Wireless Applications of a Refactored Prosthesis

W.A.R.P.

COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Sponsored by:

EEL 4915: Senior Design II

Group 9

Daniel Mor	CpE	Dmor574@knights.ucf.edu
Niko Tubach	CpE	Ntubach@knights.ucf.edu
Brandon Ashley	CpE	Tbash@knights.ucf.edu

Dec 06, 2016

Team Members

Daniel Mor

Starting in 2011, Daniel has been studying Computer Engineering at The University of Central Florida. Actively involved with the Computer Science community, focusing on programming and Software Engineering with a background in robotics and embedded systems. He has gained experience by participating in various projects through student organizations such as: The UCF Men's Rowing Team, Robotics Club, American Society of Mechanical Engineers (ASME), American Institute of Aeronautics and Astronautics (AIAA), and Limbitless Solutions. He additionally served as Vice President of the Association for Computing Machinery at UCF (ACM) from May 2013 - May 2014 and then President from May 2014 to Jan 2015. Daniel has accepted a full time position with IBM Cloud Object Storage.

Niko Tubach

A Computer Engineering student at the University of Central Florida who began pursuing his degree in 2012. Niko gained a background in Software Development through his involvement with a multitude of group and personal projects, he brings this knowledge with him in order to positively impact his current work environment. He is currently working as a Software Engineer at the Lockheed Martin Missiles and Fire Control test center. Niko has accepted a full time position with Lockheed Martin Missiles and Fire Control.

Brandon Ashley

As a student at the University of Central Florida, Brandon has studied Computer Engineering with a focus in Software Engineering. Outside of UCF, Brandon has gained experience as a Software Engineer with Fresh Lines Web and Mobile Development. After a year working in the field of software development, Brandon was able to become well versed in server side implementation using open source technologies. Brandon has since moved on to a position as a Full Stack Developer developing a software as a service specializing in collaboration amongst music professionals.

Table of Contents

1. Executive Summary

Limbitless Solutions is a non-profit organization founded by Albert Manero in the Spring of 2014 which was created to deliver high quality, inexpensive bionic limbs for use by children in accordance with the ethos, “Nobody should profit from a child in need”. Since its inception the organization has matured and refined the art of creating low cost, 3D printed bionics. A multitude of teams work tirelessly on improving every facet of the design and manufacturing process in order to improve the quality of life for children in need.

This experiment started with the basic objective of assisting children by putting into practice engineering techniques taught at the University of Central Florida. Initially, Limbitless produced a bionic arm subsisting of various printed circuit boards interconnected in order to sense electrical impulses within the body. The onboard processor utilized these signals when triggering state changes of the hand, controlled by actuators. The system includes a power source, microcontroller, electromyography (EMG) sensor, electrodes, and a servo motor. Each board was assembled by hand and extremely sensitive to manufacturing defects and external noise. Although the system was enhanced with each iteration, a major overhaul was becoming evidently necessary to put an end to consistent problems. “The Ultimate Bionic Arm” (T.U.B.A.) was a Senior Design team commissioned in the Fall of 2015 to realize a single board solution, addressing many of the original issues. Although T.U.B.A. was largely successful, many technical improvements are still sought after.

The current ambition of the organization is to provide technicians with the capability to wirelessly transfer data and remotely update the microcontroller housed within the bionics. The purpose for including this feature is to streamline the process of providing firmware updates to end-users. Furthermore, a more practical implementation of T.U.B.A. is needed to provide optimal functionality with a limited power supply. Further research and development towards a practical software architecture, advanced hardware, and exceptional human-body interface are the technical challenges Limbitless Solutions aspires to overcome.

After brainstorming in a collaborative effort between the team members and the directors of Limbitless Solutions, the “Wireless Applications for a Refactored Prosthesis” or W.A.R.P. project was established in order to provide a solution to the challenges previously presented. The project will focus on redesigning previous systems to include wireless communication which will be used as a training utility for children in conjunction with the prosthetic device. This new module will double as a controller functioning similarly to the current devices in use, while also acting as a standalone peripheral.

The W.A.R.P. team aims to provide a working wireless architecture while also introducing a new suite of software applications which interface with the device via Bluetooth and the internet. A new printed circuit board will be designed, incorporating the best features of previous implementations and additional sensors which provide

the ability to recognize gestures, monitor power consumption, and provide a live debugging interface for developers. The final result being a fully integrated hardware and software stack which seamlessly work together to provide a fluid experience to users and technicians alike.

2. Background

2.1 Introduction

Limbitless Solutions is a non-profit organization which designs, manufactures, and donates bionic limbs for use by children at no cost to families. Started in the Spring semester of 2014 by a small collective of students at the University of Central Florida, Limbitless has gained exposure around the globe as a philanthropic organization which provides arms for children in need. Not only do these prosthetics provide functionality of an actual arm, but they additionally provide confidence to the children who use them. Many other positive developmental effects have been proven to be attributed to using a prosthetic and this organization aims to provide personalized, cost effective, and functional alternatives to expensive traditional prosthetics.

The bionics designed and produced by Limbitless Solutions have gone through many iterative improvements since its inception. The original arm which houses the electronics features three main 3D printed parts. The three main components include: an actual hand which is opened and closed by manipulating the tension on a cord, a housing for the electronics to hold all controllers and actuators, and lastly an interchangeable sleeve to cover the inner workings. Due to its modular design, each component can be easily redesigned for improved functionality or aesthetic appearance.

The efforts of Limbitless Solutions can be seen as with the story of Alex Pring. Alex lived most of his early life functioning without a right hand due to a birth defect. After being connected with Alex through a non-profit organization named “e-NABLE”, Albert Manero assembled an interdisciplinary team of engineers at UCF and started working on the first functional prototype. Prosthetics, especially those for children, can often cost families thousands of dollars each. They become an ongoing expense due to prosthetics needing to be replaced as the child grows out of them, similar to outgrowing a pair of shoes. As a prototype produced by the team, Alex’s arm cost less than $350 to manufacture, giving Alex a working right hand for the first time. Additionally, 3D printing technology allowed the arm to be customized for the most optimal size, weight, and custom aesthetics. Most importantly, this was all done using the same powerful technology which is found in other more expensive solutions.

Six-year old Alex received his bionic arm during a news conference on July 25, 2014 and became a media sensation overnight. While he previously disliked other kids repeatedly asking him what happened to his arm, today he happily shakes hands with curious observers. The arm he received allowed Alex to do things he previously wasn’t able to such as ride a bike, moreover he gained something much more valuable - confidence. Alex, his family, and Limbitless Solutions were approached by media from around the world to tell their story. This was just the beginning, as more families heard of their success they contacted the organization in hopes of getting a bionic arm for their loved ones.

As Limbitless realized the potential of their creation, they participated in the Collective Project, a call for revolutionary ideas set forth by Microsoft. Limbitless and a few other hand selected organizations were chosen to have their story broadcasted across the globe through a series of commercials. Through a partnership with Disney's Marvel, a new bionic arm, shown by Figure 1, was designed based off of Iron Man's character portrayed by Robert Downey Jr. In the commercial, Robert presents Alex with the newly crafted arm while in character. The video went viral and in the following month the Limbitless team was flooded with requests for bionics by families who also faced the plight of prosthetics being overtly expensive. This popularity brought Limbitless Solutions into the limelight once again and lead to many strategic partnerships with companies around the world.

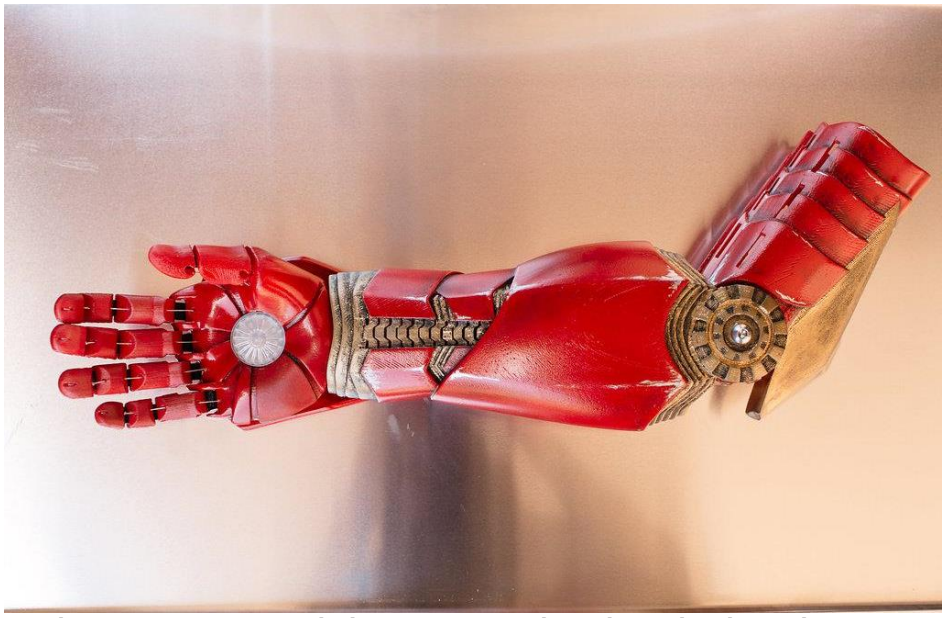


Figure 1: Iron Man styled arm presented to Alex Pring by Robert Downey Jr. (Reprinted with permission from Limbitless Solutions)

Through collaboration with UCF, Limbitless was provided with special access to the university's machine shop located in the engineering building and space to further develop future prototypes. As Limbitless continued to grow, the university prepared office space on the third floor of the Harris Engineering Corporation (HEC) building to operate out of. This access allowed the organization to flourish and expand design and production at an accelerated rate. Students participate in the organization as volunteers or interns in order to gain experience while working towards furthering the goal of helping children. Electrical and Computer Engineering students continue active development of the electronics and software. Mechanical Engineering students volunteer to design better enclosures and mechanics for the bionics, while artists paint them to complement the child's personality and preferences.

The year 2015 proved to be filled with acknowledgement of Limbitless Solutions' accomplishments, receiving several awards and the opportunity to speak

at multiple engagements. One of which being the “Michelle Akers Award”, presented by the UCF Alumni Association, an award given to individuals whose endeavors bring positive attention to UCF. This was awarded due to the team’s ability to bring UCF to global recognition via social media, speaking at conferences such as TED, and becoming involved with various talk shows and news features.

Additionally, the Limbitless Solutions team was able to secure other awards. Presented by Governor Rick Scott and Volunteer Florida, the “Champion of Service Award” and Orlando’s News 6 “Getting Results”. 2015 was also the year Albert Manero was inducted into the Order of Pegasus, the highest honor a student can receive from UCF. In the following year of 2016, Limbitless Solutions has been honored with Orlando Business Journal's IQ Award. The recognition from these awards can be attributed to the dedication and hard work shown by the engineers at Limbitless and the endless potential the organization has for the future.

With a collection of accolades and partnerships under its belt, Limbitless has a bright future ahead of it. Multiple engineering teams work on the many challenges involved in designing better bionics. Future iterations will address common problems, while increasing the capabilities of the technology. In the future, Limbitless aims to mass produce these bionics in order to help a wider range of people. The organization is rapidly expanding as it tackles new and challenging problems such as building smaller arms, elbows, legs and various other biomedical solutions.

2.2 Previous Iterations

2.2.1 The Original

As previously mentioned, the first version of the bionic arm developed by Limbitless Solutions was given to Alex Pring. The designs have since been open sourced and released online under the Creative Commons License along with instructions for its reproduction (http://www.thingiverse.com/thing:408641). The realistic hand was created by Steve Wood, who published the design for his “Flexy-Hand” online and made available to the team. The hand utilizes a uniquely flexible filament called “FilaFlex” which in this application acts as a type of elastic joint between individual fingers. FilaFlex is used similarly to everyday ABS or PLA plastic commonly used by 3D printers. A Kevlar cord would be strategically inserted into each finger and pulled in order to close the hand. Simultaneously, the FilaFlex would apply an elastic force in the opposite direction to keep it actively open.

The device was responsible for resolving when to change the state of the hand, and control a single actuator in order to physically pull the Kevlar cord to do so. The electronics used were composed of prefabricated prototyping boards in order to achieve this desired functionality. Electromyography (EMG) sensors would be attached to the user’s residual limb by using a set of three

electrodes and adhesive pads. The signal is amplified, filtered and lastly processed in real time by an onboard Arduino based microcontroller (MCU) which uses this data in order to algorithmically decide how and when to initiate a state change. For this design to properly function, Alex would use a shoulder harness around his body to hold the weight of the arm while he uses it. Lastly, Figure 2 shows the 3D printed electronics enclosure, shaped as a forearm, that was designed and attached to the hand to complete the functional aesthetics of the bionic.

Figure 2: Photograph of the first version of Alex Pring's arm (Reprinted with permission from Limbitless Solutions)

2.2.2 An Improved Model

Along with the progression of the onboard electronics, Limbitless made the decision to personalize each arm in order to complement an individual recipient's interests. Through this junction, art meets engineering. In an effort to not only provide a functional prosthetic, but to likewise provide a way for the children to uniquely express themselves. The figure below showcases three of the popular designs which have been donated by Limbitless Solutions.

Figure 3: Artistic sleeves for bionic arms
From bottom to top, arms styled in the themes of the movies Transformers & Iron Man, and a custom abstract design
(Reprinted with permission from Limbitless Solutions)

The incredibly eye catching aesthetics shown in Figure 3 are what most users remember, but the functionality which is of utmost importance to the user comes from the electronics housed underneath the aesthetic sleeve. The Limbitless team upgraded several of the internal components in order to extend battery life and improve the torque output from the servo motors, resulting in a stronger and faster grip. The board which acts as the control center for the entire arm was remodeled to shrink the form factor in favor of a decreased device weight and to make room for improved servo motors and batteries.

At this point, the practice of layering prefabricated prototyping boards is still being done as was in the original implementation. The team became better at manufacturing them, lowering the total cost further while improving on the build quality and lead time to delivery. Many of these upgraded electronics served their purpose and were subsequently included in the delivery of new bionic arms donated by Limbitless. The organization was aiming to mass produce these boards and a new design was needed to allow autonomous manufacturing on a larger and more consistent scale. Figure 4 shows the layered PCB design originally used by Limbitless. Figure 7 on the other hand, shows the progression of design in the mechanical arm from the original to the current version.

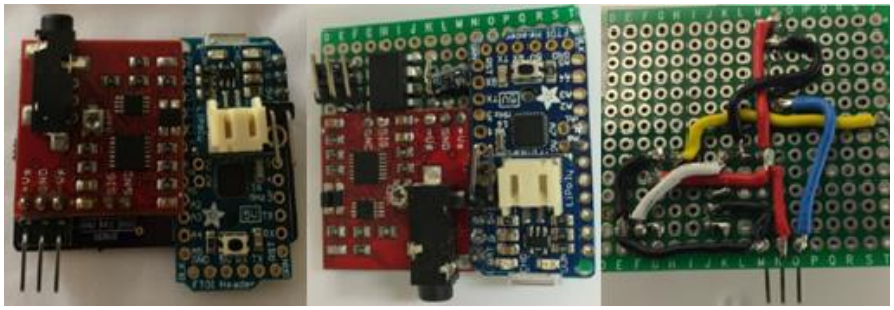


Figure 4: Image of the layered PCB from the improved design (Reprinted with permission from Limbitless Solutions)

2.2.3 Current Design

To address the major problems described, a new team of engineers was formed in order to produce a single board solution, Figure 5. This new design had the objective of including all previous functionality while allowing development to move away from relying on third party prototyping boards. A significant portion of the original cost when constructing a working board was comprised of the EMG sensor purchased from Advancer Technologies. In response, the team used open source schematics to integrate the sensor into their solution. Furthermore, rather than use an Arduino based Adafruit Pro Trinket, as was used in the previous design, the team opted to embed an Atmel Atmega328P microcontroller. This is the same chip used in the original Arduino design and allowed the team to directly port the original code to the new chipset.

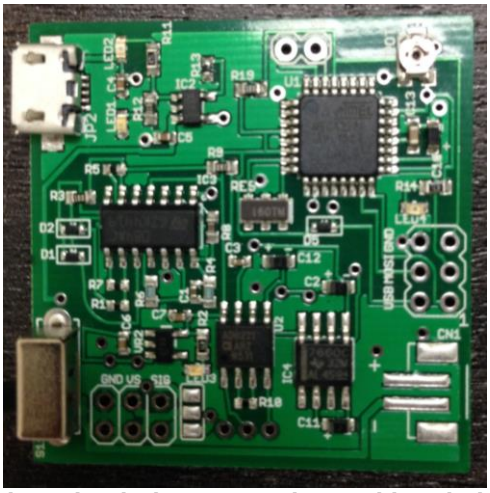


Figure 5: Single board solution currently used by Limbitless Solutions (Reprinted with permission from Limbitless Solutions)

2.2.4 T.U.B.A.

"The Ultimate Bionic Arm" (T.U.B.A.) was a Senior Design project whose team was commissioned in the Fall of 2015 by Limbitless Solutions to realize a more advanced single board solution. The team aimed to conduct research and development on more exotic features, while also solving minor design issues found in previous iterations. This board, Figure 6, is capable of but not limited to: controlling a large number of actuators simultaneously (used to form hand gestures) and haptic feedback which simulates a central nervous system by providing a vibrating sensation on the wearer's residual limb. This team also had other features which were planned as stretch goals, but never successfully completed due to time constraints. The team originally planned on implementing a Bluetooth interface which would be used as a medium for wirelessly configuring and reprogramming the microcontroller. This would have allowed the electronics to be securely sealed to prevent tampering from outside sources. The team successfully completed their project in the Spring of 2016 and called upon future Senior Design teams to implement the features which they were not able to.

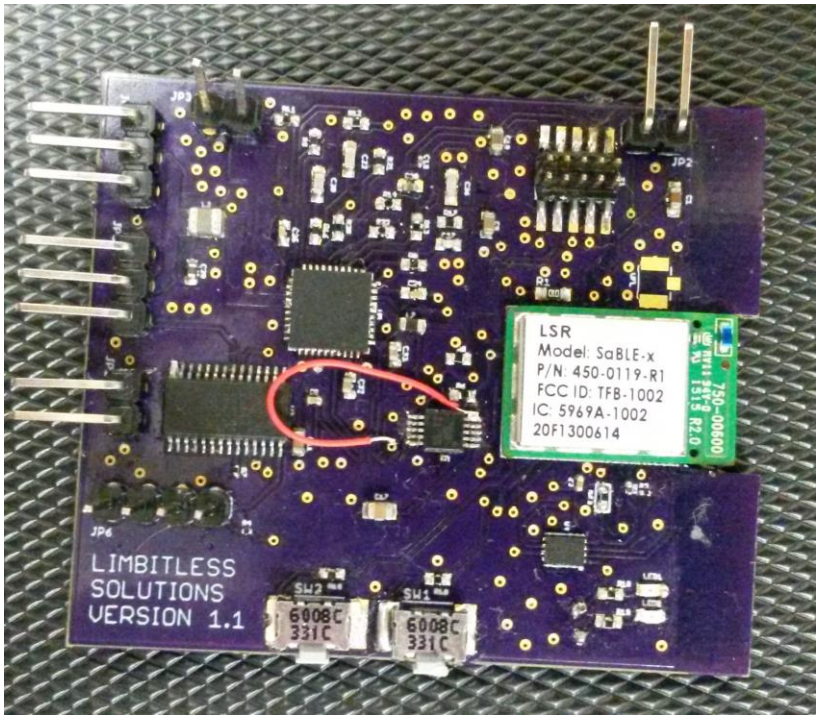


Figure 6: Printed Circuit Board for The Ultimate Bionic Arm (T.U.B.A.)
(Reprinted with permission from Limbitless Solutions)

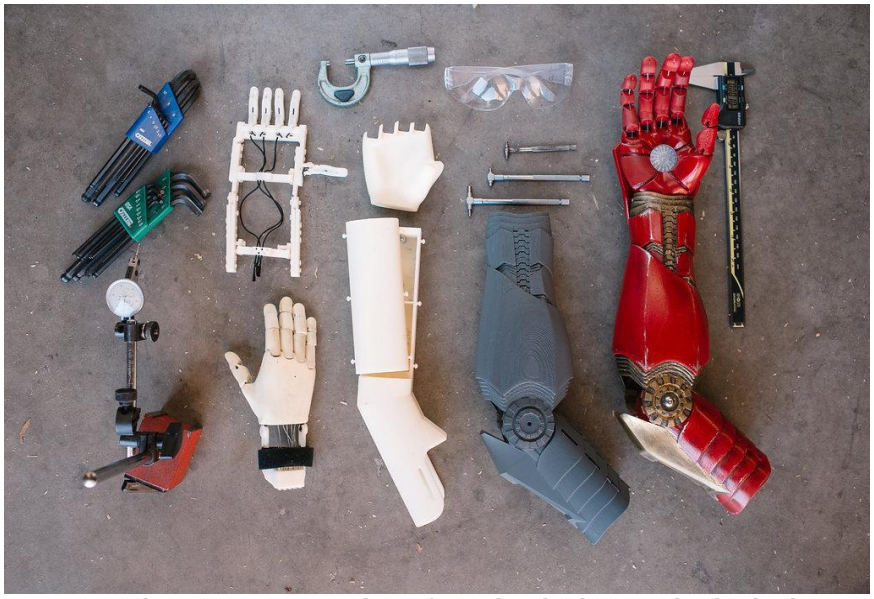


Figure 7: Progression of mechanical & aesthetic designs (Reprinted with permission from Limbitless Solutions)

Additional credit goes to KT Crabb of Kt Crabb Photography, Tyler Pierce of Soaring Wings Media and Alyssa Marie of Alyssa Marie Art & Photography for the media published pictures of Limbitless Solutions products shown in this section.

3. Project Description

This section describes the motivations fueling the project, as well as the outlined goals and objectives which W.A.R.P. aims to complete by the end of Senior Design II in December 2016. In addition, hardware and software requirements will be introduced and explained to set unambiguous and functional goals which will be referenced throughout the design process. The following section will also include various non-mandatory requirements or “stretch goals” that will be attempted. Realistically, due to time constraints, these additional requirements may not effectively make it to the final design. In this case, these goals may become suggestions for future teams to look into.

3.1 Motivations and Goals

The original inspiration for this team’s requirements originated from an expressed concern from both Limbitless engineers and customers alike who found the modern calibration mechanism inconvenient. Throughout a typical day, a user will perspire which causes the resistance of their skin to dramatically change. Since EMG sensors heavily rely on consistent working parameters, a threshold needs to be configured to ensure the correct behavior. Currently, the user will secure three EMG electrodes to their residual limb and manually configure a thresholding variable directly in code. This would not only require re-uploading the code to the microcontroller, but also require the user to carry a computer with them at all times. Thus the idea for a wirelessly enabled device was born, as the natural progression of the technology.

Due to the advent of modern smartphones, most people unknowingly carry powerful computers with them at all times. Many of these devices are capable of using multiple forms of wireless communication such as LTE, 3G, Wi-Fi, Bluetooth, and even NFC. The functionality of these devices can be expanded through the development of custom mobile applications which can be programmed to seamlessly network with the bionics without the need for a tethered connection. This idea evolved into creating a standalone peripheral which this paper will from now on reference as the “cuff”, “device”, or “board” which works independently of the current bionic arms.

