cares about physical fitness and health.

Power Symbol

A power symbol is a symbol indicating a control that activates or deactivates a particular device. The well known on/off power symbol was the results of the logical evolution in user interface design. The symbol is typically associated with the future, technology, and electronics.

9.3.2 Color Theory

The use or colors in branding and marketing is incredibly important. Researchers found that up to 90% of snap judgments made about products can be based on

color alone. Results from other studies showed that the relationship between brands and colors hinges on the perceived appropriateness of the color bring used for the particular brand. It is also important for new brands to pick colors that ensure differentiation from other entrenched competitors. While brands can sometimes cross between two traits, but they are mostly dominated by one. Brands are far more important for colors to support the personality the company wants to portray instead of trying to align with stereotypical color associations. [58]

Red

Red is the color of fire and blood, so it is associated with energy, war, danger, strength, power, and determination as well as passion, desire, and love. Red is an emotionally intense color. It enhances human metabolism, increases respiration rate, and raises blood pressure. This color is also commonly associated with energy, so it cane be used to promote things such as exercise and physicality.

Orange

Orange combines the energy of red and the happiness of yellow. Orange represents enthusiasm, fascination, happiness, creativity, determination, attraction, success, encouragement, and stimulation. To the human eye, orange is a very hot color, so it gives the sensation of heat. Orange is not as aggressive as red. Orange increases oxygen supply to the brain, produces an invigorating effect, and stimulates mental activity. In heraldry, orange is symbolic of strength and endurance.

White

White is associated with light, goodness, innocence, purity, and virginity. It is associated to be the color of perfection. White means safety, purity, and cleanliness. In advertising, white can be used to suggest simplicity in high-tech products.

Black

Black is associated with power, elegance, formality, death, evil, and mystery. Black denotes strength and authority; it is considered to be a very formal, elegant, and prestigious color. Black gives the feeling of perspective and depth, but a black background diminishes readability. Black contrasts well with bright colors. Combined with red or orange, black gives a very aggressive color scheme.

10 Project Summary & Conclusion

Weightlifting can be unexpectedly difficult considering all the factors that come into play, such as exercise form, repetition speed, and exercise variety among many others. From the very beginning, the conceptual goal of our Smart Gloves design has been to provide users with a device that can maximize the efficiency of their weightlifting workouts and take out a lot of the guesswork that often stumps newcomers and experienced lifters alike. Of course, Smart Gloves will not be able to physically lift weights for its users, that part is still up to the individual; instead, it will provide meaningful simplifications in the workout process that will help lifters of all varieties get the most out of the time they spend in the gym. In addition to streamlining a user's weightlifting routine, Smart Gloves will promote safety by checking to ensure the user is using proper form and pacing themselves during repetitions. This will minimize the chance of injuries such as sprains, breaks, and tears, all of which could seriously impede a lifter's long-term progress and prevent them from reaching their personal goals. We also believe that our proposed design could potentially be used to great effect by physical therapists that are trying to rehabilitate patients who have suffered serious injuries, as Smart Gloves could be programmed to monitor a wide array of rehabilitative exercises.

Following a conceptual establishment of the goals and motivations behind our Smart Gloves design, we determined that the use of an Inertial Measurement Unit would offer the most design flexibility and provide relative positioning data that could be used to accurately track the motion of a user's two hands. Likewise, we determined that Bluetooth LE wireless communication protocols would best serve our design and determined other components that would need to be implemented such as microcontrollers and linear voltage regulators. After identifying which major components would be necessary, we compared the identifying characteristics of multiple considerations and determined which components would be used in our project design. Using these components, detailed schematics were created in Section 5.1 that established the connections between the three major sub-systems of our design and outlined how each component will function in our proposed implementation. Using breakout boards containing the bulk of our components, we began prototyping our design by testing communication and data transmission. Further hardware tests that will be performed on our finalized prototype were further discussed in later sections, each with the goal of realistically testing the hardware requirement specifications set up in the beginning of the report.