The W.A.R.P. project aims to advance upon the technology in the existing prosthetics in use by Limbitless Solutions while also developing a new standalone peripheral. EMG electrodes would be placed on the bicep and interface with the PCB housed within a Velcro cuff. The cuff would be wrapped around the bicep and battery powered in order to provide a free range of motion to the wearer. Onboard sensor data would be processed and transmitted wirelessly via a wireless connection, optionally engaged, in real time to a nearby smartphone. A custom smart phone application would be developed to display the status of the module and manage received data in an intuitive user-friendly manner, providing a helpful tool for users, maintainers, and developers alike. This application would also be used to remotely

configure the electronics within the module without requiring it to be reprogrammed as is currently needed. An additional use would be to enable communication with a remote server to send diagnostic information and receive software updates to push to the hardware. This is particularly helpful to tailoring the calibration settings for children as they grow and their muscles strengthen. Increased use will require the calibration or thresholding to be adjusted and currently this requires a complex process only available at Limbitless Solutions headquarters.

This peripheral will serve multiple purposes, but first and foremost it will be used as a training platform. Once Limbitless approves the development of an arm, they would ship this product to the family in order to acquaint them with the technology. The children could then use the cuff as a type of “training wheel” before the actual arm arrives. The goal of the W.A.R.P. project would be aimed at providing a reliable and secure wireless connection, allowing for remote configuration of the hardware. Additional hardware would be added to permit features and control the power of the system in order to make the new module. Maintaining low power consumption, size, price, and weight are essential during the process of integrating additional features requested by Limbitless. Furthermore, the technology within the prosthetic should be reliable and easy to operate with limited technical knowledge.

In addition to enabling wireless communication, the team plans on redesigning how power is distributed throughout the board by using a more appropriate voltage regulator. The team will also integrate an accelerometer into the board in order to collect data regarding the device’s real world orientation and motion. This data will at some point be used for in conjunction with the EMG sensor for gesture recognition which is similarly done by devices such as the “Myo Armband”. W.A.R.P. plans on making this data readily available and capable of being streamed in real time to linked smartphones and analyzed in a central server for analysis. Future teams will then be able to utilize this information while developing the necessary gesture recognition algorithms.

In previous sections, it was mentioned that the wireless connection is necessary to enable the user to easily modify the software based threshold of the EMG sensor. As the previous designs actively poll the sensors for updated information, W.A.R.P. will modify the device to utilize hardware interrupts as a strategy of lowering power consumption. To realize this goal, it may be necessary to include additional hardware such as a digital potentiometer which can programmatically control a hardware based threshold. In addition to the major upgrades which were outlined, this device aims to be highly configurable through the mobile interface. Multicolor (RGB) light emitting diodes (LEDs) can set the mood and act as a niche aesthetic feature which will impress users. The mobile application will allow the selection of the color, while retaining the option to fully disable them in order to save power.

One thing to clarify is that this project will act as a standalone peripheral. The technology used within this device is based on previous designs with heavy

improvements/ and the features heavy improvements/ and has been heavily improved. Various components will be added to increase functionality and make it more powerful. But this device also aims to replace the printed circuit boards which are currently used in the bionic devices from Limbitless Solutions. This will generally be a stretch goal as time constraints may not permit many of these ideas to come to fruition, but will be a constant effort while designing this board. Furthermore, W.A.R.P. shall be designed to be used interchangeably as the PCB within the cuff and the board used within new bionics. The main reason for this goal, is that the wireless technology developed for the cuff is planned on being integrated in future devices regardless, and this would reduce research and development costs for Limbitless and help make this product available even sooner.

Overall, the W.A.R.P. project will improve existing technology produced by Limbitless Solutions by producing a standalone peripheral used as an educational tool. Lastly, this technology is planned to eventually be re-integrated and used in future designs and will attempt to include as many of the previous features as possible to make this transition easier.

3.2 Objectives

The primary objectives of the W.A.R.P. project are summarized as follows:

- Enable two-way wireless communication between the device and a smartphone
- Utilize the wireless link to push configuration changes from the smartphone to the device
- Redesign the power supply and regulators to provide a stable voltage and current to all on board peripherals
- Manufacture a compact printed circuit board which will fit in both the cuff module and existing bionic arms
- Develop a mobile application which will facilitate communication between the smartphone and device. This will further act as the control center for the device.
- Configure and deploy a central server which will be accessible over the internet for the custom smartphone applications to interface with
    - Design a database backend used to store relevant data
    - Develop a web API to be utilized by the smartphone in order to effectively interact with the server and database
    - Organize an administrative web page through the server for use by Limbitless developers to upload updated code
- Integrate new board into Velcro cuff for ease of use by wearer

3.3 Requirements Specifications

3.3.1 Hardware Specifications

The hardware design will involve components similar to those used in modern iterations of the technology in production by Limbitless Solutions due to the constraints imposed on the design by the organization's directors. As such the "cuff" module will remain under 0.5kg with the electronic components composing approximately half of the total weight. This constraint is imposed in order to make using the device easier for younger children. The module is intended to have an operating lifetime of at least 5 hours, but preferably up to 10 hours of active use before requiring to be charged. Furthermore, a maximum recharge time of 5 hours is required to permit the device to be fully charged overnight and better match up with a child's daily schedule. In order to fit the entirety of the electronics into the "cuff" the maximum dimension specifications are 100mm x 100mm x 25mm for the PCB only. Minimizing the size of the PCB will permit this technology to be utilized by more children and has a higher likelihood of being used in future iterations. The transmission of data to and from the module will take place using Bluetooth Low Energy (BLE), also known as Smart Bluetooth with a minimum transmission speed of 0.1Mbit/s. To ensure an efficient usage of available power to the cuff module the "slave" latency of 500ms utilized by BLE will be expected as an upper bound.

The printed circuit board (PCB) will additionally need to meet a minimum set of safety and legal requirements. Due to stringent FCC regulations the team intends to use a pre-approved wireless module, which includes an antenna. Since this device will be pre-approved, the team will be able to bypass the long and expensive process of trying to get a wireless device approved by the FCC. Using a pre-approved antenna also eliminates the need for designing an antenna and having to file for approval. Aside from needing these devices, the board will also need to include hardware to monitor the health of the battery and prevent dangerous failures of the system. General Hardware requirements are shown in Table 3-1.

<u>The specific hardware requirements are outlined as follows</u>:

- Interface with an FCC approved wireless module which is capable of two-way communication between the device and a smartphone in real time
- Power and control 1 - 2 RGB LEDs (brightness and color)
- Interface with an inertial measurement unit (IMU) and make this data available to the onboard microcontroller
- Allow the primary microcontroller to be wirelessly reprogrammed
    - If necessary, an external flash memory will be included in the design to store compiled programs as they are being transferred. Thusly the microcontroller should be able to read and write data to this memory

    - Additionally, this may require a secondary microcontroller to act as an ISP and control the reprogramming process in order to prevent data corruption
- Regulate power for the microcontroller, peripherals, sensors, and servos.
    - This may require a constant current source for the servos
    - This may require a buck converter
- Include a micro USB interface to allow optional tethered serial communication, reprogramming, and charging.

Hardware based stretch goals may include:

- Integrate EMG sensor circuit which is capable of interfacing with the EMG electrodes and being fed into the microcontroller's ADC for software based analysis
- Power and control 1 - 2 servo motors and incorporate a system to programmatically disable power to each as needed
- Enable hardware interrupts from analog EMG inputs to be triggered by using a comparator alongside a programmable digital potentiometer to control thresholding
- Monitor current and voltage from battery and USB (power consumption and battery status)
- Incorporate a weighted average software thresholding algorithm to decrease false-positive triggers due to noise

Table 3-1: General Hardware Requirements

Description	Specification
Cuff Module Weight	0.5kg
Price	Under $400 for the final design
Electronic Components Weight	0.25kg
Operating lifetime of product	8 - 10 hours
Recharge time of product	5 hours
PCB dimensions	100mm x 100mm x 25mm (Approximately 4in x 4in x 1in)
Minimum Data Transmission Speed	0.1 Mbit/s
Slave Latency	Maximum response time of 500ms

3.3.2 Mobile Application Specification

The mobile application will be made available to users who receive the completed product or any other compatible devices from Limbitless. Currently there is no integration with a remote device to provide feedback nor is there integration to allow remote calibration. The application will need an intuitive User Interface (UI) to allow for all individuals, adolescent or otherwise, easy accessibility to the tooling provided.

This application in conjunction with the device serve the purpose of assisting in the teaching of an individual to use the bionics. The application would be used as a feedback tool to visually see how the device interacts with the wearer. The user could make corrections and understand how it is working prior to receiving the actual prosthetic arm. Not to mention this can also be used a debugging tool for hardware and software engineers working on upgrades as previously described. The application will also act as intermediary between the central server and the device to exchange information between the two.

As the current devices lack a proper interface, real time calibration can be difficult to achieve. The application will act as a primary control center for each board and allow thresholding and control signals to be pushed in real time. As updates are received from the central server, the app will seamlessly transfer this data and signal the device to be reprogrammed on the fly without a tethered connection. Moreover, as the access point, push notifications can be sent to users to provide them with up to date information from Limbitless and facilitate communication with directors.

If time permits, the application will also act as an interface for a type of social network between the families who use the solutions provided by Limbitless. This will allow the children to meet and communicate with others who face similar challenges through a unified interface. Additionally, this will provide the organization a simple way to reach and interact with many families simultaneously.

<u>Mobile application based requirements are outlined as follows</u>:

- Develop Android and iOS mobile applications
- Receive push notifications containing information from Limbitless
- Configuration view which allows a user to modify settings
    - A graphic color picker to select the displayed LED color
    - Various fields to modify thresholding parameters and possibly select thresholding algorithm to be used
- View to display live data logged from the device
    - Logged data is sent to the server to be further analyzed
- Function to sync changes and settings to the device
- Receive software updates from the server and push them to the device

Mobile application stretch goals may include:

- Develop a social network to provide a unified interface through the mobile application for families to interact and communicate with each other
- Sync changes in real time as they are configured, rather than through a button press
- Enable secure wireless BLE bonding

3.3.3 Web Application Specification

The web application is intended for use by technicians and administrative users (admins). This is to provide feedback for all active systems in a single central location. Feedback is to include any useful information that can be gathered from logging the device's movement, power consumption, response time, or errors that have occurred. This interface can be used to manage the information sent and received from the various active mobile applications.

Web application based requirements are outlined as follows:

- Internet accessible web application
- Secured and only accessible by properly authorized administrators
- Provide the functionality to monitor device health and database entries for users who have the debugging feature enabled.
    - Charts, graphs and real time displays
- Accommodate an administrator to upload compiled source code to be transferred to the smartphone in order to initiate a remote reprogramming session.

3.3.4 Server Application Specification

The primary goal with this service is to provide a unified web API which will facilitate authentication between the smartphone and server in addition to a standardized method of transferring information between the two. The server application will interact directly with the database and act as an intermediary to ensure secured information is only accessible to authorized users. Furthermore, this application will act as a centralized location for all gathered data and provide a stable and more powerful computer to run analysis on this data if required.

Server application based requirements are outlined as follows:

- Provide functionality to authenticate users
- Ensure information is only modified or sent to authorized users

- Design database to store information securely and in an accessible manner
- Implement a software library to interface with the database
- Define a web API for external devices to interact with the server application
- Employ a mechanism to push notifications to users

3.3.5 House of Quality

The final goal for W.A.R.P. is to make the project a success in the eyes of the team's sponsors at Limbitless Solutions in addition to completing all requirements set forth by the Senior Design instructor. This device will prove to be an aid to the children that will be transitioning into using bionic limbs in the future. Additionally, this peripheral will attempt to test the viability of various features for possible use in future iterations of the electronics within the arm. With this in mind a House of Quality (HOQ) was created to visually display the balance of the priorities when marketing the project versus designing it. Different marketing goals require various tradeoffs when developing the technology shown in Table 3-2.

Table 3-2: House of Quality

▲▲ = Strong Positive Correlation ▲ = Weak Positive Correlation ▼▼ = Strong Negative Correlation ▼ = Weak Negative Correlation			Engineering Requirements			
			Size	Functionality	Power Consumption	Cost
			+	-	+	+
Marketing Requirements	Cost	+	▼	▲	▼	▲▲
	Aesthetics	+	▲		▲	▲
	Ease of Use	+	▲	▼	▲	▲

4. Research

4.1 Introduction

Before testing a single circuit or spending any of the available finances, the team will conduct research to get proper background knowledge on the problems defined in the project description section of this paper. This research will hopefully allow the team to properly formulate a solution or reconsider modifying the parameters of the project to accommodate the new information. Specifically, this section will include a concise selection of hardware components which are being considered or it may involve vetting the viability of components which have been previously requested to be utilized by the sponsoring organization. Furthermore, various software development techniques, languages, tools, and libraries will be described for the purpose of later using this information in the final selection of what will be incorporated in the design.

This project was originally started by Limbitless Solutions, continued by T.U.B.A. and succeeded by the W.A.R.P. team. There are many implicit constraints imposed upon the project many of which are not mandatory, but would build upon previously completed work. As such, many topics which would have been researched more thoroughly will be briefly discussed due to the fact that these topics may be covered in more detail in the paper written by the T.U.B.A. team. In an effort to prevent repeating the same content, this paper will avoid repeating these details and either reference the other paper, or implicitly expect the other paper to be referenced before this one.

A small number of components which were utilized by T.U.B.A. are the first to be considered and vetted since this previous team has already invested time in selecting these components. The primary goal of this project is not to copy the previous team, but improve what was produced in an effort to satisfy Limbitless Solutions, the supervising professor, and the team members themselves. With this being explained, a few of these components will be re-incorporated into the new design since this team agrees they are the best choice while considering price compared to capability of the component. Both of which are incredibly important to the sponsoring organization.

4.2 Existing Similar Projects and Products

As part of researching potential solutions which can be utilized in the design of this project the team looked toward referencing other products which use similar technologies or attempt to solve the same problems. This section will specifically avoid mentioning products produced by Limbitless Solutions as they were described in the

background section. This will also encourage the team to look toward foreign solutions which may involve techniques that had not been considered yet.

4.2.1 Myo Armband

This product is relatively new, released by Thalmic Labs in 2014. A few of the basic ideas for this project were inspired by the Myo Armband as it provides a solution to one of the larger engineering hurdles Limbitless faces. The problem, put simply, is that more complex controls require more inputs. Currently the devices are limited to one input provided through a set of three electrodes whereas using more inputs would become bulky and unmanageable.

The Myo Armband utilizes both dry EMG electrodes and having those electrodes work in conjunction with an accelerometer to recognize gestures. The armband includes eight EMG sensor channels, where each channel utilizes three dry electrodes which are packaged in a small elastic band which is seated on the wearer's bicep. The entire device weighs 93 grams, 0.45 inches thick, includes haptic feedback and is controlled by an ARM Cortex M4 Processor. The device utilizes medical grade stainless steel EMG sensors and accommodates a nine-axis inertial measurement unit (IMU) which includes a gyroscope, accelerometer, and magnetometer. The entire device is marketed at $169 at the time of writing this and communicates using Bluetooth.

The W.A.R.P. team strongly believes that incorporating this device or its technology into the bionics produced by Limbitless should be one of the next goals. As an intermediary, this project has taken inspiration from this device and plans to include an IMU and provide the hardware and software platform to begin development for gesture recognition to provide more inputs to the algorithms which control the bionics. As such, future iterations should look into the possibility of creating a custom version of the Myo Armband with dry EMG electrodes or possibly integrating the original product itself with the new bionics.

4.2.2 DEKA Arm System

The DEKA research and development firm created one of the world's most advanced bionic limbs. The original idea behind the arm was conceptualized in 2006 and after a $40 million grant, the first prototype for the system was produced in 2008. The arm's approval in 2014 by the FDA was complemented by the fact that over 90% of patients who used the arm were able to operate it and fully handle the fine motor skills it offered. The DEKA Arm System similarly uses electromyography and is capable of carrying out multiple simultaneous powered movements in order to perform six different user-

selectable grips. Additionally, force sensors allow the robotic hand to precisely control its grasp.

The DEKA Arm System can be seen as existing on the opposite end of the spectrum from the software and hardware produced by Limbitless Solutions as it is a highly expensive piece of equipment which was developed over the course of eight years with millions of dollars in funding. As such, comparisons to Limbitless and the W.A.R.P. project should be done carefully to consider these factors. Regardless, this arm is a prime example of the type of product the W.A.R.P. team would like Limbitless to produce one day and even achieve it at a fraction of the cost.

4.2.3 Bebionic Prosthetic Hand

Similar to the DEKA Arm System, Bebionic is one of the world's most advanced bionic hands. Independent motors operate each finger and provide 14 preset grips where the entire device is housed in a very small form factor. While including some state-of-the-art features with an attractive design, the device is still relatively expensive with a price tag of around $11,000. This product although impressive mostly involves mechanical design improvements. The electronics are impressive, but mostly due to their small form factor and would only be available at a high cost. Due to these considerations, the Bebionic hand is not actively being considered as this team is mostly focusing on electronics and software improvements.

4.2.4 John Hopkins Modular Prosthetic Limb

The modular prosthetic limb (MPL) was researched and designed by the John Hopkins Applied Physics Lab. This team additionally participated in DARPA's "Revolutionizing Prosthetics" program in 2006 alongside a team from DEKA. The MPL includes more than 100 sensors which are embedded in the arm and hand including but not limited to: absolute position sensors, contact sensors, torque sensors, joint temperature sensors, 3-axis accelerometers, 3-axis force sensors, incremental rotor position sensors, drive voltage sensors, and drive current sensors. This device provides a full realistic range of motion and is powered by pneumatics and hydraulics to yield a higher strength. As most advanced prosthetics, the device is controlled by EMG sensors in addition to an array of other sensors. Similar to the other bionics previously described, the MPL is a much more expensive piece of technology and is far too large and specialized. The W.A.R.P. team looks at this project for inspiration, but will not attempt to emulate it.

4.2.5 Touch Bionics i-Limb Line

The i-Limb line from Touch Bionics offers a variety of models of bionic arms, each controlled by electromyography and a few that are controlled by a smartphone application. Each offering is fairly similar with individual motors and sensors for each digit. The i-Limb quantum is set to have many gestures and interchangeable grips, the i-Limb revolution behaves closer to that of an actual hand, limiting the mobility to the natural bonds of a human hand, the i-Limb ultra takes the natural behavior of the i-Limb revolution and adds even more functionality than the i-Limb quantum with automated gripping positions and gestures to assist in daily tasks. From a software perspective, the W.A.R.P. team may include similar functionality as is seen from the mobile application used by Touch Bionics to control the prosthetics remotely.

4.3 Relevant Hardware

The hardware encompasses any physical component, mostly electronics which facilitate the operation of the bionic device. This includes the onboard microcontroller, wireless modules, power distribution system, printed circuit board, various discrete and active components and any sensors or integrated circuits which are included. Without the hardware, the software becomes unusable and vice versa. As such, both components are equally important when redesigning the electronics for the device. The following section will be subdivided into major systems, components or problems that are related to hardware and will include various points of research.

4.3.1 Wireless Communication

4.3.1.1 Wi-Fi

Allows devices to connect wirelessly to a remote network called a Local Area Network, or LAN, by utilizing the 2.4 gigahertz ultra-high frequency (UHF) primarily and 5 gigahertzes super high frequency (SHF) ISM radio bands. These radio bands are used by adapters integrated with an electronic device to transmit data packets to either a centralized transceiver (router) or another electronic device (known as an ad-hoc connection). The received packet data is then decoded and processed; if a router is the receiver it would transmit the information to a router through a hardline Ethernet connection which in turn would send it to a destination on the internet. The use of Wi-Fi as the wireless transmission medium can additionally be considered due to the proliferation of Wi-Fi as a standard for connectivity, allowing for a familiarity by end users with the technology. Additionally, with the creation of the WPA2 protocol, Wi-Fi is one of the most secure options for wireless connectivity. This being said, Wi-Fi does not allow for the ease of connection establishment that other Wireless LANs provide, especially if you wish to connect using the WPA2 protocol for optimal security. Thus, when taking these facts into account Wi-Fi seems like

a good alternative to a better solution in respect to W.A.R.P. for wireless connectivity.

4.3.1.2 Bluetooth

Similar to Wi-Fi, Bluetooth operates within the ISM radio band but only within the 2400 - 2483.5 MHz range. The W.A.R.P. project intends to utilize LSR's SaBLE-x Bluetooth Low Energy Module which is the same module used by T.U.B.A.'s v1.1 as requested by Limbitless Solutions. As previously described, the team was unsuccessful in getting a working version of Bluetooth and left this task to future teams to create a fully operational wireless board.

The SaBLE-x module is an "all-in-one" solution that allows for wireless communications as well as providing an integrated application processor. The module integrates a Texas Instruments CC2640 and provides a PCB trace antenna which is FCC approved, removing this responsibility from any development team. The CC26xx family of chips incorporates two ARM processors, where one is solely responsible for handling wireless transmissions and the second is capable of performing application specific operations. The module as a whole allows for up to Class 2 level transmissions which approximates a transmission of up to 10 meters.

LSR in partnership with Texas Instruments provide a plethora of hardware and software which greatly simplify the development for the SaBLE-x module. From TI, there is an extensive user manual which describes the operations and functions of the CC26xx family and a software development kit (SDK) with detailed documentation is also provided to properly develop firmware for the device. LSR provides development boards, 3D models, mobile applications to test wireless capability and various check lists to consider when creating a design using this chip. It is due to these many considerations, in addition to the request by Limbitless and T.U.B.A. to continue using this module that the W.A.R.P. team has decided to fulfil this request. Additional modules have been researched, the majority of these devices failed the various requirements set forth such as: minimal size, price, footprint, power consumption, small number of supporting components and a maximum wireless range. Lastly, but most important, it is imperative that any Bluetooth module selected for the project include an FCC approved antenna to minimize expenses for Limbitless.

4.3.2 Microcontroller

Since the SaBLE-x module contains an integrated Bluetooth module and microcontroller, this section will be an extension of the previous to further elaborate on the peripherals provided by the CC2640 which is embedded in the SaBLE-x. Other microcontrollers such as the Atmel Atmega, Attiny series, TI MSP430 series and other previously considered microcontrollers won't be elaborated upon in this paper to reduce redundancy as these were mentioned

in the T.U.B.A. paper. The previous versions of the arm used either the Arduino based Atmega328P or Adafruit Pro Trinket Attiny85 microcontroller which are both produced by Atmel. Due to the low power consumption characteristics provided by the Texas Instruments MSP430 line of microcontrollers, this was a primary contender for the central controller of the devices. Due to the integration of Bluetooth and microcontroller into one single chip, this consideration was dropped in favor of the SaBLE-x module which the W.A.R.P. team agrees with. For more information on the considerations of these chips, please reference T.U.B.A.'s paper.