Upon beginning Senior Design this semester, most members of our group lacked experience designing a project of this scale. Through many hours of self-guided research and trial-and-error, we learned the skills necessary to take the conceptualization of Smart Gloves and turn it into a realistic design that will meet the goals, objectives, and requirements that were set for this project. As evident in this paper,

all work done on this project was split into hardware and software categorizations to evenly distribute the workload across all four members and cater to each individual's strengths. Many challenges and obstacles were overcome throughout the semester, and in the end all of us have grown tremendously through this process, which has left us better equipped to enter the engineering profession upon graduation.

A Appendix - Referenced Designs

A.1 Adafruit Feather nRF52 Breakout Board

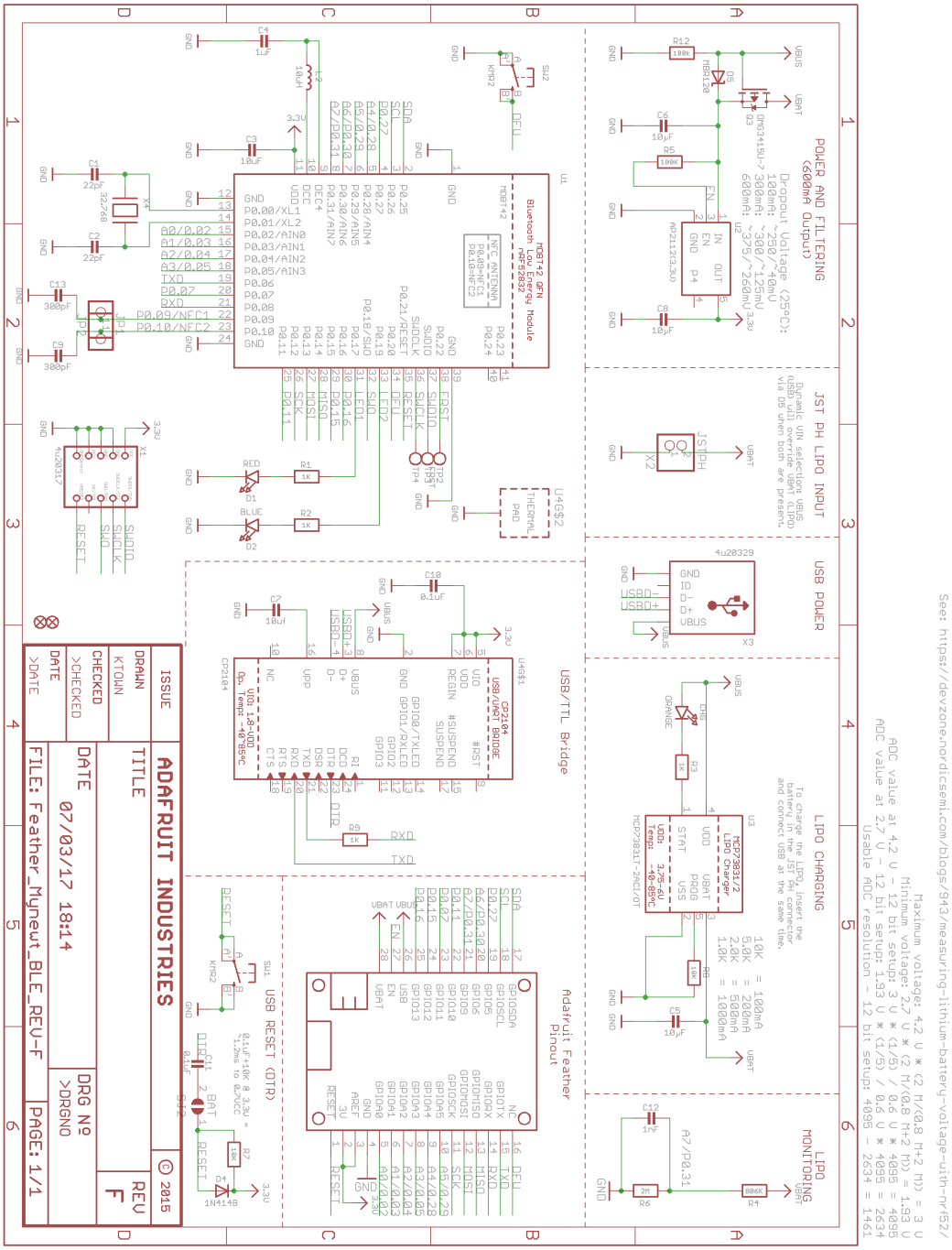


Figure A.1: Adafruit Feather nRF52 Breakout Board [Appendix B.3]