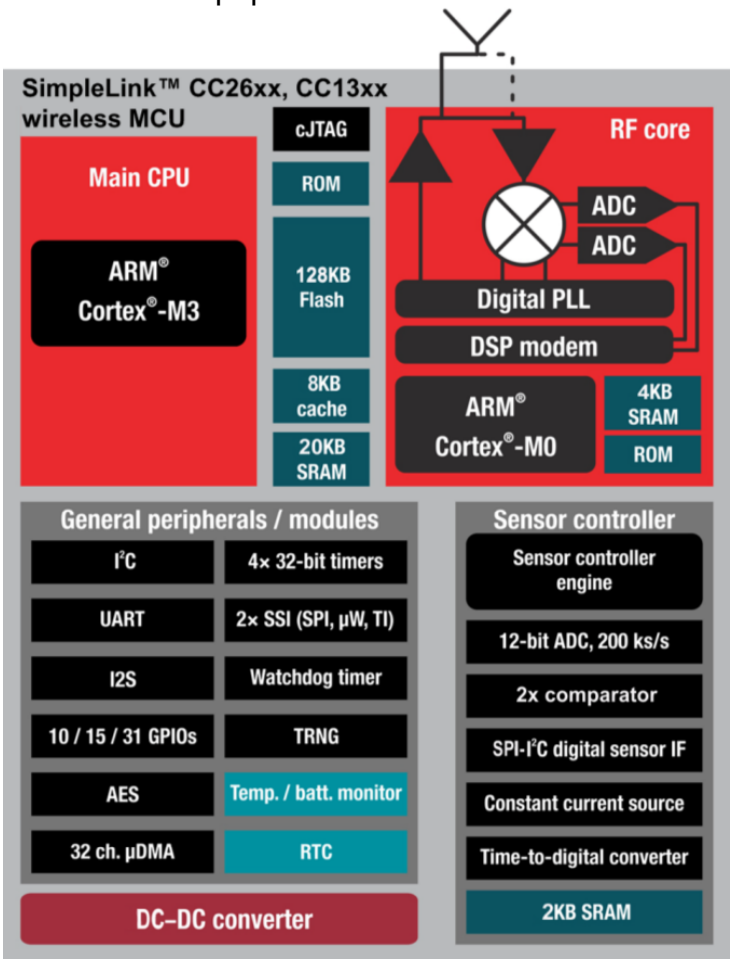


Figure 8: CC26xx Block Diagram
(Reprinted with permission pending from Texas Instruments)

Figure 8 above shows a block diagram which describes the inner workings of the CC2640 which is housed within the SaBLE-x module. This diagram is incredibly important when attempting to understand the peripherals inside the device and how they are organized in the overall system. Additionally, this diagram attempts to explain that the device contains two processors which handle different tasks. Figure 9 shows how efficient this processor is in terms of power consumption and describes the modes which can be programmatically selected in order to optimize power consumption.

Parameter	Test Conditions	Min	Typical Average Current				Max	Unit
			1.8V	3.0V	3.3V	3.8V		
Shutdown	No clocks running, no data retention				200			nA
Standby 1	With RTC, CPU, RAM and partial register retention. XOSC_LF				1.2			uA
Standby 2	With Cache, RTC, CPU, RAM and partial register retention. XOSC_LF				2.7			uA
Idle	Supply Systems and RAM powered.				550			uA
Active	Core running CoreMark				1.45mA + 31uA/MHz			
Radio Recieive			11.8	7.9	7.4	7.0		mA
Radio Transmit	+5 dBm output power		13.6	9.0	8.4	7.9		

Figure 9: SaBLE-x TX & RX Current Consumption Specifications (Reprinted with permission pending from LSR)

The most impressive aspect of the SaBLE-x is the sheer amount of peripherals which were incorporated into the small 11.6 mm x 17.9 mm x 2.3 mm package with 39 SMD pads. The device contains 8KB of 4-way set associative cache RAM, an 8 channel 12-bit ADC, capable of using SPI, I^2C, RTC, UART, contains a watchdog timer, temperature and supply voltage sensors, analog comparators, four 32-bit timers (or eight 16-bit timers) with PWM capability, and various other peripherals which are highlighted in Figure 9. The chip incorporates both an ARM Cortex-M0 processor dedicated for the RF core and an ARM Cortex-M3 processor for host applications.

The 32-bit ARM Cortex-M3 includes various programmable modes to facilitate low power as shown in Figure 9. A compact JTAG interface also reduces the number of pins required for debugging and interfacing with the chip. The device additionally provides the ability to utilize regular hardware interrupts and even nested interrupts which can be dynamically prioritized. As far as memory is concerned, the chip comes with 128KB of nonvolatile flash memory which is organized in sets of 4KB pages. In addition to all the other features, the chip handles security by including an Advanced Encryption Standard (AES) Engine with 128-bit key support with a low latency.

One of the original requirements of the project was to include a method for wired data transfer and communications. This is most likely going to be implemented through a micro or Nano USB connection and the design will need to include the hardware to facilitate this. The SaBLE-x includes an integrated RS-232C serial communications or UART which contains a transmitter and receiver which can be used to pass data back and forth through the physical USB connection.

The CC26xx contains up to 31 GPIO pins depending on the configuration with up to five 8-ma drive strength pins. Each I/O pin can be multiplexed to any digital peripheral through the I/O controller to provide high customizability. Nearly all GPIO pins contain pullup or pulldown resistors and can retain their state during all sleep modes. One of the stretch goals involves using the EMG sensor as a trigger for an interrupt through an analog comparator. The Sensor Controller which is built in to the microcontroller is capable of being operated in power-down mode in order to reduce power consumption. The sensor engine can read and monitor sensors in this state or perform these tasks autonomously and offload computation from the main CPU. This looks extremely promising for the reasons previously stated, as this module is capable of everything this project needs from a computational perspective and due to its integrated nature will require less space and passive parts to operate.

4.3.3 Sensors

4.3.3.1 Electromyography

EMG sensors permit the seamless interface with the wearer's body to control the bionic devices. All previous versions of Limbitless' bionics in addition to all of the more expensive prosthetics utilize EMG in some way, shape or form. The original arms used prefabricated versions of the schematic shown in Figure 10, while modern boards incorporate a similar version integrated with an Atmega328P microcontroller. T.U.B.A. specifically opted to using the MyoWare EMG board also sold by Advancer Technologies rather than integrate the circuitry into their project.

As the next step in furthering the designs produced by T.U.B.A. and Limbitless Solutions as a whole, the W.A.R.P. team plans on integrating a variation of this schematic, but modified for use by a single power supply rather than two as this schematic implies. This will allow the board to be powered by a single battery, while providing a negative voltage to the op-amps through a voltage inverter IC. Specifically, this schematic functions by using the AD8226 instrumentation amplifier to amplify the difference between the two raw signals coming from the muscle. This signal is then rectified using op-amps and two diodes and then further filtered and smoothed using two more op amps.

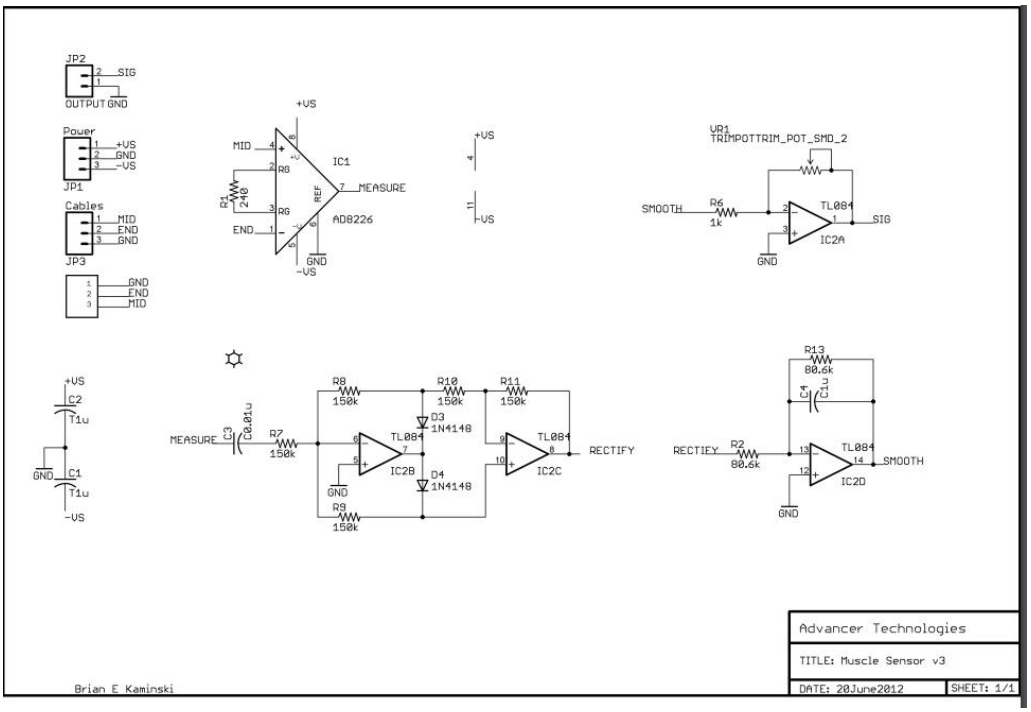


Figure 10: EMG Sensor Schematic
(Reprinted with permission pending from Advancer Technologies)

4.3.3.2 Inertial Measurement Unit

An Inertial Measurement Unit or IMU is usually a combination of accelerometers, gyroscopes, and sometimes magnetometers into one compact integrated circuit. Together these sensors can comfortably calculate orientation, position, and velocity. An accelerometer specifically is designed to measure acceleration in m/s^2 or in G-force (g). They can be used to sense static (gravity) and dynamic (sudden start/stops) accelerations. A gyroscope is capable of measuring angular velocity along the three axes of rotation as roll, pitch, and yaw (sometimes labeled x, y, and z). Lastly, a magnetometer measures magnetic fields. Since earth has a significant magnetic field, a magnetometer can be used as a compass to determine absolute orientation. Combined, these three sensors are often labeled as 9-axis IMUs.

As previously described, the main inspiration to include an IMU in the board was derived from the Myo Armband which incorporates both an accelerometer and gyroscope to track motion and recognize gestures from the wearer of the device. W.A.R.P. plans to include similar functionality or at the very least provide the hardware and software interfaces which will allow this functionality to be developed.

Various motion based sensors were examined with the main caveat that they must be I^2C compatible. The most popular chips seemed to be the ADXL345 3-axis accelerometer, HMC5883L 3-axis magnetometer, and the ITG-3200 MEMS 3-axis gyroscope. Individually, they were each readily

available at a low cost and provide a superior specification in regards to noise levels and SMD footprint. The main issue with these popular chips is that to get the full functionality, all three chips would need to be incorporated in the PCB and this would likely take up too much room. Due to this consideration, an effort was made to find an integrated solution.

In an effort to find an IC which better utilizes space requirements, the team came across the LSM6DS3 which is a 3-axis accelerometer and 3-axis gyroscope. The LSM9DS1 is similar to the LSM6DS3, but additionally includes a 3-axis magnetometer. Both of these devices look very promising with a small SMD footprint and relatively low pin count and low requirements for external passive components. The LSM6DS3 is 2.5 x 3 x 0.83 mm and consumes around 0.9 - 1.25 mA with an input voltage of 1.71 - 3.6v. The LSM9DS1 on the other hand is slightly larger at 3.5 x 3 x 1.0 mm and consumes around 1.9 mA with an input voltage of 1.9 - 3.6v. Both devices utilize are capable of interfacing with an I^2C bus, and both include internal sensors and embedded peripherals. The team believes that these two inertial measurement units are the best choices for the project and have purchased evaluation boards for each to ensure the viability of using them in the final design.

4.3.4 Memory Storage

4.3.4.1 Internal

With the usage of the TI CC26xx series MCUs, specifically the CC2640 within the SaBLE-x module and the CC2650 for alternative testing, there are defined internal memory components. The CC26xx series utilizes 128KB of flash memory as well as 8KB of static RAM (SRAM) that functions as cache memory with an additional 20KB of ultralow-leakage SRAM. The 20KB of SRAM is split into twin blocks of 4KB and twin blocks of 6KB all of which can be used for storage of data and execution of code. The data retention of these blocks can be configured individually such that they are enabled or disabled in order to minimize power consumption. The 8KB cache can be used as general purpose RAM if the flash is disabled. Additionally, the ROM embedded to the chip provides a pre-programmed TI-RTOS kernel, Driverlib and lower layer protocol stack software which functions as a BLE controller. This ROM also allows for reprogramming over SPI or UART by providing a bootloader. This in addition to the flash memory available, which is in-system programmable, allows for Over Air Download (OAD) through partial updates of the flash image. This means that the application code can be updated over the air, independently of the stack.

4.3.4.2 External

Memory residing outside of the MCU can be used to store programs to be flashed through use of the internal bootloader and may be used for transferring new programs to the MCU. The one caveat to external storage is

that programs cannot be run from it, only referenced and then copied to internal memory, thus any programs must still fit within the internal memory allowance. Candidates for consideration are EEPROM and flash external memory. Flash being a subsect of EEPROM which utilizes cheaper, less efficient NAND gates compared to its forbearer which uses NOR gates for quick propagation in exchange for cost. Due to the nature of future reprogramming within the scope of the W.A.R.P. project block-wise rewriting would be simpler as data transfer will be committed in a similar fashion over BLE transmissions. For the purpose of viability of inclusion towards the final W.A.R.P. design sample 1MB, I^2C compatible flash external memory devices will be tested.

4.3.5 Power

This board, with its many peripherals, sensors, and actuators will require a variety of input voltages and must be able to source and switch a relatively high amount of current. The SaBLE-x and many of the digital peripherals will need to be operated at a regulated 3.3v, while the servo motors will each require around 7v and peak at 1.5A each (at the most). This means that at its peak, the battery should be able to source nearly 3A and provide those two voltage levels and route them to the correct devices. Originally, this team planned on using a boost converter as working with a single cell battery is simpler than a multiple cell battery (which needs to be balanced). After research and consultation with colleagues, it was determined that the maximum current draw would be too much for a battery and would result in an unstable behavior. This behavior was tested as to boost a 3.7v battery to 7.4v and power a servo motor. The result was that as soon as the servo drew over a few hundred milliamps, the voltage from the regulator would drastically drop due to the extremely high current requirement of boosting the voltage.

Due to the problems stated above, the team plans on continuing the design used by T.U.B.A. and utilize a buck converter to attain the proper power levels. While similar to T.U.B.A., this project plans on redesigning the power subsystem to use a more practical DC-DC Buck Converter. The previous team used a TPS65257 which is far larger than necessary, whereas the majority of the features are not used and most of the pins are grounded or not connected. The team plans on using a simpler switching converter such as the MC33063AP which occupies less space while being fully utilized. The MC33063AP is capable of accommodating an input voltage between 3 - 40v and can output up to 1.5A with a controllable output voltage. The IC is also available in as a Dual Inline Package (DIP) as to make it easy to prototype on a breadboard and free to acquire a sample from the manufacturer.

The end goal of W.A.R.P. in addition to the previously mentioned requirements is to make the device programmatically configurable. One way to achieve this desired goal is to include the possibility to enable and disable

power to the servo motors at will through the use of GPIO pins. This can be implemented by connecting GPIO pins to the base or gate of a transistor to control the flow of current into the motor. This simple addition will make it possible for future mechanical teams to design a mechanism to physically lock the servo in place, where it would only draw power when moving. Although it at first seems like a simple solution, there are special considerations such as including one or two diodes to prevent reverse current from damaging circuitry as the servo motor is switched from the enabled position to disabled. A servo can often be modeled as a large inductor and may temporarily store energy and release it when the proper conditions arise. Other specific choices must be made such as whether to use a BJT or MOSFET to control the power. Both are capable of switching the servo at a high enough frequency, and it is possible to find either device with the proper voltage and current requirements. But it is most likely that the team will choose to work with a MOSFET due to their specialty at controlling higher current devices at a high frequency.

When necessary, a power control transistor will be used to enable or disable power to a component or subsystem to allow a granular control of the board as a whole. With the exception of servo motors which consume the majority of the power, the SaBLE-x module will be configured to sleep mode and utilize hardware interrupts when possible. This is different from previous iterations which simply poll the sensors and rarely if ever use sleep mode to reduce power consumption. A novel idea would be to use the EMG's analog signal to be fed into an analog comparator which is built into the microcontroller and compare it against a programmable analog signal which can then trigger an interrupt for the microcontroller. This can be achieved by using a digital potentiometer which can programmatically select a resistor value and be used to control the reference voltage fed into the analog comparator. When the EMG signal is above a threshold set by the digital comparator and the reference voltage, an interrupt would be triggered to wake up the MCU to continue its operations.

4.3.6 Enclosure

4.3.6.1 Cuff

The current concept for the cuff to hold the PCB to the user is an armband similar to one used to hold a cellphone. The cuff will use Velcro in order to provide a variable fit for different sized arms within the user base. In addition, the cuff will house the PCB within a similarly sized pocket which will most likely be elastic. This layout for housing the PCB will provide the lightest and easiest mode of keeping the board close to the EMG detection site of the user's arm. Additional security for the PCB that is being considered is a plastic-like case that will prevent the board from being smashed accidentally.

4.3.6.2 Aesthetics

One requirement is to include one to two RGB multicolor LEDs to be powered and controlled by the device. Each channel of an RGB led should require about 25 mA and the brightness or intensity could be controlled by a PWM signal. This would require about six PWM signals for the LEDs alone. This is in addition to the two other PWM signals needed for controlling the servo motors. The SaBLE-x module is capable of providing 8 PWM signals based off of 16-bit timers, but in the chance that there wouldn't be enough pins, a PWM driver or I/O expander such as the SX1509 could be used. At this point, the specific type of LED which will be used has not been decided. For simplicity, this version of the board may just include a built in SMD LED and include a ribbon connector for future version to allow the LED to be located in a different position (not on the PCB itself). It may be required to use a constant current source, depending on the type of LED. To keep the design or implementation simple, a lower power LED will be utilized.

4.4 Relevant Embedded Software

The embedded software includes any programming which is executed on the printed circuit board or specifically in this case on the SaBLE-x module or one of its coprocessors. This software will be responsible for controlling and signaling embedded peripherals and external modules using various internal registers and flags as specified in the data sheet. Furthermore, this low level software will bridge the hardware with the higher level software to facilitate the passing of information and control signals between each other.

When possible, the team will opt for using pre-existing software libraries such in order to reduce the workload and focus on the final product. The first consideration will be SDKs provided by the manufacturer since these are often the most reliable and contain an abundance of documentation and examples to work off of. The second consideration is third party open source and possibly closed source libraries

(depending on the usage license) with the expectations that they are actively being maintained and updated.

This project will specifically plan on using the tools provided by Texas Instruments such as Code Composer and a JTAG module to interface with, develop, and debug software for the SaBLE-x module. The team will also consider using the TI-RTOS (Real Time Operating System) which includes a real-time multitasking kernel and built in communication protocols which can be used in order to accelerate the development process.

4.4.1 Languages

The W.A.R.P. team is composed of Computer Engineering students with a strong background in both embedded and high level software development. Although the team feels confident that programming the chipset in the hardware assembly language is possible and may at some points be necessary, the goal is to stray away from using assembly language unless absolutely necessary. The reasoning for this as with most software is to avoid wasting time creating something that exists. Using a high level embedded language would ensure that there is sufficient documentation to reference. By using more popular languages, there is a higher probability that the team would be able to consult others with more experience.

4.4.1.1 C

When programming embedded systems, the C programming language is by far the most popular. It provides low level access to the memory which allows the software to set register values to control peripherals, timers, and GPIO pins. The language additionally provides constructs found in higher level languages such as custom data types, the ability to work with data structures easily, and most importantly it is consistently used within embedded systems around the world. Many of the libraries provided by TI and LSR are written in the C programming language or a variant of it and it may be difficult to work in another language and fully integrate these external libraries. Another possibility would be to use C++, which is an object oriented version of C. Providing all the best features of C, and allow the team to introduce OOP software design for simpler maintenance and readability.

4.4.1.2 Rust

In an effort to research new and upcoming languages for use in this project, the team discovered a programming language named “Rust”. It was designed to be a safe, concurrent, and practical language which supports pure-functional, imperative-procedural, and object-oriented styles of programming. It was created by a Mozilla employee around 2009 and announced in 2010.

Since this time, a compiler has been designed with the first stable version released on May 15, 2015. The syntax of the language itself is similar to C and C++, but with the major exception that it does not allow null pointers (dangling pointers). This prevents many errors which often appear in C++. The main considerations for the use of this language are first and foremost if the compilers are capable of supporting the SaBLE-x and inherently the CC26xx family's chipset and architecture. After much research, it appears that the primary concern, even if the compiler is compatible with the architecture, is the lack of supporting libraries and tools which are only available after a language has matured. More research will be conducted on the feasibility of using this language, but there doesn't seem to be enough resources to fully vet this language as it has only had a stable compiler available for less than a year.

4.4.2 Hardware Interface

The ARM Cortex-M3 processor will be hosting all the application software and logic which needs to be executing on the board. This will involve configuring the GPIO pins and interfacing with internal and external peripherals. This software will also communicate with the ARM Cortex-M0 coprocessor which acts as the RF core and physical layer of the BLE stack. The application software will be responsible for framing data to be sent out over Bluetooth to the mobile application and additionally verifying the contents of transmissions received over the wireless link.

The BLE stack which will be utilized in the SaBLE-x module has very specific hardware and software based timing requirements in order to properly function. Additionally, this system would require an increasingly complex inter-process communication mechanism to communicate information between the RF Core and application layer of the BLE stack. Rather than "rebuild the wheel" so to speak and start from scratch, the team looks toward the industry standards of using a real time operating system (RTOS) which will handle these complex scheduling issues.

Specifically, a RTOS includes a kernel, software services, and hardware drivers to encourage cross platform design and accelerated development. The kernel handles the allocation of memory, resource allocation, process scheduling, interrupt handling and does this all in real-time. The software services include various low level services such as task scheduling, semaphores (process synchronization), software interrupts, hardware interrupts, timers, and basic libraries for manipulating common data types. Lastly, the RTOS maintains hardware drivers which interface with integrated peripherals within the MCU. These drivers may involve low level functionality such as PWM, I^2C, I2S, SPI, GPIO, UART, and even USB.

Due to the sheer amount of information concerning each RTOS, this paper will avoid going in depth about various architectures. Since the SaBLE-x contains a TI CC26xx family chip, the simplest choice of RTOS would be the TI-RTOS produced by Texas Instruments themselves. This includes both generic and chip specific hardware drivers which can be utilized to achieve the desired functionality on this device. Furthermore, the TI-RTOS has no licensing fees and is completely free for development and distribution. TI-RTOS is also open-source under a BSD license and is programmed using the C programming language. Another major competitor and potential option would be the FreeRTOS which contains all major faculties necessary to bring this product to market without locking the embedded software to a single manufacturer that is TI. Although FreeRTOS is a very competitive product, it simply doesn't have the necessary support for TI chips built in and may require additional code to be produced.

4.4.3 Application Software

The application which is running on top of the RTOS will require a BLE stack software library. This code will assist in the abstraction of low level communications and adhere to the strict Bluetooth standards, allowing the device to properly communicate with other devices. The RTOS will have to be programmed in such a way to provide a high priority thread which will be responsible for transmitting and buffering data. The BLE side of the embedded software will also include a custom profile and services which will be advertised to any paired device. The application software will interface with the custom BLE profile and update shared attributes based on sensor data and incoming commands from the mobile host.

The SaBLE-x directly interfaces with both the BLE stack and the hardware on the PCB through hardware drivers provided by the RTOS. The most widely used driver will involve the I^2C protocol as the majority of chips on the board will utilize this communication method to reduce the number of GPIO pins required. Each embedded peripheral such as I^2C, UART and SPI will each require its own thread to execute concurrently and synchronously transmit and receive data. Additionally, each external peripheral such as sensors and external devices will also require a separate thread (or task) to allow the precise controlling of the priorities in the RTOS kernel.

The programming for this device will be much different than any other microcontroller such as AVR (Arduino), PIC, MSP430, or other smaller devices. The majority of the software will involve configuring peripheral and synchronizing the sharing of data between threads and then finally executing the scheduler of the RTOS. Many of these configurations can be done dynamically at run-time, but this runs the risk of wasting too much of the limited memory (both RAM and Flash). As such it appears that the best course of

action would be to configure all devices statically at compile-time to reduce this footprint. Although the device has 128KB Flash memory, it really isn't as much as it seems considering the overhead of the RTOS and other peripheral libraries. As such the almost definitely requires external memory (Flash or EEPROM), and at a higher capacity than previously thought as will be explained in the following section.

4.4.4 Wireless Reprogramming

Wireless reprogramming is an extremely complex task, especially with limited hardware. This task will involve having a compiled and verified software image transmitted from a central server to the mobile device through Wi-Fi or LTE and then again to the embedded device through BLE. There are many potential areas of failure during this process and it may not be guaranteed to always work due to transmission failures or packets becoming corrupted. As such, the team aims to ensure this feature will work the majority of the time as a proof of concept and may improve the consistency if time permits.

There are multiple methods to reprogramming the SaBLE-x module remotely. All methods involve transmitting the pre-compiled application image wirelessly over BLE and storing them in memory. One option is to store the new disk image within internal memory, but testing with relatively basic applications, it is expected that the final software will be between 70KB up to the maximum of 128KB of internal memory. This simply means that if the program will use over half of the memory, it will be difficult or impossible to store a secondary program internally without corrupting the original program before the re-flashing sequence even begins. This process can be made easier with access to proprietary software such as IAR Embedded Workbench, but the price tag excludes this option even if it would resolve all the memory based issues. Another choice would be to utilize an external Flash or EEPROM which contains at-least 128KB of memory, but preferably more than this to accommodate the additional software for managing the transfer. As such it appears than 256KB or more would be the optimal option for external memory. Due to the speed difference, flash memory using SPI would be optimal. This is in comparison to using slower EEPROM with the I^2C protocol (or SPI in this matter). With both of these options it would require the additional overhead of including a Boot Image Manager (BIM) on top of the BLE stack to manage the verification and validation of any new disk images. When the device is first powered on, similar to BIOS on a PC, the BIM will check the validity of both the internal and external image and use the best one. To initiate a reprogramming sequence, the application software will configure a "validity" bit in its own disk image to signal the BIM on the next boot sequence that the image is invalid. This will cause the BIM to copy the image from external memory - reflashing the device.

Other safer methods exist to reprogram the chip, but they require additional hardware. Rather than allow the chip to re-flash itself, the device could be monitored by an external secondary MCU. This secondary MCU could then hold the SaBLE-x pins at the correct voltage and signal level to trigger a proper reflashing as is done by an in-system programmer (ISP). The master microcontroller could then initiate the transfer sequence from the external flash memory to the SaBLE-x's JTAG pins. This process has many advantages since it'll lower the probability of corrupting the SaBLE-x and even if the software was corrupted, the secondary MCU can attempt another reflash without a problem. The complexity of this includes finding or building the correct software for this secondary MCU to act as an ISP in addition to facilitating the transfer of data between the external memory and SaBLE-x. Other complexities involve the synchronization of the two MCUs, and how will the reflashing sequence be initiated and/or signaled to be completed. Although not inherently difficult, this adds more thoughts to the design of the architecture and this is not to mention the additional footprint on the PCB of an additional MCU.

4.5 Relevant Server Software

The server side software will be running a cloud server, effectively an instance of a virtual machine operating on a remote server farm. The instances come in many forms by way of operating systems and distributions of those operating systems. The obvious operating system to use is Linux due to its tested reliability as it is the most frequently used server side operating system.

4.5.1 Host Providers

4.5.1.1 AWS Amazon

Amazon's Amazon Web Service (AWS) provides an entire suite services allowing the accomplishment of many common server side tasks. Some of these services have a "free tier" allowing the W.A.R.P. team to deploy server side software at little to no cost. Aside from being necessary in production, these services can be used during the testing and staging phases of software development lifecycle.

Amazon Elastic Compute Cloud (EC2) will be used to provide cloud server instances of the Ubuntu 14.04 LTS Operating System. During development and staging, this service can easily be used without any cost, once in production, the cost of this service depends on the usage. Considering this product's audience is a niche community, it may be possible to remain free for the first year provided the following is limits are met per month:

- 750 hours of EC2 running Linux, RHEL, or SLES t2. micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

The service provided by Amazon to store static assets, i.e., user uploaded files, is Amazon Simple Storage Service (S3). This service allows for any files uploaded to the server to be pushed to a dedicated storage system rather than using the server's local storage, doing so normalizes scaling and maintenance. Additionally, these assets can be cached to reduce S3 calls and improve download performance with the use of Amazon Cloudfront Content Delivery Network (CDN). The per month constraints for the free tier of S3 and Cloudfront are as follows:

- Cloudfront - 50 GB Data Transfer Out, 2,000,000 HTTP and HTTPS Requests
- S3 - 5 GB of Standard Storage
- S3 - 20,000 GET Requests
- S3 - 2,000 PUT Requests

Amazon offers a service for push notifications called Simple Notification Service, this provides a layer between the native push notification queuing service and the server sending notifications. This abstraction grants developers minimal effort to send notifications to multiple device types, e.g., Android devices and Apple iOS devices. The free tier can be maintained so long the following constraint is met per month:

- 1 million Amazon SNS requests

4.5.2 Cloud Operating System

4.5.2.1 Ubuntu

Of the server side Linux distributions, Ubuntu Server Long Term Support (LTS) is the most common. This is important as its popularity ensures vulnerabilities can be discovered in a minimal amount of time. Ubuntu guarantees five years after the release date, of support with security updates always remaining free of charge. The latest release is Ubuntu 16.04 LTS with support ending in late 2019. Additionally, this operating system has an image available for use with Amazon's cloud-computing platform.

This latest version of the Ubuntu Cloud image comes pre-installed with many packages that were not pre-installed in earlier versions. This allows for faster replication of the application in the event of the need for horizontal scaling. Another feature of the latest version is the ability to push kernel crash dumps containing log information about a given crash can be pushed to a remote server.

4.5.3 Security

4.5.3.1 Bcrypt

Passwords should never be saved to a database as plaintext, a method of encryption should be adopted to keep user information safe. The bcrypt function offers a slow password hashing implementation making brute force attacks computationally heavy. This function implements Blowfish encryption, a block cipher with symmetric-keys of varying size, with the addition of a "salt", data randomly generated per password concatenated to the end before encryption. Adding the "salt" prevents the running of an algorithm against a password lookup table to discover the encryption key.

Though bcrypt is currently very slow on modern computers, as hardware improves it could become possible to execute attacks in a reasonable amount of time. Further functionality of bcrypt includes adaptation, to prevent newer computers from attacking by increasing the cost to a higher threshold. Considering the ability of reconfiguring the salt as the power of computation increases, it is unlikely this will be replaced in the near future. Below is an example of the password digest stored in the table of the user needing later authentication.

"$2a$10$sduvZ25FV8Frko7cVNVWBu2bY2mfsSEuCgSnciEn6DDPNUExvvfoS"

4.5.3.2 JWTs with HS256

Data will be passed using the Hypertext Transfer Protocol Secure (HTTPS) in the form of JavaScript Object Notation (JSON). Each request will

need to be checked with the server after email-password authentication to ensure the requested action is authorized. The headers of an HTTP request can be used to bear a token used for subsequent authenticated requests. This token represents a hash of encrypted data identifying a requestor, using the same notion used to pass data in the body of a request.

The tokens the W.A.R.P. project will be using for request authentication are formally known as JSON Web Tokens (JWTs), this is an industry standard method for making secure claims between individual entities (i.e. server and client). The JWTs take the form of a header, payload and the token signature delimited by ".". The header contains information specifying the algorithm used to encrypt the token. The payload contains claims made by the requestor in the form of minimal user information and an expiration timestamp. The signature is generated via a secret key kept on the server side as an environment variable, the header and the payload. The token is signed by HMAC, hash message authentication code, and SHA256, secure hash algorithm computed with 32-bit words, using the server application's secret key. Below is the general structure of this token packed into the headers of a given request for some protected route.

```
Authorization: Bearer xxxxx.yyyyy.zzzzzzz
```

After the initial user authentication, the token is generated and passed back to the client side application in the response headers. This token is then stored in the client's local storage to be passed back to the server side and decoded. During the decoding process, the token goes through a number of validity checks to ensure it has not been altered nor has it expired. After a token has been validated, it can then be parsed and the requestor's claims can be assessed to determine if the request is authorized.

4.5.3.3 SSL/TLS Certification

The Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS), will be set in place to ensure clients connecting to the server via HTTP are transporting information securely. SSL/TLS acts as an encrypted wrapper around the web traffic, protecting against the interception of malicious individuals. Using this protocol requires the initial generation and constant renewal of certifications through a Certification Authority, the renewal process can be automated with the use of a cron job to execute a script generating a new certificate every sixty days. After the creation of the certificate pairs, the web server must be configured use HTTPS.

4.5.3.4 Only Public-key Authentication

Establishing a remote connection with between a development machine and a production is essential to maintenance and released deployments. Secure Shell (SSH) can provide a way of authentication via public-key cryptography, cryptography using asymmetric public private key pairs each one

validating there has been no manipulation of the other via comparison of a digest computed by the private key and a digest derived from the decrypting of a signature originally encrypted by the corresponding private key. Another way to gain access to a remote server is to request a password after ssh attempt, this can be susceptible to “man in the middle” attacks, as such this should be disabled after a production server’s inception.

4.5.4 Database

4.5.4.1 PostgreSQL

Object-relational databases provide a way to persist data created, updated, or removed throughout the application’s lifecycle while having a schema capable of supporting object-oriented paradigms directly. PostgreSQL provides many features in the form of extensions and data types not available in other relational databases. Multi-Version Concurrency Control (MVCC) frees the database from relying on mutex locks to ensure atomicity, consistency, isolation, and durability by providing a state of the database not visible to other transactions until changes have been committed.

Additional extensions can be enabled or supported through third party libraries. PostGIS, a PostgreSQL extension for a Geographical Information System, allows the creation of a spatial data type column with support for geography allowing efficient querying for location based analytics. The “citext” data type is a type of string or text that is case-insensitive, making it ideal for storing items that must guarantee uniqueness such email addresses used and indexed in a user table. UUIDs, Universally Unique Identifier, can be used as a column type, providing a long and guaranteed unique key, possibly used in place of the default primary key. JSON data types are also supported by PostgreSQL, this allows the creating and updating of hashes or key-value pairs. This can be used to help support multiple device authentication for a single user with many authentication tokens.

JSON data type as a column allows for multiple objects to be hashed as key value sets into one attribute of a model. Implementation of a technique suitable for storing a limited number of tokens would involve keeping track or sorting for the least recently created token and removing after the generation of new token. To make all tokens easily accessible to the function used for user authentication after decoding the JWT found in a HTTPS request, the randomly generated token is to be set to the key with a value of an expiration hash. After candidate user has been identified with the user information passed in the payload of the JWT, a single check for the existence of the claimed token in the set of Authentication Tokens attribute attached to a user. Below is a possible token stored in the map of tokens.

```
{"cef9690aefaa4bd6b148bd08f36fac2a"=>{"created at"=>1467923885}}
```

While this implementation proved to be secure, the complexity behind was somewhat unnecessary. The final authentication/authorization specification still protected against playback attacks and allowed multiple tokens to be generated, yet these tokens resided in their own table and included the following fields: jti, typ, aud, iss, sub, exp, jwt, and claims. With the jti and aud being used as primary keys. Using the exp column as an expiration field and configuring a background process to sweep every 120 minutes for expired tokens.

4.5.5 Languages, Libraries and Frameworks

4.5.5.1 Ruby

Ruby is an open sourced, object oriented programming language originally developed in the mid-1990s by Yukihiro Matsumoto. Ruby features a dynamic type system allowing for easy variable instantiation. Additionally, there are no primitive data types with the allowance of every instance of any type to be treated like an object. These features allow for metaprogramming, in Ruby this can be defined as manipulating class or instance methods at runtime, to assist it making easily readable and maintainable code.

Ruby has one the largest and most motivated communities, this is important to ensure continued support. This community has provided open source libraries and documentation to aid in overcoming any trouble that is not easily recognizable by a person who has not encountered a trouble of some kind. Included in the mass of available Ruby libraries lie some very notable frameworks allowing for rapid development and easily implementable testing suites.

4.5.5.2 Ruby on Rails

Among Ruby frameworks, Ruby on Rails (RoR) is certainly the most popular. RoR was released in 2004 as an open sourced Model View Controller (MVC) framework authored by David Heinemeier Hanson, and since, over 3,000 other contributors. MVC can be defined, in the instance of RoR, as models built as classes directly referencing database tables, controllers acting as a way to perform actions manipulating and retrieving the stored data exposed via Unified Resource Identifiers (URIs), and views which simply display data.

In the scope of the W.A.R.P. project, the view piece of the RoR framework will be served in the form of serialized JSON. This data can then be easily consumable by any of the client side applications in a humanly digestible way. Leaving out the view of this framework additionally allows for performance gains after initial application load, this involves leaving out what is known as

the *Asset Pipeline,* the RoR way of shipping Cascading Style Sheets (CSS), JavaScript, and HTML per client request, thus speeding up request times. The CSS, JavaScript, and HTML can instead be resolved during the initial browser request for the web application.

4.5.5.3 Rust

As previously discussed, Rust offers an extremely performant infrastructure to build upon. While the consideration for its use in the embedded application is yet to be determined, Rust is certainly a candidate for server side implementation involving current heavy tasks. Current support for creating Ruby calls to Rust services exists and is well documented. Rust is considered to be pragmatically similar to Ruby, in that it offers microprogramming and macros, while these abstractions exist, they do not take a hit in performance over the C language. While producing the entire server side application in Rust could prove to be challenging, the primary development could be done using RoR and make bindings to Rust where necessary.

4.5.5.4 Elixir

Elixir's creator, José Valim, is the co-founder of a company specializing in the production of useful tools for the Ruby on Rails infrastructure and is a core contributor to the RoR project. The first appearance of Elixir was in 2014, it is built to run on top the concurrent, functional programming language, Erlang. While the Elixir language itself still remains in its infancy, Erlang has been around for three decades and has been responsible for the success of many large networking applications. The ideology of Elixir is very similar to that of Ruby, providing a rich, helpful, and actionable development environment via dynamic typing and extensible tooling. Aside from this, another similarity shared with Ruby is syntax as it has been heavily influenced by the Ruby community, yet approaching problems can differ as Elixir is a functional programming language.

Elixir may be a better fit for the W.A.R.P. project for many reasons, firstly considering the stretch goal of incorporating a social network for Limbitless' bionic limb recipients. This feature will require a potentially large network dependent upon the company's growth, thus having the need for a distributed system. Additionally, the feature involving real-time communication between the mobile application, used by the bionic limb recipients reading Bluetooth transmitted feedback, and the web/server application used to collect, process, and render the feedback can efficiently be produced with an Elixir Program. The Elixir ecosystem is rapidly expanding, surrounded by a community dedicated to shipping well-polished tools and libraries. Ruby will certainly have victory over Elixir in the number of repositories of plugins and add-ons but Elixir is Erlang/Open Telecom Platform (OTP) compatible as such Erlang's extensibility can be shared without any cost at runtime.

4.5.5.5 Phoenix

Ruby's influence in Elixir's design around a fantastic developer experience is as much as an influence of RoR around the design of the Phoenix framework. Phoenix has similar capabilities to that RoR in terms of providing an opinionated architecture for designing, building, testing, and maintaining applications for the both server rendered views and JSON serialized data, with increased performance. Both Phoenix and RoR currently support an out-of-the-box solution for websockets, a client-to-client streaming service, through the idea of channels, however, the Phoenix/Elixir implementation can handle far more concurrent connections per server thus lowering the cost of the load balancing distribution.

4.6 Relevant Client-Side Software

4.6.1 Platforms

4.6.1.1 Mobile - Android & iOS

The completion of the mobile application is the primary goal of client side development of the W.A.R.P. project. Android is one of the most popular mobile operating systems, produced my Google, the primary language used in development is Java. Java is a language the entire W.A.R.P. team is comfortable in, therefore achieving the full functionality described in the requirements will be easily obtainable.

Aside from targeting a single mobile platform, the team finds it prudent in securing applications in the two most popular platforms. iOS is Apple's mobile operating system with the primary language being Swift. Apple provides a very declarative approach to solving mobile application development with the Integrated Development Environment (IDE), Xcode. Xcode allows developers to drag and drop visual components into the current application's interface and generates functions to call to these visual models to be placed in the source.

4.6.1.2 Browser

The modern browser supports applications derived from many languages transpiled or compiled to JavaScript, CSS, and HTML. Browser development will take the form of web application development, the scope of this platform is to provide a panel for the individuals maintaining a technical relationship with the end user in providing updates to embedded software and configuration. Additionally, this application could serve as an administrative tool to monitor any exchange over the network in the event of the stretch goal's completion of adding a social network.

4.6.1.3 Desktop

Porting the web application design and functionality to a cross-platform desktop application will be completed in the best case scenario. The features are to expand slightly beyond that of the web application, allowing Bluetooth interaction between the bionic or cuff and the technician's computer. This can be useful for rapid iterations of testing the bionics' configuration over wireless transmission over a mobile device.

4.6.2 Languages, Libraries and Frameworks

4.6.2.1 Swift

Swift is the language used during the development of Apple software for both OS X, Apple's desktop operating system, and iOS. Swift's first appearance was in 2014, designed to replace the existing developer language at the time, Objective-C. This movement away from Objective-C allows for more performant code and an increase in type safety. While being a young language, Swift receives support from a large corporation and has been adopted by the community at large, currently on its third stable version.

4.6.2.2 Android SDK (Java/Go)

Google's mobile operating system, Android, has primarily been developed using the Android Software Development Kit (SDK) written in Java. As of recent, the highly performant Go programming language ships with native support for Android development. Golang offers an excellent concurrency model and an intuitive and simple to use Git based packaging system. Golang's type system is in the same discipline as that of the C programming language.

4.6.2.3 JavaScript

The technology around JavaScript has advanced significantly over the past few years, introducing paradigms and ideas commonly found in more powerful languages such as Scala. There are a few decisions to be made when beginning a JavaScript project, as there are currently many flavors of JavaScript in existence. The standards organization of which JavaScript is under is the European Computer Manufacturers Association (ECMA), overseeing syntactic decisions of the language. ECMAScript (ES) is the common reference to describing the language version followed by the year the version was finalized e.g. ES2015. Current design of the most recent finalized versions dictates to lead toward a more traditional pragmatic approach, featuring functional programming paradigms, instead of a raw scripting approach. Traditionally JavaScript features a dynamic type system, allowing the declaration of primitives with a single keyword and can be later reassigned another type, however many members of the community have attempted to add a static type system to the language. In the case of JavaScript, this can be considered a great addition as errors are not commonly discovered at runtime because the weak type system allows for silent failures.

Outside of ECMA, other organizations have released other revisions of JavaScript, most notably Microsoft's TypeScript. TypeScript offers many of the features of the latest versions from ES2015 with the addition of a type system and a compiler to compile the TypeScript to JavaScript. Any syntactic errors can now be resolved at compile time instead of runtime. Another approach to adding a static time system is known as Flow, developed by Facebook. Flow is simply a type checker, with a source primarily comprised of OCaml, and has no compile time and can also be used optionally. Making the static type optional allows for the gradual addition of type inference, simplifying the onboarding process for new developers as statically typed JavaScript is still a fairly new concept.

4.6.2.4 React

React is a view library published and maintained by a dedicated team at Facebook, providing a declarative way to display data dynamically using a component based architecture. React's views can be written as pure functions and simply receive data to display, process the data and render markup representing the data. All React components return markup, with a parent, normally holding state, calling the render () function inside of a class extending the React Component class, directly manipulating Document Object Model (DOM) elements. This type of functionality is made possible via JSX preprocessing, JSX is an XML-like (Extensible Markup Language) extension to the previously covered ECMAScript. This syntax extension grants developers the ability to pass attributes down a tree of view components to render data in a manageable way through a unidirectional data flow. Treating views as functions leads to higher testability including automated testing for client-side code. Example code for doing this can be viewed in Figure 11.

```
class App extends React.Component {
  render() {
    let {
      error,
      loading,
      clearError,
      children,
    } = this.props;

    return (
      <div className={styles.app}>
        <Nav
          title={"W.A.R.P."}
          className="App-header"
          logo={logo}
          className="App-logo"
          alt="logo" />
        {children}
      </div>
    );
  }
}

```

Figure 11: Example code featuring React syntax

4.6.2.5 *React-Native*

React-Native includes React and shares the same features as React-DOM but for native mobile applications. React-Native forms JavaScript bridges with native application code allowing functions to be delegated directly to Java or Objective-C/ Swift, depending on the mobile platform. The featured JSX syntax extension is still used but in a slightly different manner, making calls to the platform specific markup. Furthermore, access to the device's local storage is provided for the storing of authentication information and caching. An example can be seen in Figure 12.

```
class App extends React.Component {

  render() {
    return (
      <View style={styles.container}>
        <Image
          style={styles.imgWrapper}
          resizeMode={Image.resizeMode.contain}
          source={require('./src/img/warp-logo.png')}
        />
        <Text>
          WARP BT Scanner
        </Text>
        <Button
          style={styles.button}
          onPress={BleManager.checkState()}>
          Scan
        </Button>
      </View>
    );
  }
}
```

Figure 12: Example code featuring React-Native syntax

4.6.2.6 Application State Management with Redux

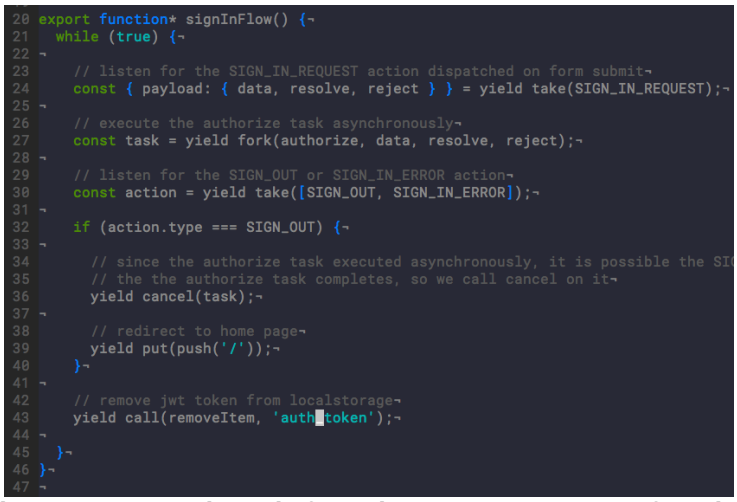
While React is capable of rendering an application's current state, managing said state in a global way is not included, only on a component level. Doing so should be done in such a way state and side-effects can be managed explicitly. Redux provides an architecture to structure an application's state in an immutable state tree, slicing substates as data structures to be duplicated and merged into a new structure representing the next state. Handling side-effects, variable changes outside of the state's scope such as asynchronous server requests, can be easily handled in one of two ways, with the use of Thunks or Sagas.