A.2 Adafruit BNO055 Breakout Board

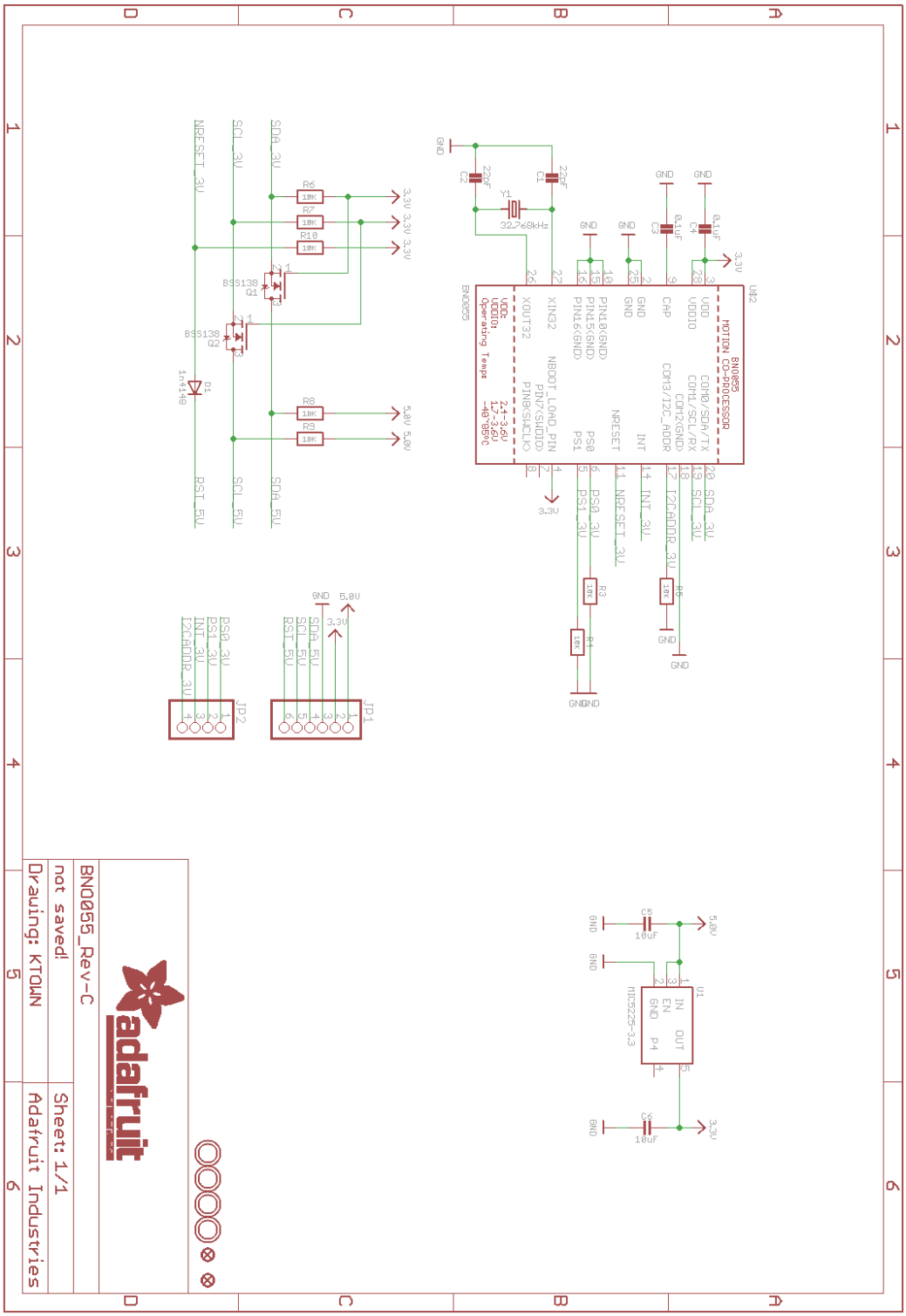
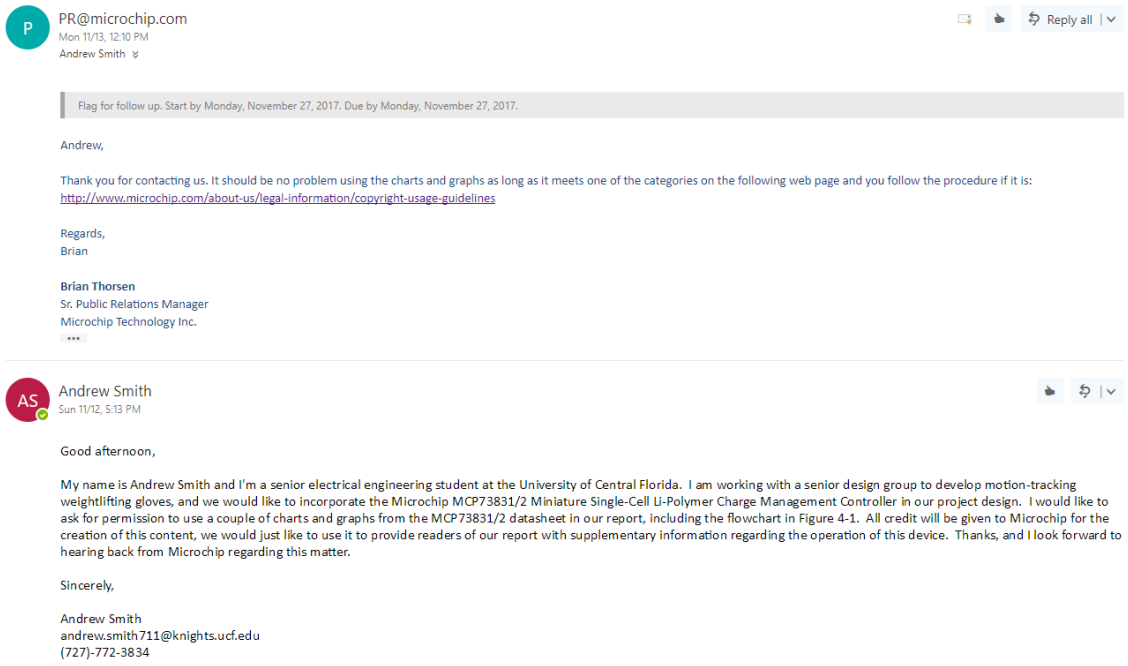


Figure A.2: Adafruit BNO055 Breakout Board [Appendix B.4]

B Appendix - Permissions

B.1 Microchip LIPO Charger Flowchart Permission



PR@microchip.com
Mon 11/13, 12:10 PM
Andrew Smith

Reply all

Flag for follow up. Start by Monday, November 27, 2017. Due by Monday, November 27, 2017.

Andrew,

Thank you for contacting us. It should be no problem using the charts and graphs as long as it meets one of the categories on the following web page and you follow the procedure if it is:
http://www.microchip.com/about-us/legal-information/copyright-usage-guidelines

Regards,
Brian

Brian Thorsen
Sr. Public Relations Manager
Microchip Technology Inc.

Andrew Smith
Sun 11/12, 5:13 PM

Good afternoon,

My name is Andrew Smith and I'm a senior electrical engineering student at the University of Central Florida. I am working with a senior design group to develop motion-tracking weightlifting gloves, and we would like to incorporate the Microchip MCP73831/2 Miniature Single-Cell Li-Polymer Charge Management Controller in our project design. I would like to ask for permission to use a couple of charts and graphs from the MCP73831/2 datasheet in our report, including the flowchart in Figure 4-1. All credit will be given to Microchip for the creation of this content, we would just like to use it to provide readers of our report with supplementary information regarding the operation of this device. Thanks, and I look forward to hearing back from Microchip regarding this matter.

Sincerely,

Andrew Smith
andrew.smith711@knights.ucf.edu
(727)-772-3834

Figure B.1: Microchip LIPO Charger Flowchart Permission

B.2 Basic Microntroller Layout - Figure 3.5

B.3 Adafruit Feather Schematics - Figure A.1

B.4 Adafruit BNO Schematics - Figure A.2

Designed by Adafruit Industries. licensed under Creative Commons Attribution, Share-Alike license 3.0 Available at *https://creativecommons.org/licenses/by-sa/3.0/us/*

Bibliography

[1] Thilakshan Kanesalingam. Motion tracking glove for human-machine interaction: Inertial guidance. *https://macsphere.mcmaster.ca/bitstream/11375/14425/1/fulltext.pdf*, April 9, 2010. [Online; accessed 10-October-2017].

[2] SENSORCOMM 2015. Integrated smart glove for hand motion monitoring. *https://www.thinkmind.org/download.php?articleid=sensorcomm_2015_2_50_10108*, 2015. [Online; accessed 10-October-2017].

[3] Notch. Notch: Smart motion capture. *https://wearnotch.com/*, 2017.

[4] OptiTrack. Motion capture systems. *http://optitrack.com/*, 2017.

[5] Moov. Moov fitness coach. *https://welcome.moov.cc/*, 2017.

[6] Starlino. A guide to using imu (accelerometer and gyroscope devices) in embedded applications. *http://www.starlino.com/imu_guide.html*, December 29, 2009. [Online; accessed 12-October-2017].

[7] Diego Emilio Serrano. Design and analysis of mems accelerometers. *http://ieee-sensors2013.org/sites/ieee-sensors2013.org/files/Serrano_Accels.pdf*, Novermber 3, 2013. [Online; accessed 12-October-2017].

[8] Diego Emilio Serrano. Design and analysis of mems gyroscopes. *http://ieee-sensors2013.org/sites/ieee-sensors2013.org/files/Serrano_Slides_Gyros2.pdf*, November 3, 2013. [Online; accessed 12-October-2017].

[9] Brian Ray. Zigbee vs. bluetooth: A use case with range calculations. *https://www.link-labs.com/blog/zigbee-vs-bluetooth*, August 24, 2015. [Online; accessed 5-October-2017].

[10] Luca Ruggeri. Top 5 wireless ways to communicate with your controller. *https://www.open-electronics.org/top-5-wireless-ways-to-communicate-with-your-controller/*, March 3, 2015. [Online; accessed 5-October-2017].

[11] Naresh Gupta. *Inside Bluetooth Low Energy*. Boston : Artech House,, 2013.

[12] Mike Row. Introduction to the world of microcontrollers. *https://learn.mikroe.com/ebooks/picmicrocontrollersprogramminginassembly/front-matter/introduction-to-the-world-of-microcontrollers/*, 2016.

[13] Microchip. Li-po mcp73831/2 datasheet. *http://ww1.microchip.com/downloads/en/DeviceDoc/20001984g.pdf*. [Online; accessed 10-November-2017].