Thunks, in the scope of state management with Redux side-effect solutions, make heavy use of the ES2015's proposal of async/await and is therefore coupled to this while also being coupled to the dispatching actions being called. Sagas, however, are built using ES2016's (ES-Next) feature of generator functions, and have the ability to listen for a change in state and trigger calls without the need of actions to be dispatched to such functions. With Sagas, actions simply dispatch actions as normal, behaving as triggers in the form of action listeners and can be cancelled based on another action or set of actions. This decoupling, offered by Sagas, allows for side-effect testing outside of the state tree by simply asserting the action to be dispatched equals the expected action type and payload.

```
// The initial state of the App
const initialState = fromJS({
  loading: false,
  error: false,
  authenticated: getItem("auth_token") ? true : false,
  currentUserData: fromJS({
    name: false,
    links: fromJS({
      avatar: false,
    }),
  }),
});

function appReducer(state = initialState, action) {
  switch (action.type) {
    case CLEAR_ERROR:
      return state.set('error', false);
    default:
      return state;
  }
}
```

Figure 13: Example code featuring a Redux reducer function



```
export function* signInFlow() {
  while (true) {

    // listen for the SIGN_IN_REQUEST action dispatched on form submit
    const { payload: { data, resolve, reject } } = yield take(SIGN_IN_REQUEST);

    // execute the authorize task asynchronously
    const task = yield fork(authorize, data, resolve, reject);

    // listen for the SIGN_OUT or SIGN_IN_ERROR action
    const action = yield take([SIGN_OUT, SIGN_IN_ERROR]);

    if (action.type === SIGN_OUT) {

      // since the authorize task executed asynchronously, it is possible the SI
      // the the authorize task completes, so we call cancel on it
      yield cancel(task);

      // redirect to home page
      yield put(push('/'));
    }

    // remove jwt token from localstorage
    yield call(removeItem, 'auth_token');

  }
}

```

Figure 14: Example code featuring a Saga generator function

In Figure 13, the initial state is taken from a JavaScript object and is made Immutable then the instance method set is used to replace state based on the action type. In Figure 14 the listener is activated, waiting for a state of SIGN_IN_REQUESTED to be resolved before continuing. Once this state is reached, the process is forked and another listener is made active.

4.6.2.7 Electron

Electron is a solution to developing cross-platform desktop applications, using web technologies, derived from GitHub's text editor application, Atom. This framework is already in use my many production applications, developed by companies such as Slack and Microsoft, and has been made available for use under the open source MIT license. Electron is built on the same engine used by the Google Chrome web browser, Chrome V8, a very high-performance JavaScript engine written in the C++ programming language. Using this technology would allow the porting of the web application to a desktop application with ease, allowing native calls to the desktop's Bluetooth driver.

4.7 Possible Architectures

Brandon Ashley	Daniel Mor	Niko Tubach

Diagram Legend: Each color represents a team member which is responsible for a block

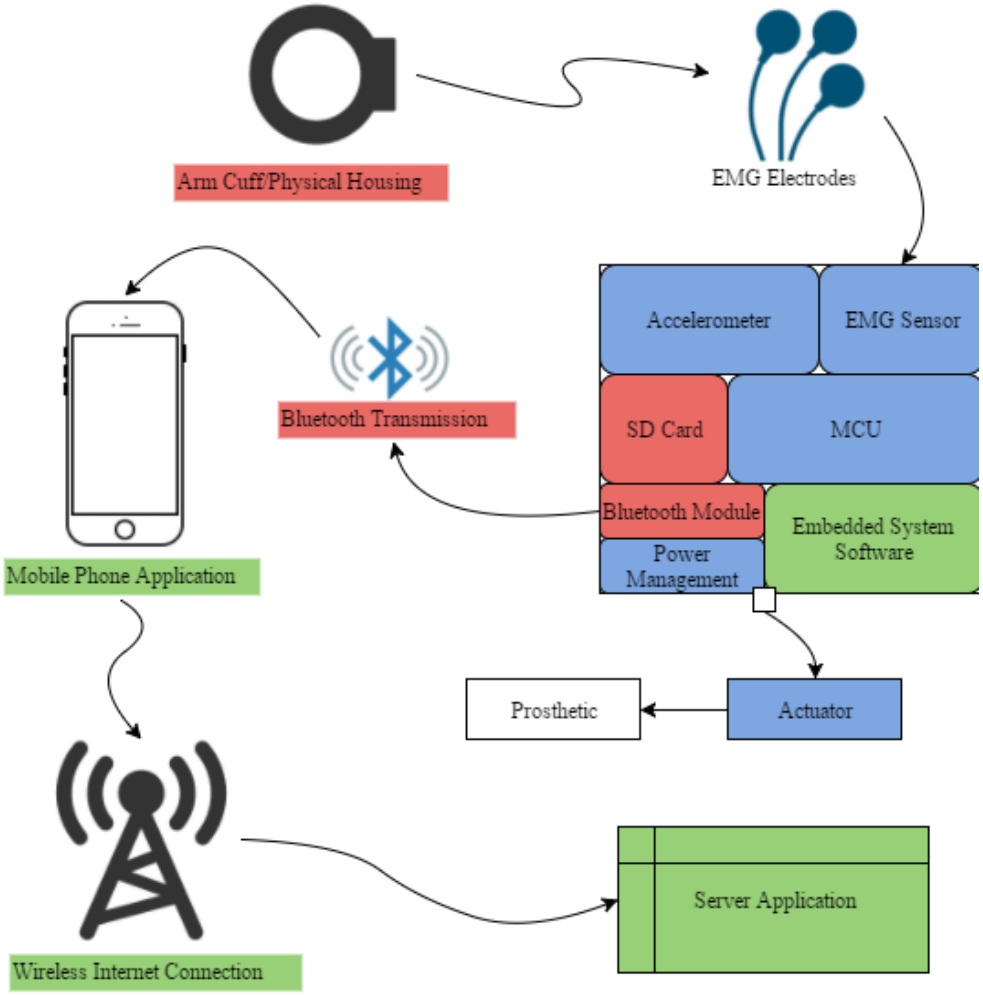


Figure 15: High Level Overview of the System

Figure 15 describes the overview of the entire system and its various connections. Starting from the EMG electrodes which interface with the user and the cuff which houses the electronics. The PCB interfaces with various sensors, and subsystems and transmits wirelessly over Bluetooth to sync with a nearby mobile phone. Using an LTE or Wi-Fi connection, the custom application on the phone sends and receives data as needed from a remote server. This diagram simplifies the intricacies of the connections between the sub-system and acts as a high level view of the proposed system for ease of reference.

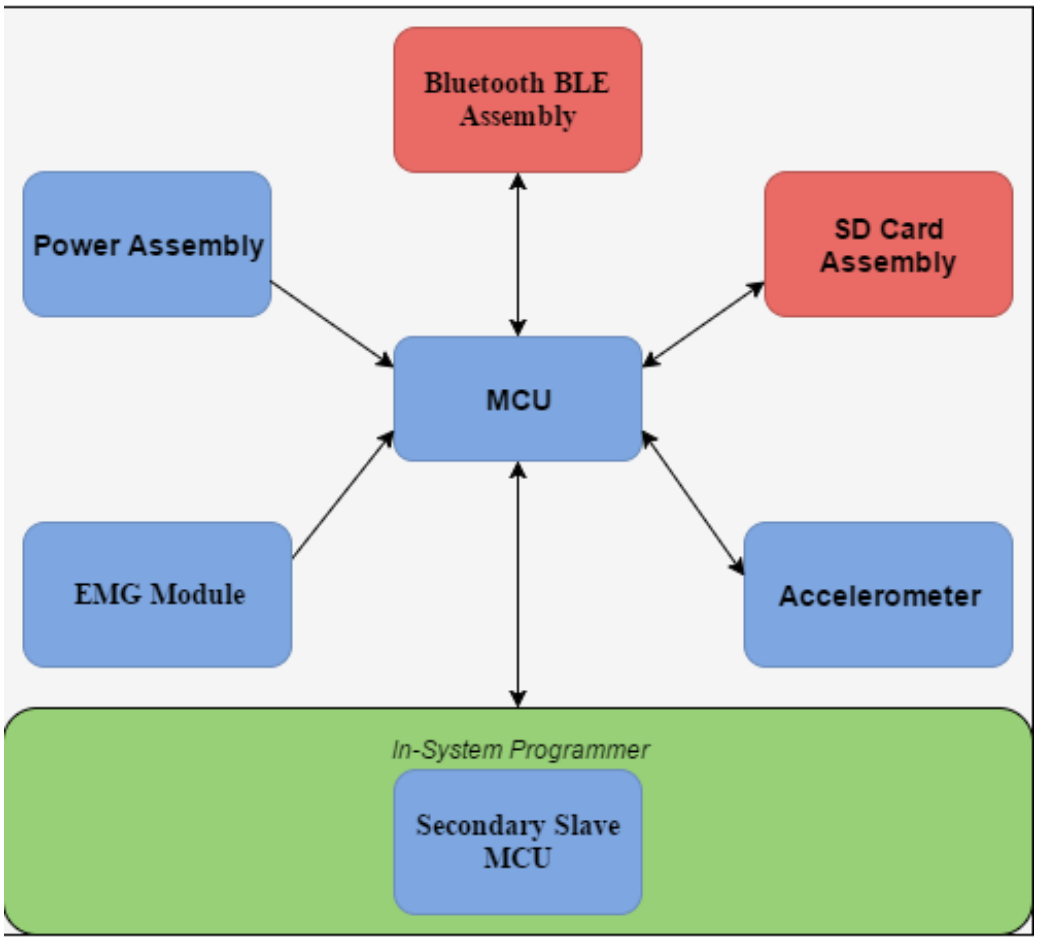


Figure 16: High Level Overview of PCB Components

As shown in Figure 16, a close up view of the simplified connections between the subsystems within the printed circuit board including an expected flow of power and data between the components. The color describes the focus of each team member responsible for designing the subsystem, while the team as a whole will focus on integration and testing. The microcontroller handles all processing on the device and interfaces with most other systems including the various on-board sensors and peripherals. This includes EMG sensors and the accelerometer which provide information about the outside world to the microcontroller along with the SD card reader and secondary microcontroller which would be used to reprogram the primary microcontroller wirelessly.

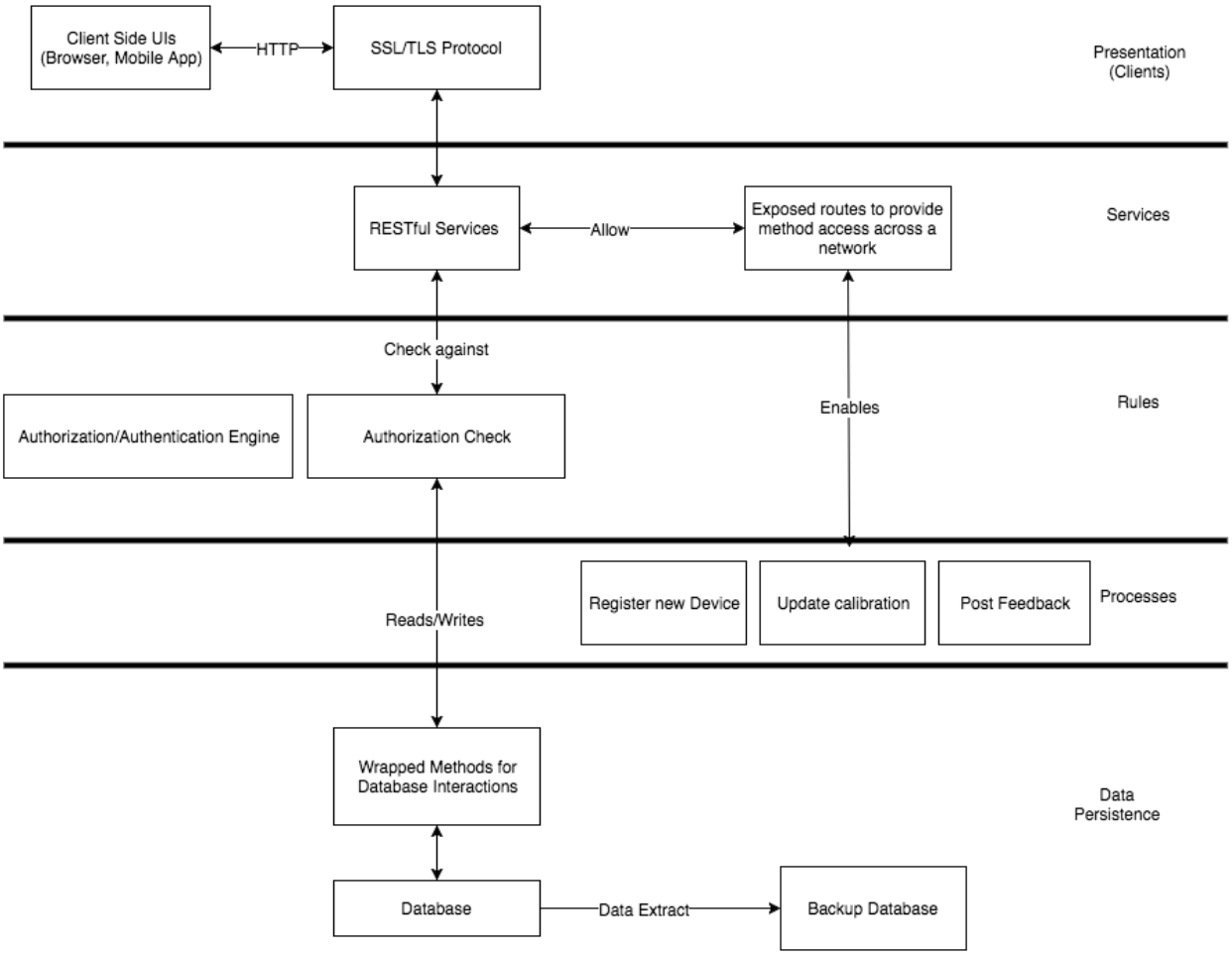


Figure 17: High level architectural overview of the application layers

Figure 17 showcases the proposed system's architectural view from a software analysis standpoint. Specifically, it utilizes a customized layered architectural model to convey a better understanding of how the software components used within the system will interact with one another. The presentation layer includes the mobile application and internet protocols utilized on the client-side to interact with the software of the server side system. The services and rules (security) layers work together to enable and limit the processes of the system which provide authorized actions for a user. The final layer of the model is data persistence which describes the database layout as well as the wrappers used to interact with it in the effort of preserving data for future use.

5. Design Standards and Constraints

Conforming designs to utilize well known design standards is crucial in order to increase reliability in the usage of the device and increase the speed of development. By utilizing internationally accepted standards, the device will be more widely accepted and multiple products will not need to be developed, where each one conforms to an individual standard, specific to different countries. This will allow the product to reach a larger audience and minimize developmental costs.

5.1 Design impact of relevant standards

Standards can be both positive and negative when taking the design of a product into account. Generally speaking, relevant standards improve the design process by allowing for uniformity and continuance in design, but occasionally hinder creativity due to the fact that there are limitations placed on the process. The standards used for the W.A.R.P. project are intended to further the design's ability to be used across the country and reduce the need to train new engineers being familiar with custom standards.

When beginning a design from scratch, it is often helpful to start from a template which can guide the engineer on the path which is most often used. These templates or standards are often tried and true design techniques which showcase the most used techniques for implementing the idea. Standards accomplish something unique, which is to make a product accessible to people from around the world. Furthermore, standardizing the design process increases the possibility that many engineers will be familiar with the technologies. As an example, Universal Serial Bus (USB) is an industry standard which has specifications for hardware and software to meet in order to be considered a proper USB device. This makes a design which uses this standard extremely predictable, knowing how much power can be drawn, the speed of the data transfer and how the code should interface through it. Lastly, these standards allow interoperability and reduce doing work from scratch that probably exists already. Rather than writing a software library to interface with a USB peripheral, it is possible to use pre-existing drivers which accommodate the standard. With all this explained, standards have an important place in the design of W.A.R.P. as to increase the sustainability of the design as time moves forward. This will allow many sub-systems to be re-implemented in future designs and allow other teams to avoid the need of starting from scratch.

Standards aren't always as helpful as they originally seem, since it is possible that the standards would be too long, specific or intricate to actually be understood. Often times standards are described in books which span hundreds or thousands of pages in addition to costing money to even access. It may take years of practice to fully grasp the intricacies of the design which the standards convey. Sometimes it is simpler to bypass a standard and do something in a custom format, rather than invest time into understanding a standard. This time could be used to actually design the

product and the money could be set forth to manufacture it. In this example, it really depends on the resources available and the simplicity and how widespread a standard actually is to decide which ones should be used. Some must be used due to legal requirements, such as wireless regulations provided by the FCC. Some standards on the other hand are so common that it is practically pointless to not utilize them, such as the protocols which enable internet communication such as TCP / IP.

As described in the sections above, using standards have both pros and cons which can't be overlooked. The team has decided to only use the standards that are legally required, such as the ones regarding the FCC and wireless signal propagation as this would require a much higher level of knowledge on electromagnetic fields (EMF) in addition to the fact that components can be purchased which already integrate the requirements of the standards into the device itself. Other standards will also be used which simplify the development, without requiring too much of a steep learning curve to utilize. When possible, components which easily enable using these standards will be chosen over others.

5.2 Hardware Standards

The physical printed circuit board in which the design will be implemented will be manufactured to be RoHS compliant, a major feature of which revolves around a lead free design. This will increase the safety factor when engineers, families, or the children themselves interact with the circuit board. The PCB will adhere to IPC-A-600E as a Class 1 device upon visual inspection. (Industry standard for defect detection). This is a high quality assessment of PCB design to meet industry level inspection standards. Additionally, the International Protection (IP) rating of hardware for ingression protection against foreign elements will be used to measure the W.A.R.P. design with the goal of achieving an IP20 rating.

Inter-Integrated Circuit (I^2C) communication protocol is a widely used industry standard which allows multiple "slave" digital integrated circuits (IC) to communicate with one or more "master" chips. Only requiring two signal wires in order to exchange information, this greatly reduces the number of GPIO pins which must be utilized by the master device which in the case of W.A.R.P. will be the MCU. The I^2C bus consists of two signals: SCL (clock) and SDA (data), where the master will generate a clock signal and broadcasts it to all slave devices in order to synchronize communication between them. The data line is then used to transfer information to an individually addressed peripheral by using a specified protocol. This will be used due to the fact that I^2C is very well supported in hardware, and many software libraries are available to speed up programming.

The Universal Serial Bus (USB) is a hardware and software standard which regulates the available power for a device, and provides the capability to negotiate different parameters. USB is used by many devices as a standard way to provide power and transfer data at high speeds. Additionally, the USB connectors are very

predictable and come in a standard size as well as mini and micro sizes for devices with a smaller form factor. It can be expected that any home which uses technology should have cables, power supplies and other accessories for USBs (including in their computers). W.A.R.P. would like to embed a USB connector into the PCB for at the very least providing power, charging or possibly even data communication depending on time constraints discussed in the following section.

Although it is not necessarily considered an official standard, pulse width modulation (PWM) is a widely used technique for multiple purposes in electronics. It works simply by pulsing power on and off at a high frequency in order to control the current flow through a device or even to encode messages in the signal. The reason this is discussed in a section for standards is due to the fact that servo motors often use a standardized PWM encoding to control position or speed. This well-known method enhances the interoperability of using many servos from competing brands.

The Joint Test Action Group (JTAG) is an organization which developed a method of verifying designs and testing PCBs after being manufactured. JTAG became the name of a standard (IEEE 1149.1) which has been extended across many manufacturers and includes many specialized variants with different features. W.A.R.P. plans on including a way to debug the integrated circuits by utilizing a JTAG as a debugger. Additionally, the JTAG can be used to reprogramming a device, and will be considered as a practical method to employ the feature of wireless reprogramming discussed in previous sections.

Bluetooth operates within the Link Layer of data communication and allows for master/slave connectivity for information transmission. Bluetooth enabled hardware components within and involved with the W.A.R.P. board will comply with the core specifications (revision 4.2) of the Bluetooth Standard, specifically to transmission of data and functionality for connection security. In addition to Bluetooth as a communications standard, the team primarily plans to use it due to the readily available Bluetooth development stack which is provided by many companies such as Texas Instruments.

Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used in the same manner as that of I^2C, short distance communication within embedded systems. Due to SPI's nature of providing high transmission speeds at the cost of the ability to perform more complex data management the W.A.R.P. project will avoid implementing this particular protocol aside from high throughput components such as the flash memory.

5.3 Software Standards

The BLE standard for transmission of data will be adhered to in order to provide a secure transfer of information from the cuff module to the mobile application. More specifically, as defined in the version 4.2 core specifications (Vol. 3, part H) 3.6.2

“Encryption Information”, the BLE controller will encrypt data to be transmitted using AES-CCM cryptography. This functionality produces 128-bit encrypted information from a similarly sized key and plaintext data by using the AES-128-bit block cipher (defined by FIPS-1971).

The coding styles used in the scope of the mobile application, web application and database development of the W.A.R.P. project will follow a combination of the AirBnB style guides for CSS, JavaScript, and Ruby, a curated list of industry best practices, and ideas gained from the team’s collective years of experience. Managing such coding standards allow for an easily maintainable codebase and ease of onboarding new individuals to collaborate with or replace existing project members. These coding standards must be adhered to during this project’s lifecycle, enforced by the team members during a code review.

Beginning with simplest of standards involving overall format, soft tabs of two spaces are to be used in place of tabs. This ensures that code is formatted identically across text editors on differing operating systems. The only exception to be made is the event where physical memory is limited, ASCII characters each take up equal space in memory, with this one tab is smaller than the equivalent spaces. Trailing whitespace is not to be submitted as source code, the extra space is not necessary to the program’s execution. Lines of code should only be separated to make method definition and variable declarations clear, never creating consecutive lines of separation. A clear understanding of these fundamentals should be developed prior to contributing to this project.

Clear and effective commit messages, short messages attached to the commit you’ve made in an effort to improve the existing code, should be applied when attempting to merge in a new feature. This means developing a concise vocabulary to be used amongst team members when performing a variety of edits, additions, or removals. Doing so will allow for expedient code reviews, writing better release notes after an iteration, and easier maintenance of the codebase by offering an easily digestible project log. Below is each expected keyword to be mentioned at the beginning of each commit message:

- :memo: when updating README/docs
- :bulb: when having a new idea
- :lipstick: when improving the format/structure of the code
- :racehorse: when improving performance
- :penguin: when fixing something on Linux
- :apple: when fixing something on Mac OS/ iOS
- :checkered_flag: when fixing something on Windows
- :beetle: when fixing a bug
- :fire: when removing code or files
- :white_check_mark: when adding tests
- :lock: when dealing with security
- :arrow_up: when adding db migrations

- :arrow_down: when removing db migrations
- :heavy_plus_sign: when adding feature
- :heavy_minus_sign: when removing feature

Strategic and consistent branching, the act of copying a node of the codebase to make edits to be pushed back to, will be upheld to easily track changes made feature by feature across iterations and releases. Releases are versioned in a semantic way, the first number represents the major release, second is the minor release, and the third is the patch release. The release number is to be structured as <major>.<minor>.<patch>, (i.e. 2.1.1). As previously discussed in section four, the approach to task completion will obey the process set by the Agile Methodology, eXtreme Programing, as such each iteration will last one week. Each code repository will indefinitely have a branch labeled master, a branch labeled development, and a branch labeled for each release. All of these branches are to never be pushed to directly, meaning each iteration will have many feature branches that will hold the pushed changes and be merged into branches labeled by each iteration. As a strict naming strategy, each branch be defined and will follow the format below:

5.3.1 Feature Branch

These branches are to directly reference ticket numbers offered by the task management software, decided in Team Organization. None of these branches will be “long lived” and should be retired upon task completion. In rare instances a branch of this type may be created with the intention of being broken into smaller feature branches, tasks will be defined in such a way to avoid such measures. The naming format will begin with the number representing the task or feature number followed by a brief description of a few words, all chained with underscores:

```
<ticket no.>_brief_description
```

5.3.2 Iteration Branch

Branches of this type are set to represent each iteration and have the lifecycle equivalent to that of each iteration, one week. All feature branches are to be merged into this branch after unit testing, the features involved in each iteration are decided prior to its beginning. The naming format will be the name of the iteration, keeping this simple, the name will be:

```
iteration_<iteration no.>
```

5.3.3 Development Branch

Only one branch of this type is to exist, this branch will be the parent to all iteration branches. Prior to the merge of each iteration branch, integration tests will be written to ensure that no unintended side effects are introduced to the codebase. Deploying to the staging server will be made from this branch.

```
development
```

5.3.4 Release Branch

Branches marked as release branches are to represent a particular version and stem from the development branch after thorough testing, thus determining stability. Many branches of this type are to exist while some will be retired, those that will remain active are to be of major releases (i.e. 1.0.0, 2.0.0, etc.). The naming convention for these are to be:

```
release-<version no.>
```

5.3.5 Hotfix Branch

This type will be used in the event a software bug surfacing in the current version running in production. These will have a short life cycle and are to be set into deploy after testing. Hotfixes are to merged into development, following a patch bump in as a release, then merged into the master branch, defined below. Following this format:

```
release-<major.minor.patch_bump>
```

5.3.6 Master Branch

This is the branch set to hold the completely tested code running in production, at the currently released version. Merges are to come directly from the release in order to maintain a properly versioned codebase.

```
master
```

Server side security is of the utmost importance, when handling transaction from client and storing sensitive user information. Access is only to be gained through Public Key Authentication encrypted via RSA, no plaintext passwords to be used to gain tunnel access. BCrypt will be used to encrypt passwords, chosen for its slow decryption time, this hashing algorithm can be computationally 10 million times more expensive than the common MD-5 algorithm. The hashing algorithm used for token passing for authorization between client and server will be HMAC with SHA256 (HS256), this is to keep a low memory footprint, as memory is limited in HTTP headers, while providing secure encryption requiring a server side key.

Automated testing must be used to provide continuous integration to ensure as few bugs are introduced to the system. Unit testing will provide method level testing, ensuring data is manipulated in the way intended, side-effects are managed, and errors are handled. Higher level integration or

acceptance testing will also be in place, checking the entire system interacts with itself and all services the way it should.

5.4 Constraints Overview

The W.A.R.P. team has an ever growing number of features that it wishes to include, but is hindered by various real world constraints which limit what can be accomplished. As the project is developed, the team is tasked with using their knowledge and experience to effectively work through these problems and see the requirements through to completion. Many of these constraints are unavoidable in the scope of the design process, but the W.A.R.P. team will work to engineer any possible solutions. The following sections outline and describe the various constraints imposed upon the project, and how the team plans to work with them.

5.5 Economic and Time constraints

5.5.1 Economic

Although Limbitless Solutions donates the bionics it produces, cost is still a driving factor for a multitude of reasons. The devices still cost money to research since this requires expensive evaluation boards in order to test the viability of including certain components in the final design. In addition to research, these boards need to be professionally manufactured in high quantities in order to keep the price per board low enough to be sustainable. Cheaper devices further the organization's goals of helping as many people from around the world as possible. One of the primary reasons Limbitless continues to be so successful is due to the low price point, compared to commercially available alternatives priced at thousands of dollars each with similar functionality.

The W.A.R.P. team has two primary economic constraints imposed on the project by the sponsoring organization. The first is that the team will have a budget of approximately $1000 to research, design, and manufacture a working prototype. This restricts the team to avoiding the purchase of any unnecessary equipment and only acquiring the most significant of development boards for testing. With this in mind, the team will utilize various resources provided by UCF for free to engineering students. This includes a machine shop, electronics labs and a variety of other expensive equipment which is readily available for the team to use. In the case that the research and development process is more expensive than originally anticipated, Limbitless may consider contributing a supplementary $500 more.

Another financial constraint is based on the need to keep the bionic limbs as cheap as possible. Limbitless requests that the final cost of production, the entire product should cost below $400. Specifically, the electronics, while manufactured at high quantities should cost less than $80 per board. This constraint requires the design to avoid the usage of components which are unnecessarily expensive. As such, many components are sold at lower prices, but require a much more involved effort for them to be fully functional. The cost of production will not be offset by sales as the final product will be donated to users.

In order to best utilize the budget, initial financial estimates were written down based on previous team's costs. As time progresses, these financial estimates are updated to be more accurate to the W.A.R.P. project's situation. An emergency fund was left aside to ensure financial oversights can be accommodated. The budget was distributed based on various needs such as: research, development, manufacturing, testing, emergency.

5.5.2 Time

The available time for complete design, development and implementation of the W.A.R.P. project is approximately 6 months starting from Senior Design I till the end of Senior Design II. This constraint is one of the most important, since there is simply no way around it, and it cannot be changed or bargained with. Regardless of what the project entails, the team needs to be able to complete it all by the end of this mandatory time frame.

This time constraint is further detailed with multiple limits. Throughout the course of Senior Design I, the team is expected to focus solely on research and design. This term also requires various deliverables throughout the first 3 months, such as turning in designs on certain days. Senior Design II also has expectations of focusing solely on implementation and construction. This additionally includes the time it takes for the printed circuit boards to be manufactured and assembled.

In order to work through this particular constraint, a detailed schedule of estimated milestones and task completion was created and can be found within section 10.1. Table 10-1 outlines various dates with deadlines, and specifies how long each task should take. Major processes occupy their own subsection, and further explain the estimated schedule.

To further complicate the matter, various aspects of the project are out of the team's control, especially regarding how much time it would take for acquisitions. For instance, it may take a week for each shipment of materials to be received once an order is placed. In this situation, the team will attempt to minimize the number of orders placed in order to maximize the amount of

time which can be used for the design and development of the product. Furthermore, the team members have personal, work and other university related responsibilities which need to be tended to. Due to these responsibilities, each team member has less time which can be devoted to the project. As previously mentioned, the primary way to work through these constraints is with proper planning.

5.6 Environmental, Social, and Political Constraints

5.6.1 Environmental

Printed circuit boards are manufactured using harsh chemicals and produce various byproducts, a common concern is that users who come into physical contact with them may absorb these chemicals into their body. This is an extremely valid concern as there is a well-documented history of these chemicals creating bodily harm to both people, animals, and nature alike. The particular constraint being imposed often times by governing laws is that PCB's should be RoHS compliant when made at the production level in order to offset any potential risk.

In addition, many electronic components radiate heat and use significant amounts of energy over the course of their lifetimes. Although this may be considered an issue on a global scale, this board will be charged through the electric grid as the most efficient public source of energy. The heat produced by the board will be minute and impact the environment on a significantly smaller scale when compared to the majority of household appliances.

5.6.2 Social

Since this product is produced specifically for children, it is important that they see the bionic devices as an extension of their personalities and feel comfortable around them. Especially due to societal pressures that children undergo, it is important that all devices produced by Limbitless provide a baseline of aesthetic appeal. Inherently the designs of these device favor people who are missing limbs instead of people who are not. This is of an even higher importance since these devices cater to aiding children born without certain limbs.

5.6.3 Political

In order for devices that have potential impact to an individual's health, such as a prosthesis, to be officially marketed as medical devices they need to undergo a rigorous approval process put in place by the Food and Drug Administration (FDA). Legally, the W.A.R.P. project, and other products produced by Limbitless Solutions, cannot be termed as prosthetic devices since they have yet to be approved. This particular subject is out of the scope of this team's involvement with the organization. It is possible however possible that they may begin this process moving forward as it is the best course of action to expanding the reach of the organization. The political constraints involve government entities preventing these devices from being advertised without proper approval or safety testing in order to ensure public safety.

5.7 Ethical, Health, and Safety constraints

5.7.1 Ethical

The ethics of charging people in need for this product is not a very difficult problem when considering that Limbitless donates all their products to families free of cost. The only expenses a family is expected of is for maintaining the device and providing replacement EMG adhesive pads which is a recurring expense for using the device.

Additionally, there are no inherently dangerous components which could cause bodily harm, and in order to verify these claims, the team exhaustively tests each board which is produced. As the W.A.R.P. team designs and develops the board, the designs will be verified by both the Limbitless engineers, faculty advisors, and professional contacts from within industry.

5.7.2 Health

Major health concerns can arise with children who are missing a limb due to the unbalanced nature of the body. Utilizing a prosthetic device can allow the user to use more muscles which have a low likelihood of being used. In the case of children missing their forearm, their bicep is rarely used due to not being able to hold anything. By using an EMG sensor connected to the bicep, the user is able to strengthen their muscles by using them similar to how other people do every day.

5.7.3 Safety

Currently, the bionic arms are extremely safe since they are not used for safety critical applications such as driving a car or operating heavy machinery. Furthermore, these devices don't involve the use of high voltage or high current electricity and have a very low likelihood of causing bodily harm, even to their intended user base of children. Since the prosthetic is non-invasive, using external EMG electrodes, utilizing the device causes no intrinsic harm to the user. This is compared to invasive EMG probes which are medically inserted into the body in a somewhat painful manner.

The International Protection, or Ingress Protection, (IP) rating as dictated by the international standard, IEC 60529, provides a classification of a solid object's protection against intrusion by foreign objects such as dust, water or human fingers. Any device that a child will interact with often must ensure that it is, at a minimum, secured for protection against intrusion by fingers (IP20 rating), which also ensures a child can not hurt themselves using the device.

5.8 Manufacturability and Sustainability Constraints

5.8.1 Manufacturability

The product must be designed with a set of tolerances allowing a range of error yet still yield a resultant product without malfunction. The final design must take real world manufacturing limitations into account. This may include 3D printing tolerances, where a 3D model can only be printed within a certain accuracy. Mechanisms designed with tight tolerances may not function correctly if not manufactured correctly.

Electrically, tolerances can completely undermine the design which was calculated without assuming real world manufacturing constraints. As such, resistors, capacitors, inductors and various other components need to be purchased with the smallest tolerance which can be afforded by the budget in order to make the design properly work. In addition to the design of the board itself in conjunction with the components which are used.

5.8.2 Sustainability

Keeping the business running and sustainable is a necessity in real world application of producing a product. W.A.R.P. will not negatively affect Limbitless Solutions as it is a research and development project aimed at

improving the designs and implementations of their products and combine it with modern IoT technologies. This project does in fact improve the sustainability and relevance of the products produced by the organization as they would be more likely to be purchased (if they were in fact for sale).

As an organization which donates its products and receives no returns on its investments (directly from customers). Limbitless needs a constant stream of financial assistance, grants, donations, sponsors in order to perpetually sustain the business it has been growing. In the current time, this doesn't appear to be a problem as the number of supporters for the organization is increasing at an exponential rate with the increasing number of children being helped.

Products require a defined normal operations lifespan that can be related to users so that they can have a relative expectancy of usability in regards to that product. The currently defined lifespan of the W.A.R.P. project is "until replacement". Although the 3D printed housings may outgrow the children who wear them, and they would require a replacement, the electronics themselves should last well over a year with proper maintenance. It is expected that new iteratively improved designs would at some point replace W.A.R.P., as is the lifecycle of electronics.

5.9 Other Constraints

5.9.1 Legality

To remain operational from a legal standpoint a product must maintain proper specifications that fulfill applicable codes to its design. This project follows all applicable FCC regulations for wireless transmission as well as standards (found in section six) that regulate the hardware design. Additionally, these devices are properly labeled in accordance with the FDA.

Any device that has the potential to distribute information about the user must have the data secured in order to prevent parties with malicious intent from obtaining it. With the new Bluetooth BLE 4.2 specifications in use by this design, modulation of the device's Bluetooth address can occur to keep data detection obscure from any but paired devices in addition to the standard 128-bit AES encryption used by previous iterations.

5.9.2 Inspectability

Production ready designs must be able to be maintained easily over their lifetime through open access to components that require inspection to ensure

proper device functionality. The W.A.R.P. design, consisting of an open designed PCB and soldered components, remains prepared for simple inspection throughout its lifetime.

Diagnostic information for monitoring system health should be easy to obtain through a device’s design in order to facilitate inspection. All gathered system data from the hardware in the W.A.R.P. design is transmitted to the paired mobile device for easy reference.

6. Hardware and Software Design Details

6.1 Subsystems & Initial Architecture

In order to simplify the design process, the entire system has been divided into vital subsystems which each handle an important role. Where appropriate, schematics will be included and high level implementation descriptions will be described. This will be a rough draft of the design which may change slightly before final production to allow more time for thorough prototyping and testing. It is the goal that each subsystem will be described as a black box without dependencies on the implementation of the other subsystems.

6.1.1 Communication

As previously described, the BLE stack, similar to the TCP/IP protocol involves multiple layers of software which provide services (input / output) to each other. These layers at one end interface with the physical layer to work with individual bits, bytes and wireless signals and at the other end work with the host application to utilize this data. The SaBLE-x ARM Cortex M0 processor handles the lower physical and link layers of the stack as shown in Figure 18 below. The secondary processor is in fact pre-programmed by the manufacturer and not modifiable by developers. Furthermore, the primary ARM Cortex M3 processor will handle the higher level layers of the stack as part of the TI-BLE stack library. The two processors will communicate with each other by using the ICALL library which acts as a link between the Host Controller Interface (HCI) and the Logic Link Layer of the host. This implementation will allow sufficient control over the low level communications, but additionally provide a higher level abstraction for the software to focus mostly on application.

Aligned with the BLE software protocol, the W.A.R.P. team will implement custom profiles generated using Bluetooth Developer Studio and through LSR's Developer Tool Studio. The SaBLE-x which will be used as originally planned will act as a BLE server and advertise its profile publicly. Nearby BLE compatible devices can query the server to see what services are offered and attempt to pair with the device. From a security perspective, the server can be configured to not disclose any services and only transmit information once securely paired and authenticated. Once paired, the server will advertise services which can be accessed by the client (the mobile application in this case). Each service will maintain characteristics (values) which will be shared between both devices throughout the lifetime of the connection. Modifying the characteristic values on one end of the connection will trigger a signal to be sent to synchronize the values between both devices. The services facilitate the modification of these characteristics and can be

utilized to provide an interface to apply application logic to execute in specific circumstances.

The W.A.R.P. project plans to create a BLE profile which provides a number of services which can be queried, subscribed to, and generate notifications to then be utilized by the host application. Generally, each piece of hardware will have its own service which would be accessible in order to provide detailed control of a device while separating subsystems from unnecessary dependencies. The system will have a profile named “WARP” which contains the following services: Device Information Service, PWM Service, OAD Service, EMG Service, IMU Service, Log Service, and Battery Service. Each service will be capable of exposing the read, notify, or write properties through the BLE characteristics and attributes, synchronizing this data between both devices. As an example, to control two LEDs, the LED service will expose two characteristics. One characteristic can accept commands in hex format such as 0xAABBCC, where to set LED0 to white you would send 0x070001. The first byte is a command to select a hard coded LED color, the second byte is a selection between LED0 (00) and LED1 (01), and the last byte is a color selection between 00 and 07. The other characteristic allows much finer control of LED color by using two bytes per color channel to select any color with 12 bit accuracy. These commands can then be sent from the client application where the host application will be able to utilize this updated value to control a PWM signal which physically changes its brightness. Appropriately, the channel of communication will have similar characteristics and attributes to properly convey the data which needs to be handled. This also works in reverse for sending sensor data to the client application, where the client subscribes to a service and receives updates when the value changes. A full listing of the BLE commands that the firmware accepts will be described in later sections of this documentation.

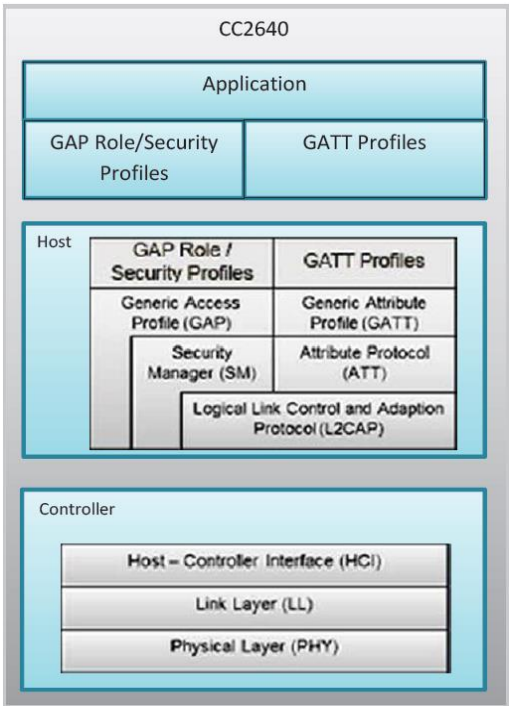


Figure 18: CC2640 BLE Stack
(Reprinted with permission pending from Texas Instruments)

6.1.2 Power

The final device is expected to be powered by a two cell lithium ion (Li-ion) battery which has a nominal voltage of about 7.4v. One design challenge is that the logic of the system is expected to operate at a nominal value of 3.3v at around 100 mA, and the servo motors will require 5-6v at high currents around 2-4 amp peak current.

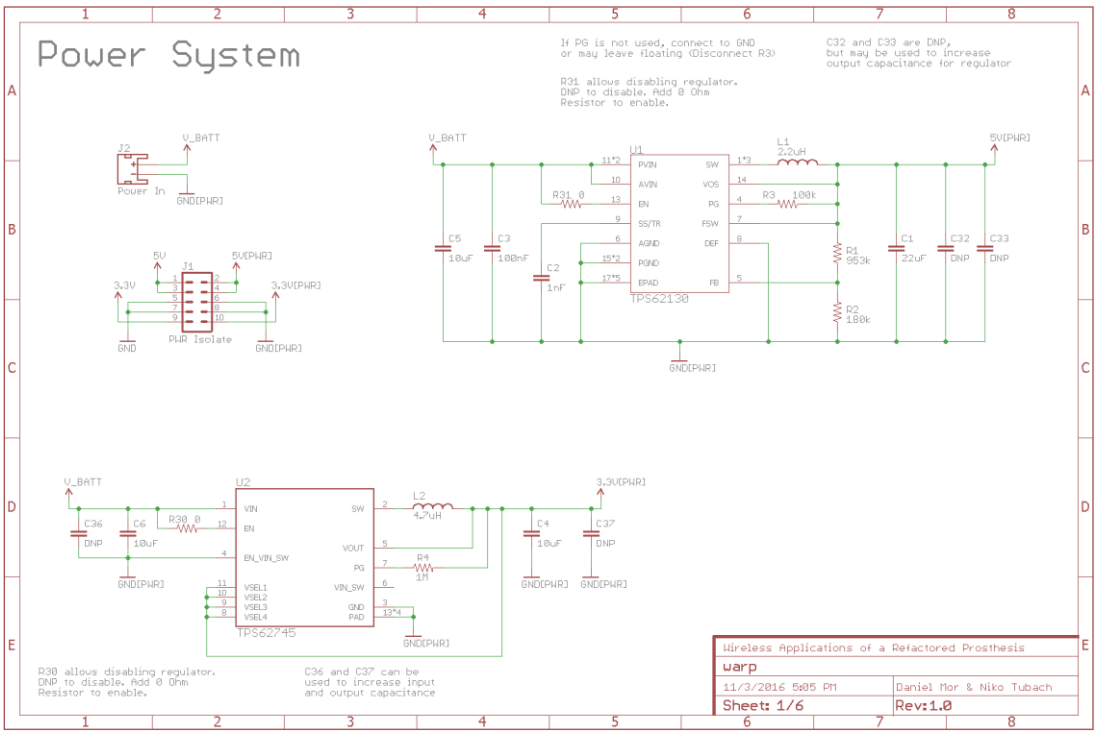


Figure 19: Power System Schematic

Another issue which is acknowledged by the team is that the original EMG sensor design required over 3.5v to function correctly and 5v to function optimally. As a result, the sensor will output over the 3.3v limit imposed by the SaBLE-x ADC. As a quick remedy, a simple voltage divider could be included in the output of the sensor. Other options include an Op-Amp buffer after the voltage divider to prevent the output impedance of the voltage divider to affect the ADC conversion. The final and chosen option is to redesign the sensor to work with 3.3v logic in order to reduce the number of components required. Originally, the LEDs were going to be powered by 5v, but would then require gate drivers to be properly switched on and off by the 3.3v logic level GPIO pins of the SaBLE-x. To simplify the schematic, the RGB LEDs are powered directly from the 3.3v power supply.

6.1.3 Processing

Figure 20 outlines how the SaBLE-x will link its pins with the other subsystems to function as described throughout the paper. In addition, a standard JTAG 10-pin ARM connector is shown in the bottom left as a method of flashing the device. The schematic includes pull-up resistors for the SDA and SCL I2C signals and a hardware debounced reset button. The majority of what makes this section of the schematic interesting will only be visible through software as it only shows the connections to GPIO pins. Through software, these GPIO pins can be multiplexed to internal peripherals such as SPI, I2C, USB, ADC, UART, and so on. Internally, the SaBLE-x also communicates with

its co-processor to pass messages up and down the BLE Stack and eventually through the PCB trace antenna as a wireless signal.

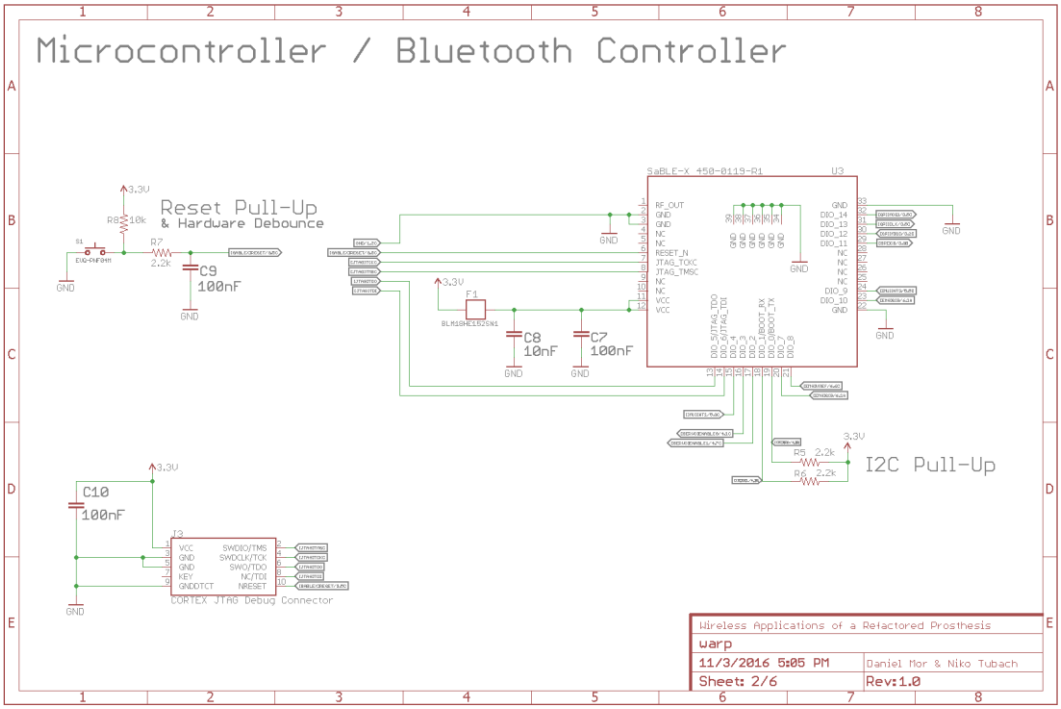


Figure 20: SaBLE-x Schematic

6.1.4 Memory

The SaBLE-x module comes with 128KB of internal flash memory. Upon first inspection, this seemed like a large amount of memory for such a device, but due to the overhead of including the TI-RTOS, BLE Stack, and Boot Image Manager, basic applications use nearly three quarters of what is available. To enable OAD to reflash the device wirelessly, the chip requires enough memory for at least two full images of the compiled program. This means that if a program requires 120KB, and the new program requires the same, the device needs, as a minimum, 240KB of memory which is over the limit of what is available. To resolve this issue, an external 8MB Flash memory IC is included in the PCB design for a number of reasons. First and foremost, to provide enough memory to allow OAD to be implemented in software without worrying about memory limitations. The second reason is to future proof the device and enable the device to log data about itself until able to sync with a mobile device. This exorbitant amount of memory is purposefully over 50 times the minimum of what is needed in order to evaluate the optimal amount of external memory required for future designs. Due to the higher speed capabilities, the external flash is interfaced through the SPI communications protocol rather than I^2C. A layout of the external flash is shown by Figure 21.