[14] Mike Grusin. Serial peripheral interface: Introduction. *https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi*. [Online; accessed 10-November-2017].

[15] SFUptownMaker. I2c tutorial. https://learn.sparkfun.com/tutorials/i2c. [Online; accessed 14-November-2017].

[16] TDK InvenSense. Icm-20948 datasheet. *http://www.invensense.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf*, 2017. [Online; accessed 15-October-2017].

[17] TDK InvenSense. Mpu-9250 datasheet. *http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf*, 2016. [Online; accessed 15-October-2017].

[18] Bosch Sensortec. Bmx055 datasheet. *https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMX055-DS000-02.pdf*, 2014. [Online; accessed 15-October-2017].

[19] Bosch Sensortec. Bno055 datasheet. *https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST_BNO055_DS000_14.pdf*, 2016. [Online; accessed 15-October-2017].

[20] Atmel. Atmega16u4/atmega32u4 datasheet. *http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf*. [Online; accessed 20-October-2017].

[21] Nordic Semi. nrf52832 - product specification v1.0. *http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.0.pdf*. [Online; accessed 20-October-2017].

[22] Texas Instruments. Tiva™ tm4c123gh6pm microcontroller datasheet. *http://www.ti.com/lit/ds/symlink/tm4c123gh6pm.pdf*. [Online; accessed 21-October-2017].

[23] Texas Instruments. Msp432p401r, msp432p401m simplelink™ mixed-signal microcontrollers datasheet. *http://www.ti.com/lit/ds/symlink/msp432p401r.pdf*. [Online; accessed 21-October-2017].

[24] Adafruit. bno055 absolute orientation sensor overview. *https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview*. [Online; accessed 20-November-2017].

[25] Adafruit. Feather nrf52 bluefruit. *https://www.adafruit.com/product/3406*. [Online; accessed 20-November-2017].

[26] Alessandro Filippeschi, Norbert Schmitz, Markus Miezal, Gabriele Bleser, Emanuele Ruffaldim, and Didier Stricker. Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion, 1 June 2017. [Online; accessed 29-October-2017].

[27] Mark Hughes. Capturing imu data with a bno055 absolute orientation sensor. *https://www.allaboutcircuits.com/projects/bosch-absolute-orientation-sensor-bno055/*, 22 March 2017. [Online; accessed 10-October-2017].

[28] Oracle. Code conventions for the java programming language. *http://www.oracle.com/technetwork/java/codeconvtoc-136057.html*. [Online; accessed 10-October-2017].

[29] IPC. Ipc-2612 sectional requirements for electronic diagramming document (schematic and logic descriptions. *http://www.ipc.org/toc/ipc-2612.pdf*. [Online; accessed 17-October-2017].

[30] IPC. Ipc-2223 sectional design standard for flexible printed boards. *http://standards.globalspec.com/std/1412637/ipc-2223*. [Online; accessed 17-October-2017].

[31] IPC. Ipc-7351b generic requirements for surface mount design and land pattern standards. *http://pcbget.ru/Files/Standarts/IPC_7351.pdf*. [Online; accessed 17-October-2017].

[32] IPC. Ipc-2615 printed board dimensions and tolerances. *http://www.ipc.org/TOC/IPC-2615.pdf*. [Online; accessed 05-October-2017].

[33] IPC. Ipc-d-325 documentation requirement for printed boards. *http://www.hytekaalborg.dk/files/indholdsfortegnelser/IPC-D-325A.pdf*. [Online; accessed 05-October-2017].

[34] IPC. Ipc-6011 generic performance specification for printed boards. *http://www.ipc.org/toc/ipc-6011.pdf*. [Online; accessed 05-October-2017].

[35] IPC. Ipc-6012 qualification and performance specification for rigid printed boards. *http://www.ipc.org/TOC/IPC-6012B.pdf*. [Online; accessed 05-October-2017].