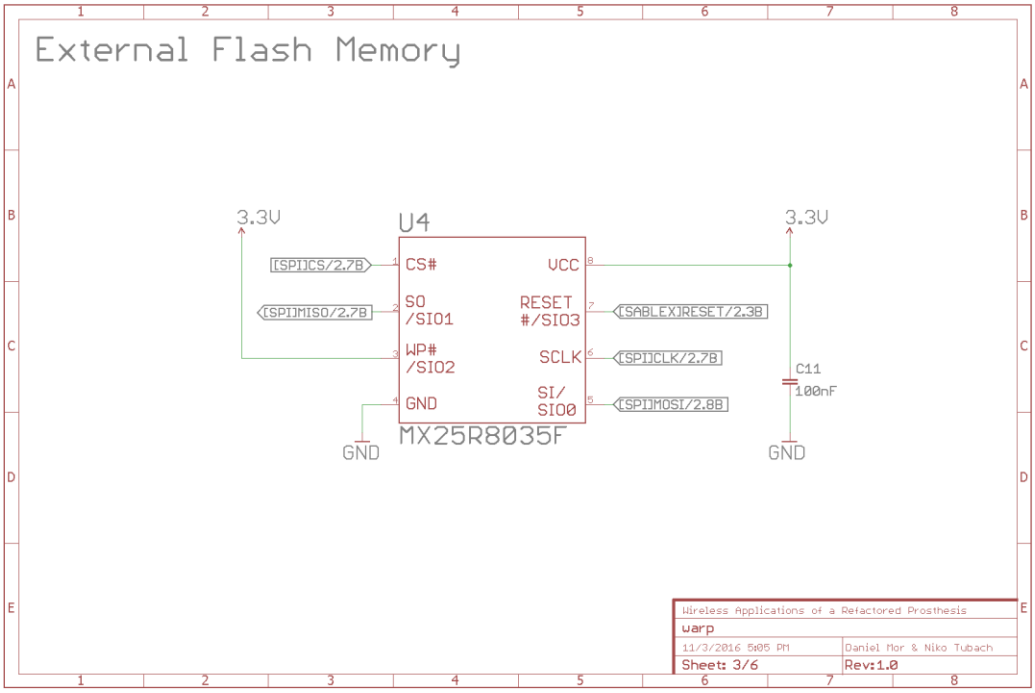


Figure 21: External Flash Memory Schematic

6.1.5 PWM Driver

Since the central processing module, the SaBLE-x, only has access to 15 GPIO pins. The majority of which are unavailable since they are configured to work with communication protocols, a PWM driver is necessary to provide enough control over various external peripherals. This is actually the same device used by T.U.B.A., but was underutilized as only two of the max 16 outputs were in use. In this case, two PWM outputs are used to control two servos. Another six PWM outputs are used to control two RGB LEDs with full color and brightness as shown in Figure 23. Furthermore, the PWM driver is controlled through I2C to minimize the GPIO pins required by the MCU to control the device. In addition to driving LEDs and Servos, two solid state relays are utilized in order to power down the servo motors at will. These relays internally act as optocouplers where a single GPIO pin can be used to power an internal LED within the relay. This LED then illuminates a photo resistor which drives the gate of a Mosfet. Ultimately, enabling a GPIO pin will allow higher voltage / current to power the servos.

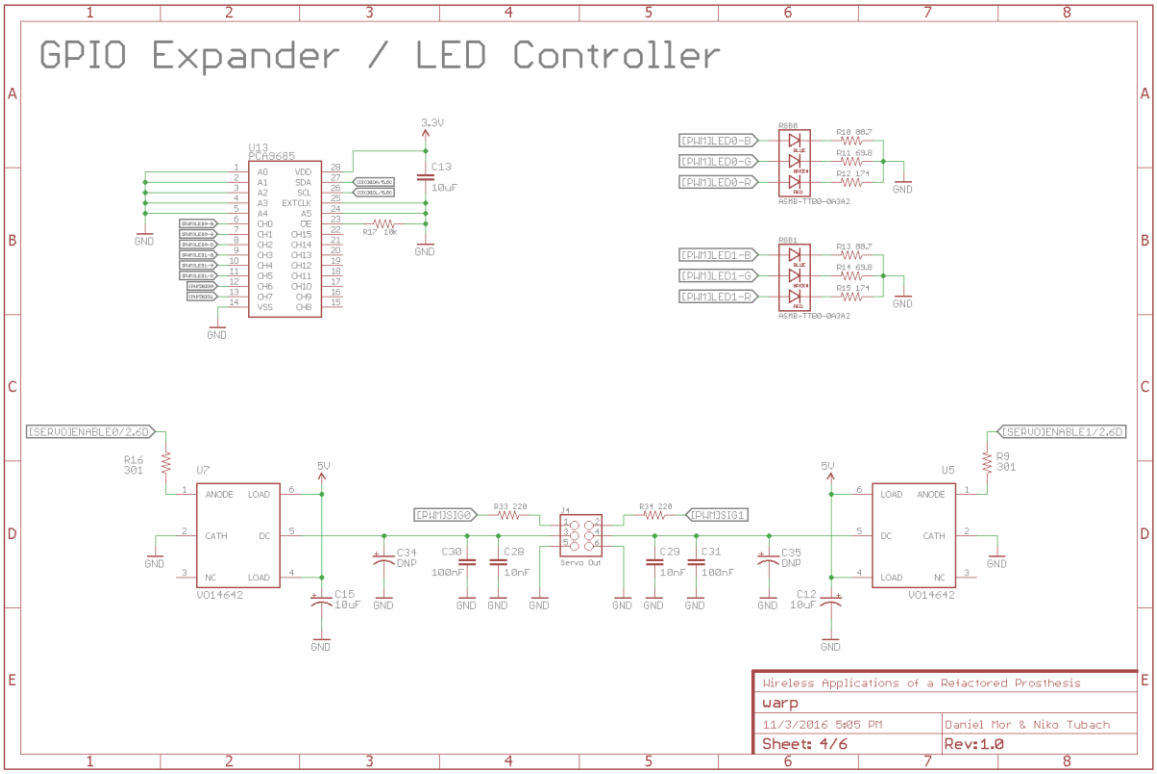


Figure 22: PWM Schematic

6.1.6 Sensing

The IMU is one of the most important additions which W.A.R.P. is including in the device as the information it provides can be analyzed in order to implement gesture recognition within the W.A.R.P. product. This data can work in conjunction with the EMG inputs to trigger more complex control systems. Figure 23 shows the device as it's configured to use I2C similar to the PWM Driver. The device provides a number of interrupts to signal to the main host controller when data is ready from each of its primary sensors. This allows the host to then send commands to request and use this data.

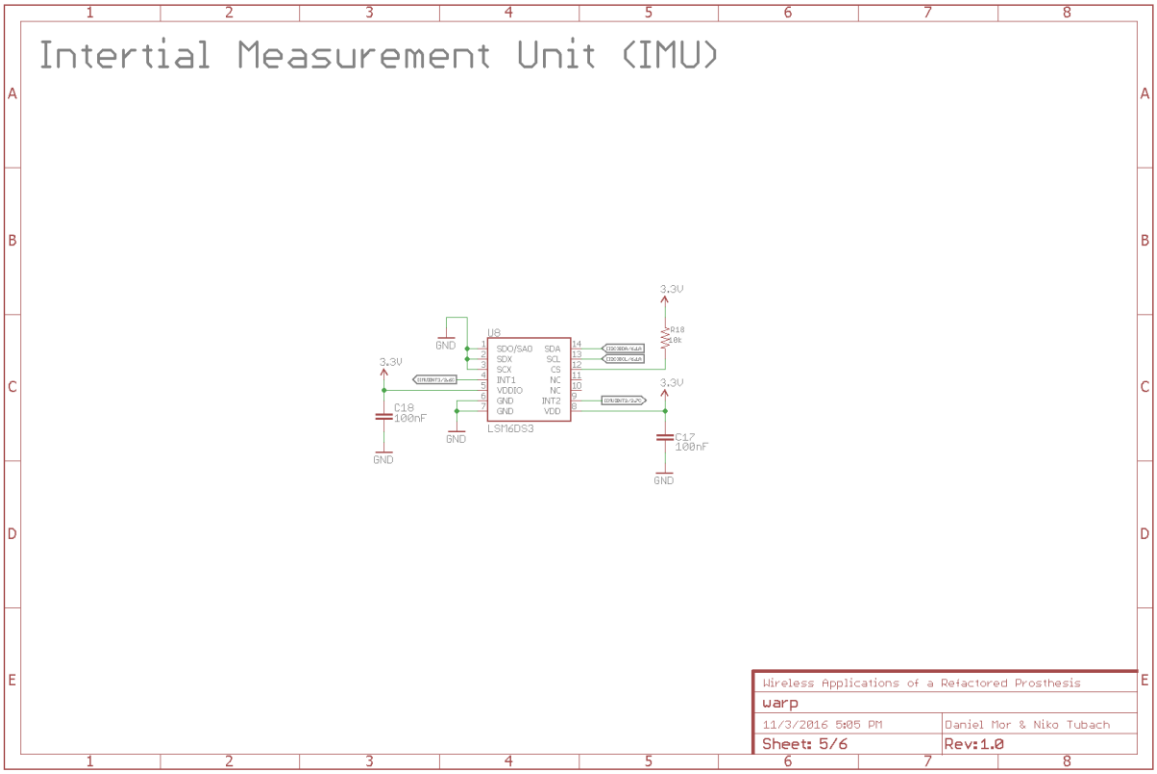


Figure 23: IMU Schematic

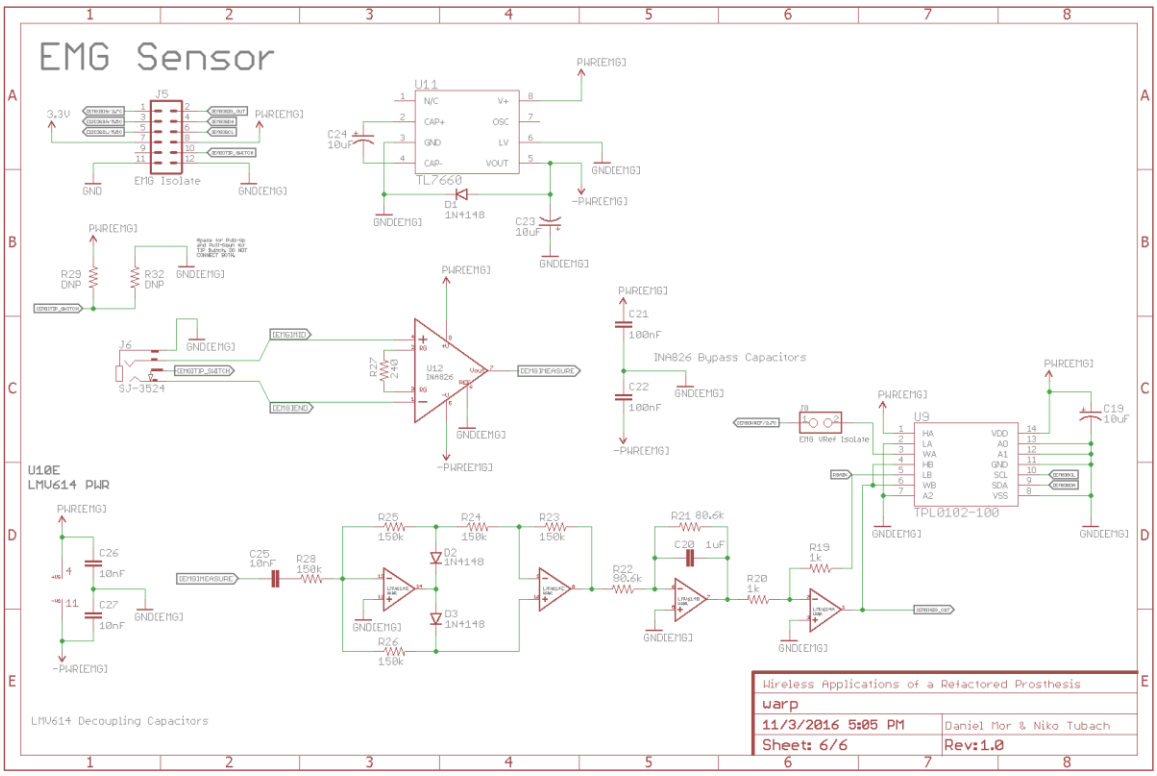


Figure 24: EMG Sensor Schematic

The EMG sensor is the heart of what makes the devices produced by Limbitless Solutions work so incredibly well. The sensor is connected to three electrodes placed on the user's muscles. The first stage of the sensor then amplifies the difference between two of the voltages with the output proceeding to the second to fourth stages to rectify, filter and condition the signal to be used by an ADC. Figure 25 is for the most part designed and created by Advancer Technologies with the addition of a voltage inverter to only require a single power supply to power the sensor. In addition, the more innovative advance is replacing the potentiometer with a digital potentiometer. This device is programmable through I2C between 0 - 100 kOhms with 256 steps in order to finely calibrate the sensor.

The team after much consideration has redesign this circuit to operate at a lower 3.3v logic level as it simplifies much of the circuitry. This was done by updating all the integrated circuits to use TI components, and especially by replacing the TL084 with the LMV614 which has a lower operating voltage.

6.1.7 PCB Layout

The schematics shown above show the logical connections between the symbols. The final step is to lay out the individual traces between the components in order to minimize noise and maximizing space usage. The final PCB layout ended up being 2.25" x 2.25", containing 104 components, 38

polygon pours, 598 wires, 357 SMD pads, and 93 vias. This design was completed for a two-layer board with the addition of debug headers joining the three subsystems: power, EMG, and digital logic. All subsystems are isolated from each other and can be shorted together using a jumper which fits on .1" vertical headers.

This design was completed as a prototype with the hopes of shrinking it to 1.75" squared using a four-layer board, smaller voltage regulators and more compact components with future iterations.

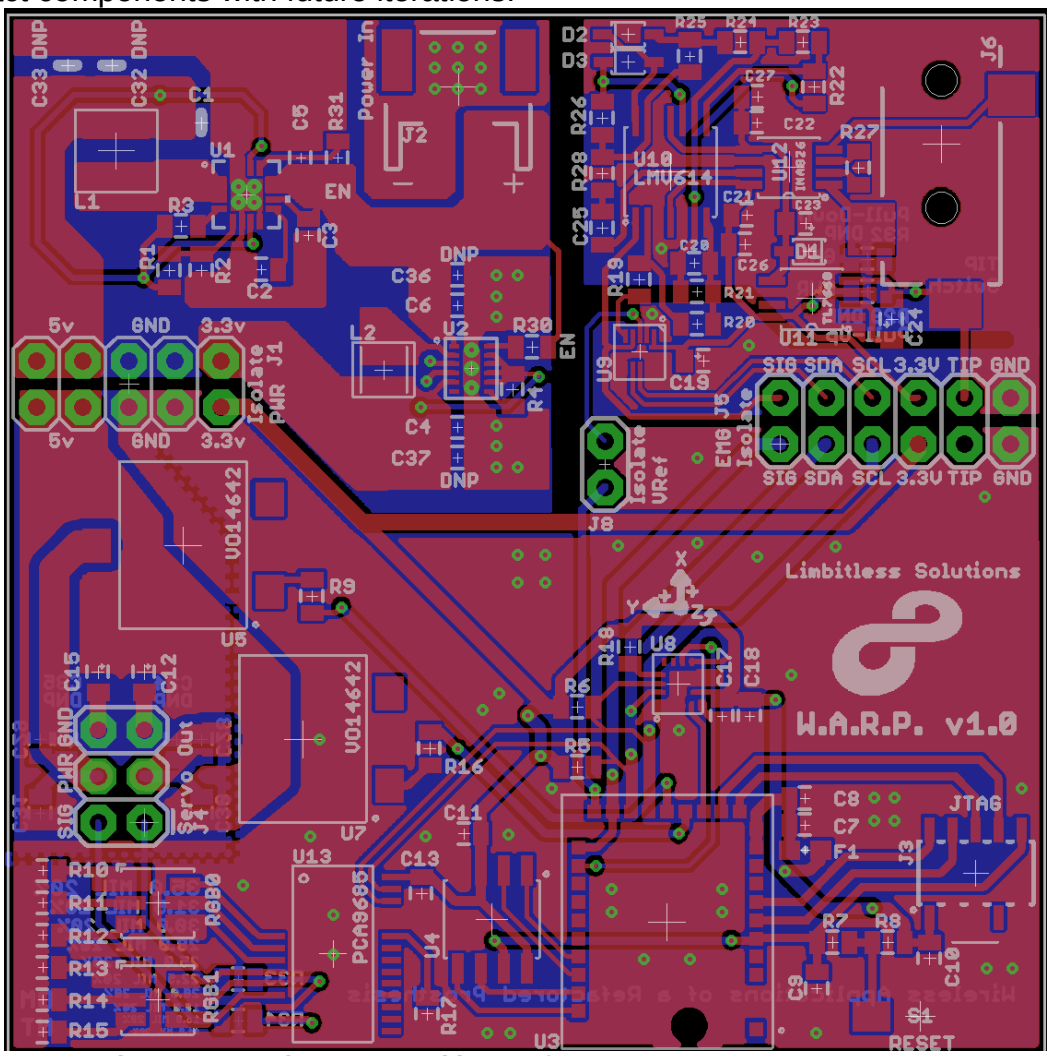


Figure 25: EagleCAD Board layout for W.A.R.P. PCB v1.0

6.1.8 Embedded Application

The code which will be executing on the SaBLE-x, includes but is not limited to the Texas Instruments Real Time Operating System (TI-RTOS) and the TI BLE Stack. This software will be used as a framework for the application software to be built on top of. The basic structure for the BLE Profile will be generated by Bluetooth Developer Studio and integrated into the software as

an API. The application code will include files which access this API to interface with the BLE Stack and RF Core. Furthermore, this code will interface with the application code to read and set data based on BLE communications.

A large portion of the embedded code will revolve around configuring TI-RTOS peripherals and drivers along with creating tasks (threads) which execute on the embedded device. Hardware interrupts will need to be configured to post data to the correct thread once triggered. Similarly, I2C and SPI drivers will also need to be configured to allow communication on the correct GPIO pins, and tasks to include the handling of data as it is received and when it requires transmission. The driver for the ADC will be configured to ensure the EMG signal is correctly sampled and passed to higher layers of the application layer.

Once the hardware based peripherals are created, semaphores will be utilized to ensure mutual exclusion of shared data between tasks. This will allow each task to execute in parallel and allow resources to be shared securely. With all these functions properly configured and initialized, the device logic will be added which will send high level commands through I2C, SPI, or even BLE. These high level commands will be executed based on a timer, when new data arrives from sensors, or due to hardware based interrupts.

Figure 26 below gives a high level technical overview of how the embedded application functions during normal use within the W.A.R.P. system. The individual components are largely self-evident but the ordering of events can be summarized by the following; upon powering on or resetting the system the Boot Image Manager will execute. Upon succeeding the system will initialize the BLE stack followed by the TI-RTOS which will then move the system into its main function. The main function will handle the majority of the initializations that will be used by the application and will end by starting the BIOS and moving into the kernel operations which is where the main application programmed by the user resides. In the case of a shutdown even the system will either gracefully exit and power off or abort application execution in the case of fatal exception.

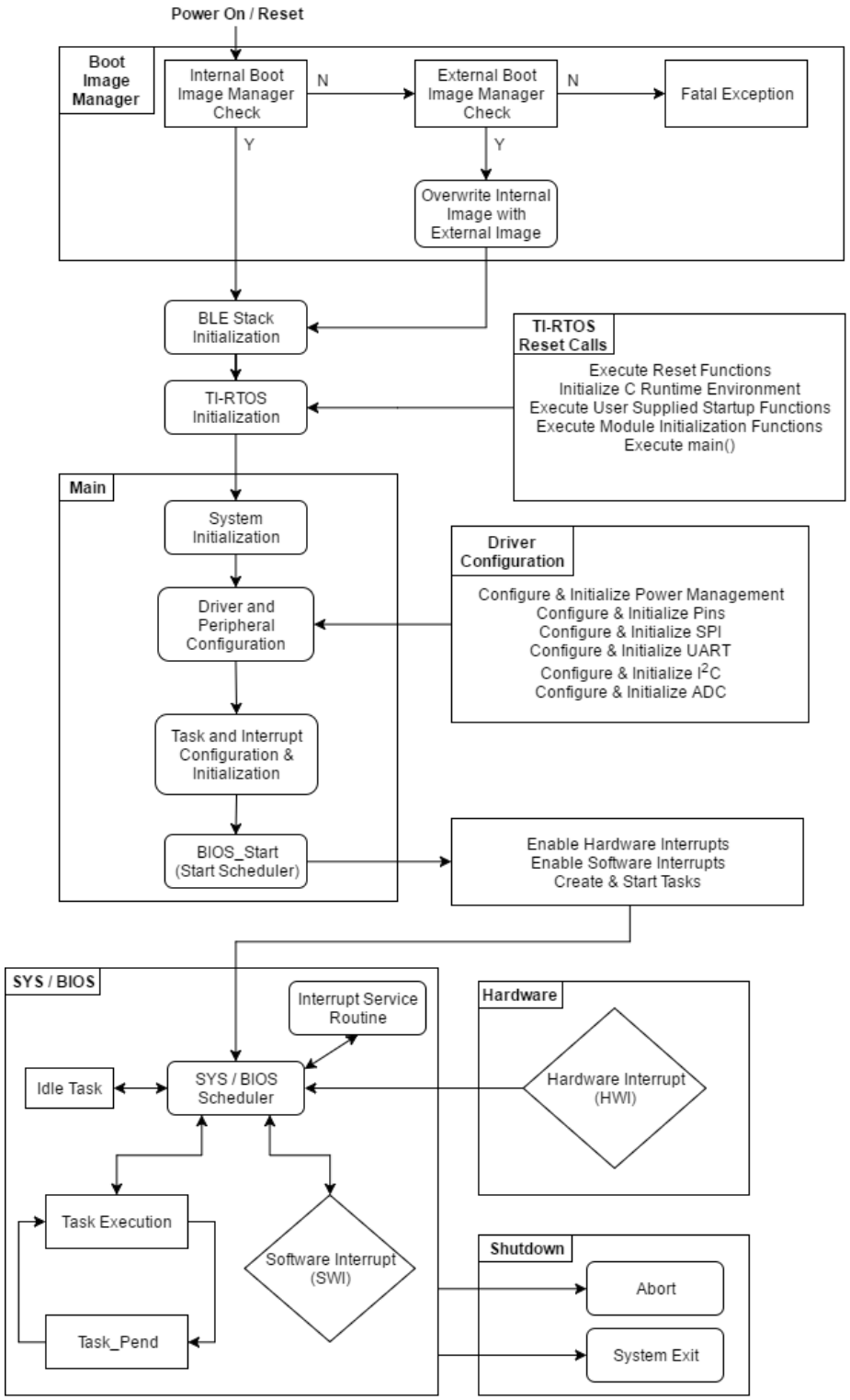


Figure 26: Embedded Software Flow

6.1.9 Client Application

6.1.9.1 Functionality

The Software Requirements state the completion of the project expects the mobile applications have the ability to interface with the Bluetooth module mounted on the bionic device or cuff. Additionally, the mobile applications will interface with the server-side application communicating any authorized actions to and from the user. The web application serves the purpose of administering any necessary updates to a user's mobile application.