[36] Rick LeBlanc. The importance of electronics recycling and e-waste. *https://www.thebalance.com/e-waste-and-the-importance-of-electronics-recycling-2877783*. [Online; accessed 10-October-2017].

[37] Holly Mangan. How addicting to electronics affects the environment and our lives. *https://www.moneycrashers.com/are-we-addicted-to-consumer-electronics/*. [Online; accessed 13-October-2017].

[38] Recreation and Sport Industry Statistical Group. The social aspect of sport and physical recreation. *https://www.ausport.gov.au/__data/assets/pdf_file/0006/276927/ABS-Social_impacts_of_sport.pdf*. [Online; accessed 12-October-2017].

[39] Robert W. Richardson. Ethical issues in physical therapy. *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4596180/*. [Online; accessed 12-October-2017].

[40] Center for Disease Control and Prevention. Physical activity and health. *https://www.cdc.gov/physicalactivity/basics/pa-health/index.htm*. [Online; accessed 09-October-2017].

[41] Deanna Dorman. Benefits of weight lifting every woman should know. *https://blog.paleohacks.com/benefits-of-weight-lifting-for-women/*. [Online; accessed 17-October-2017].

[42] American Physical Therapy Association. Who are physical therapists. *http://www.apta.org/AboutPTs/*. [Online; accessed 09-October-2017].

[43] Ben Gilbert. 10 reasons why physical therapy is beneficial. *https://www.burke.org/blog/2015/10/10-reasons-why-physical-therapy-is-beneficial/58*. [Online; accessed 09-October-2017].

[44] Andrew Dupree. How long does it take to manufacture a hardware product. *http://mindtribe.com/2016/08/how-long-does-it-take-to-manufacture-a-hardware-product/*. [Online; accessed 06-October-2017].

[45] Silicon Labs. Single chip usb to uart bridge, cp2104 datasheet. *https://www.silabs.com/documents/public/data-sheets/cp2104.pdf*. [Online; accessed 20-November-2017].

[46] Apple. Human interface guidelines. *https://developer.apple.com/ios/human-interface-guidelines/overview/iphone-x/*, 2017. [Online; accessed 20-October-2017].

[47] Apple. Swift - apple developer. *https://developer.apple.com/swift/*, 2017. [Online; accessed 15-October-2017].

[48] Chance Miller. Latest gartner data shows ios vs android battle shaping up much like mac vs windows. *https://9to5mac.com/2016/08/18/android-ios-smartphone-market-share/*. [Online; accessed 06-October-2017].

[49] Apple. About core bluetooth. *https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html*. [Online; accessed 21-October-2017].

[50] Adafruit. Adafruit unified bno055 driver. *https://github.com/adafruit/Adafruit_BNO055*, 2017. [Online; accessed 2-November-2017].

[51] Smart Draw. Uml diagram. *https://www.smartdraw.com/uml-diagram/*. [Online; accessed 16-October-2017].

[52] Fermitron. Electric manufacturing circuit design pcb orlando florida fermitron. *http://www.fermitron.com/*. [Online; accessed 06-October-2017].

[53] Develotech. Develotech. *http://www.develotech.net/*. [Online; accessed 06-October-2017].

[54] AndroidStudio. Android studio features. *https://developer.android.com/studio/features.html*. [Online; accessed 06-October-2017].

[55] Apple. About testing with xcode. *https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/testing_with_xcode/chapters/01-introduction.html*, 2017. [Online; accessed 15-October-2017].

[56] Apple. Instruments user guide. *https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/index.html*. [Online; accessed 18-October-2017].

[57] Inflectra. Testing methodologies. *https://www.inflectra.com/ideas/topic/testing-methodologies.aspx*, 4 June 2016. [Online; accessed 01-December-2017].

[58] Gregory Ciotti. The psychology of color in marketing and branding. *https://www.helpscout.net/blog/psychology-of-color/*. [Online; accessed 06-October-2017].