The channel of communication between the mobile application and the Bluetooth module will be able to maintain a full-duplex connection, receiving transmission from the module for responsive feedback, then transmitting data to when the embedded system requires configuration change or software updates to separate characteristics, resolved with a Universally Unique Identifier (UUID). The initial connection is to be established via activating the hardware to expose any necessary channels for data transmission.

The web application will be written using ECMAScript 2015, the React view library with the React-DOM package, Redux state management, along with Redux-Saga to manage the side-effects created by the asynchronous behavior of the server-side requests. The client-side applications have the possibility of sharing a large portion of source code, likely up to 60%, this could allow team W.A.R.P. to easily obtain the stretch goals. The mobile applications can share a large portion of middleware, reusing the code from the state and side-effect management services built for the web application. Modules exist in creating function calls to manage BLE connections.

Furthermore, the mobile application tooling used to configure the embedded device’s software and hardware thresholds, will be graphing the data received from the EMG signal in real-time. This can be used to intuitively set the EMG gain determined by the digital potentiometer. Pictured below are a few screenshots of this application before and after the gain had been reduced.

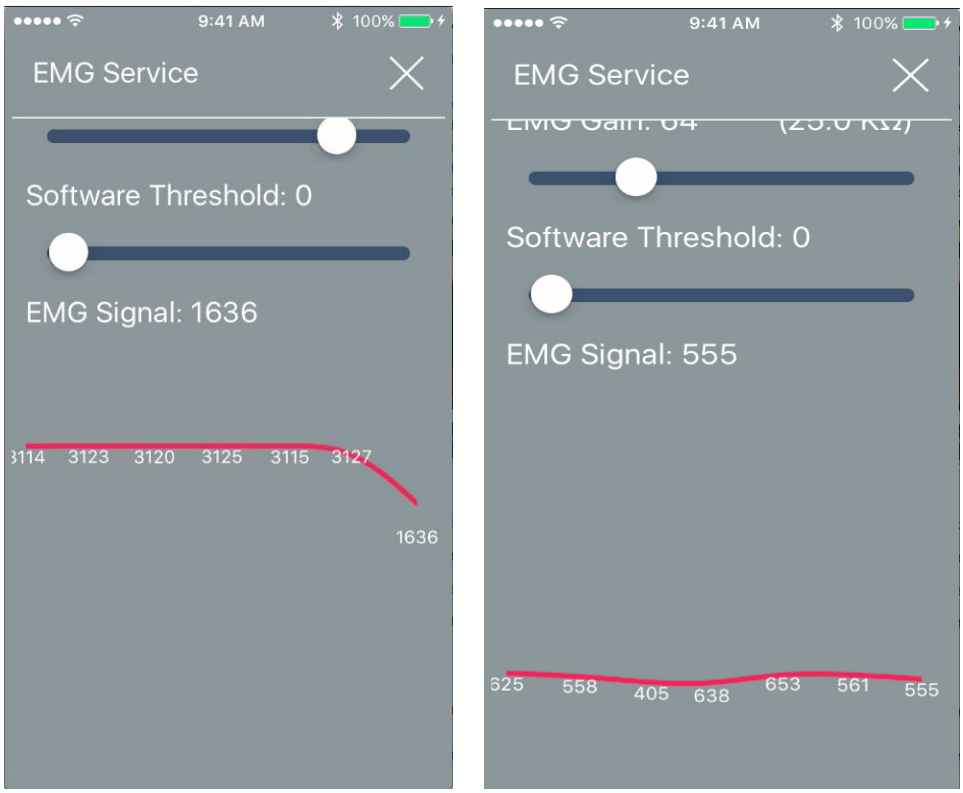


Figure 27: EMG Service GUI displaying EMG waveform with gain high (left) and low (right)

6.1.9.2 UI/UX Design

The User Interface (UI) and User Experience (UX) for all client side applications are to remain consistent with the current design of Limbitless Solutions' website while incorporating native controls where the universal standards exist. Such standards as making use of Android's navigation buttons and Android's use of Material Design for application components.

The user flow will begin by first registering the device with Limbitless by identifying the bionic unit or cuff with an established Bluetooth connection and signing up with an email address and password. Once the user has been on boarded, the user can be authenticated using these same credentials. After the initial authentication the user will receive a brief overview of the applications controls including managing updates, advance device configuration guides for changing the threshold of the sensors and choosing a color to be displayed for the unit's LED, and instructions on making connections with other recipients in the social network. After the overview and any subsequent sign ins, the user will then be brought to the home screen containing any recent notifications regarding communication or updates from Limbitless or connection requests within the social network experience. From this point the user may view live feedback from the device, connect with others or post to the network, or make updates to the unit's configuration. This design is shown in Figure 27.

Figure 28: User Flow for Mobile Application with Screenshots

6.1.10 Server Application

In effort to allow the server-side application to be as maintainable and efficient as possible, it is to be developed using Elixir, Phoenix, and the PostgreSQL relational database system, all hosted using the services provided by AWS. The pattern commonly used by the members of the W.A.R.P. team,

known as Service Objects, well only need to be slightly altered in approaching problem solving when using Elixir. Service Object are a way have creating maintainability by establishing classes with methods for managing business logical outside of the main classes used for server communication and data manipulation, controllers and models. The reason the migration away from such patterns is due to Service Objects being built off of the Object Oriented Programming (OOP) paradigms and Elixir is a Functional Programming (FP) language. These Service Objects can be reimagined as Elixir Modules and may be used to contain the functionality that exists. This approach grants the actors (models) to only deal with getting and setting data stored in the database and the controllers simply serve the purpose of managing requested resources and delegating actions to modules.

The server-side application is to support authentication from individuals with differing levels of authorization. The same approach in handling business logic with Elixir Modules can be applied here. The requestor can be identified by the incoming request's JWT, from there a user actor can be derived from which permissions can be determined and user can be checked against the record and the resource involved in the request and build the HTTP response from this, either returning the requested data or returning a status of 403 Unauthorized.

Any real-time interaction between multiple client-side applications can easily be managed by Phoenix channels. While Ruby on Rails offers a solution for real-time communication, it comes with many dependencies, the Phoenix version of channels provided by the technology already established by Erlang’s virtual machine resolving the dependencies via Elixir’s own runtime. In addition to fewer dependencies, Phoenix channels are far more performant and fault tolerant. In the event of an Elixir process going down, the runtime is capable of restoring a new process to the last stable stage of execution.

7. Prototyping

In order to prevent wasting resources and valuable time, before sending the PCB design to be fabricated, a basic prototype of the subsystems will be built in order to uncover any possible mistakes or faults with the design. To facilitate this, the team attempted to acquire the majority of parts (within the same or similar family) through purchase or free samples when possible. Since the team is interested in breadboarding the majority of the design, chips available in breadboard sized dual inline packages (DIP) for testing and in smaller packages were preferred. When this wasn't possible, chips that could be purchased or soldered to breakout boards were the next best option. This section will outline the major components which were utilized in the prototyping phase, and then show a rough draft of the combined system on a single breadboard.

7.1 Prototyping Components

7.1.1 CC2650 LaunchPad

Although the project will be utilizing a SaBLE-x module as the core integrated wireless and application processor, the CC2650 Development Board (Figure 28) was an excellent starting choice for the team to become familiarized with the development stack. The SaBLE-x module contains a CC2640 chip inside it and the software for the two chips are 100% interchangeable and compatible. The LaunchPad contains various LEDs, JTAG headers, and an onboard XDS100 JTAG emulator with a USB port for simple programming. The team ordered two of these to expedite the development process and test BLE communications between the two. Additionally, the board has schematics openly available and example code which the team can reference when designing the final product.

Figure 29: CC2650 LaunchPad

7.1.2 SaBLE-x Development Kit

The SaBLE-x Development Kit (Figure 29) was produced by the manufacturer of the SaBLE-x module and provides an additional reference to designing and programming the chip. This board was purchased to get a better idea of the functionality and pinouts of the device which will be used in the final product. Although the CC2650 is easy to work with, the SaBLE-x abstracts away many components and features in the physical package such as a built in PCB trace antenna and high frequency filters. T.U.B.A., the team working on the previous iteration of this product ran into many problems due to lack of a functional development board to work with. This board acts as a proof of concept for the custom software and reference for hardware design as it includes many sensors utilizing similar communication protocols such as UART and I^2C.

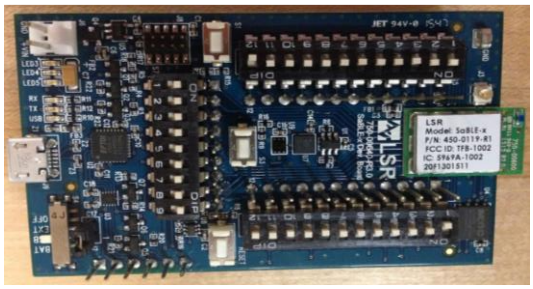


Figure 30: SaBLE-x Development Board

7.1.3 LSR SaBLE-x Breakout Board

The SaBLE-x module (Figure 30) with easily accessible pin headers can be used to breadboard the final design in the breadboarding stage of project. Unlike the development kit described above, this board is a representation of the module which will be placed into the actual PCB. This would provide the team to bypass all external components and ensure the chip functions with the team's design by itself. As a side note, the development kit has built in dip-switches which can isolate the board for similar functionality. This board more importantly may act as a redundant solution in the case of the final PCB electrically failing.

Figure 31: SaBLE-x Breakout Board

7.1.4 XDS200 JTAG Emulator

In order to program and debug the chipset, the team has a few options, but the XDS200, Figure 31, external JTAG emulator is the simplest. Even though the project hopes to be able to wirelessly program the chip, the chip still needs the firmware and BLE stack be flashed for the first time to enable this functionality. The CC2640 is capable of a two pin JTAG interface, but the final design plans on breaking out the full standard 10-pin ARM JTAG connector. This will allow the full set of debugging features to be used for the prototypes. In the future, these pins may be repurposed as GPIO pins once primary testing is completed. Lastly, these devices normally cost $300, but the team was able to acquire it through the sponsoring organization.

Figure 32: XDS200 JTAG Debugger

7.1.5 SX1509 GPIO Expander Breakout Board

The SX1509 GPIO expander board shown in Figure 32 was utilized by the team in order to increase the limited number of GPIO pins provided by the SaBLE-x module. These pins can be used to individually control individual LEDs and provide unattended PWM signals to drive servo motors. Additionally, the team decided upon the SX1509 since it is highly customizable and allows peripheral control through the I^2C protocol. Additionally, the device contains internal timers and peripherals which can sync multiples of the same chip if required and signal interrupts to the host controller. On a final note, this expander allows for variable rail voltages (up to 5.5V) that can be dissimilar between the two I/O blocks powered by VCC1 and VCC2.

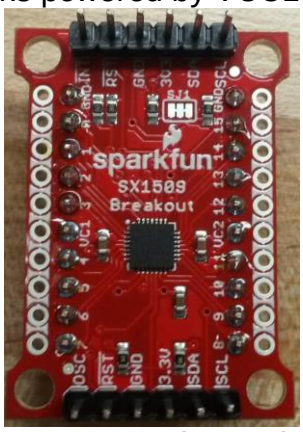


Figure 33: GPIO Expander Breakout Board

7.1.6 LSM9DS1 IMU Breakout Board

Most IMU chips provide an accelerometer, compass, or magnetometer, but usually in individual packages. The LSM9DS1, shown in Figure 33, provides all three in one extremely small SMD package which is able to be interfaced with through the I^2C protocol. As a breakout board, the team will be able to test the chip using a breadboard in order to ensure proper functionality within the design before manufacturing the PCB. This will minimize design errors as they can be found before spending the valuable time and money.

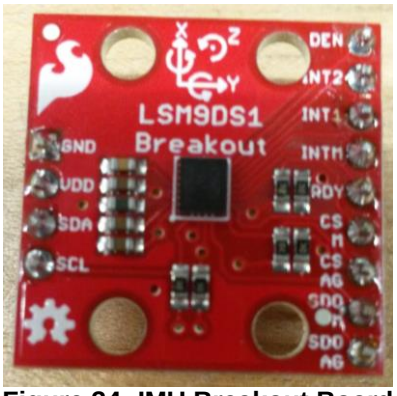


Figure 34: IMU Breakout Board

7.1.7 FT232RL USB-to-Serial Breakout Board

As is the case with the other breakout boards utilized throughout the design and prototyping process of this project, the FT232RL breakout board (Figure 34) will allow the team to interface with a computer using custom FTDI USB drivers on the computer's end to communicate with the chipset. Furthermore, the chip will convert these communication signals to a simple serial protocol which can be interfaced with through a UART. Although it may be a good idea for future teams to develop a custom USB driver and utilize the internal USB peripheral of the SaBLE-x module to interface with a computer or custom device, this lies outside the scope of this project. Lastly, a Micro USB Type-B breakout board (Figure 35) can be used with an individual FT232 chip to directly communicate with a computer through the chip without the entire breakout board.

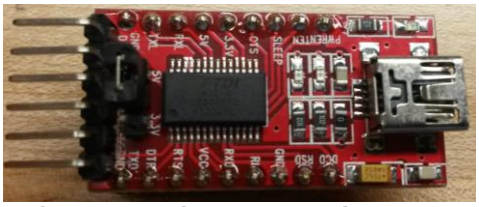


Figure 35: Micro-USB and FTDI Interface Breakout Board

Figure 36: Micro-USB Breakout Board

7.1.8 Advancer Technologies EMG Sensor v3

Similar to past projects of Limbitless Solutions, the W.A.R.P. team has obtained a EMG Muscle Sensor v3 produced by Advancer Technologies. This product has its circuit schematic openly available, which is the primary reason the team opted to use it instead of its newer counterpart, the MyoWare EMG Sensor. The team plans on using this breakout board and freely available schematic to embed it in the PCB and possibly improve it if time permits. This sensor board, shown in Figure 36, will provide the basis of testing and understanding how it works.

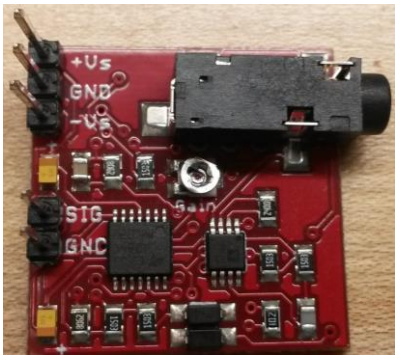


Figure 37: EMG Sensor Breakout Board

7.2 Breadboard

The perspective of the breadboard, Figure 37, can be expressed from the top-down left-to-right orientation as follows: the IMU module is seen first in the top left. Following in line are the external flash memory and the EMG sensor module. The next line contains the SaBLE-x wireless module and the GPIO expander with the six colored LEDs to its right representing the two planned LED arrays. The following far left contains the connected servo leads. The bottom line of the board showcases the FTDI USB to Serial interface module followed by the representation of the regulator for the system. A general rule of thumb to coloration of wires is that yellow represents I2C connections and servo wires, teal represents signal passing and SPI connections, white represents data communications, red and orange represent VCC while black and blue represent ground. Differences between this breadboard design and future PCB designs will include moving the modules around as they are only placed in this manner physically for ease of connection. Traceability of the design will remain consistent and future testing of the individual modules interactions will finalize the layout on the final PCB design. The following figures show the evolution of the prototyping phase from the very first working breadboard all the way to the final working prototype.

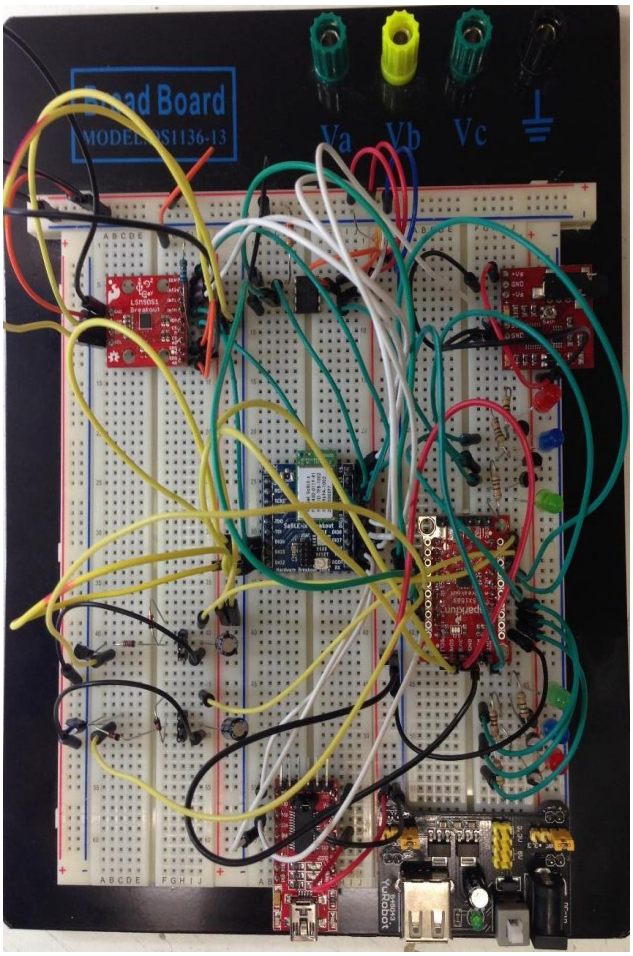


Figure 38: W.A.R.P. Hardware Design v1.0

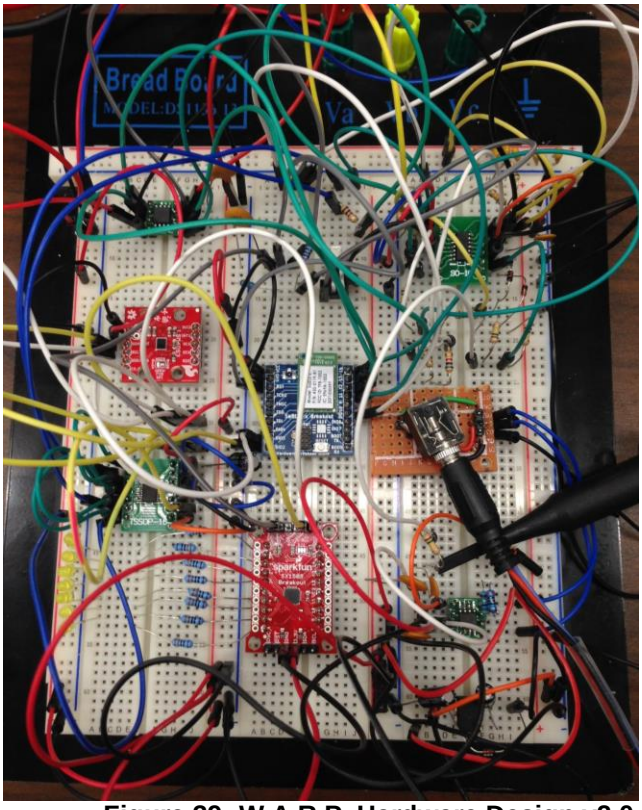


Figure 39: W.A.R.P. Hardware Design v2.0

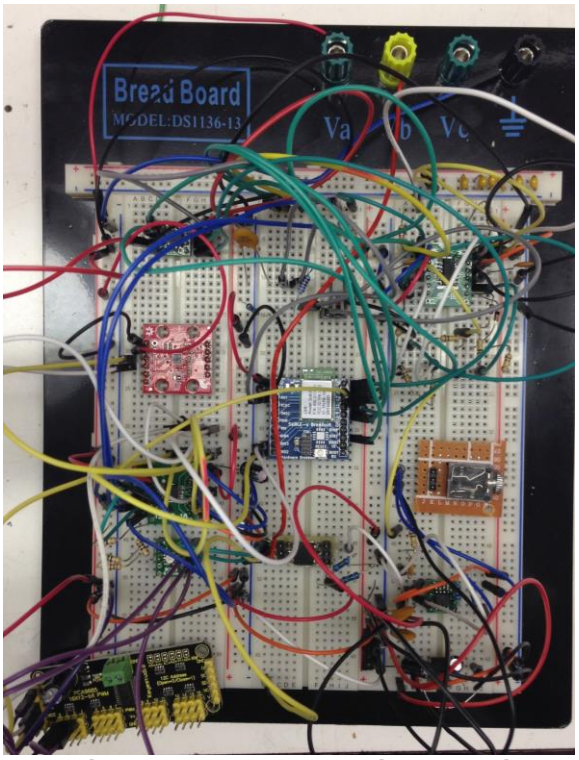


Figure 40: W.A.R.P. Hardware Design v3.0

In comparison the the first implementation of the breadboard, the final version has many modifications. The USB port and FTDI chip were removed since their functionality was not as vital with the inclusion of Bluetooth low energy. The SX1509 GPIO Expander was replaced with the PCA9685 due to the inability of the first IC to properly drive servo motors as the frequency of operation isn't optimal. The newer device is dedicated to driving PWM and has full 16-bit control over the frequency and output. The LSM9DS1 was changed with a LSM6DS3 due to reduced price and the inclusion of a tap/double-tap register found in the newer device which is capable of triggering an interrupt. Furthermore, the V014642 Solid State Relay was added as a method of controlling high currents powering the servo motors using a 3.3v logic level GPIO pin from the SaBLE-x MCU. Lastly, the RGB LED was verified to be working and bright enough when powered from a 3.3v power supply and controlled from the PWM driver.

8. Testing

Before the board was designed, much of the testing was done using simulation software such and on a breadboard using breakout boards. Once the PCB was manufactured and assembled, individual subsystems were tested to ensure proper output under real world conditions. The following sections will describe some of these testing methods and the equipment used.

8.1 Hardware Test Environment

The two currently planned and partially implemented test environments for the W.A.R.P. team are a physical one utilizing UCF supplied testing equipment and a virtual one consisting of verifying proposed designs within MultiSim. Physical testing is done within the UCF Senior Design Lab, room 456, using the provided tools such as the Tektronix MSO 4034B Oscilloscope which allows the team analyze the input/output waveforms of the analog and digital subsystems. Additional tools such as a Tektronix AFG 3022B Function Generator for simulation of AC signals such as those read by the EMG sensor, a Keithley 2230-30-1 Triple Channel DC Power Supply to supply a defined DC current and voltage and a Tektronix DMM4050 6-1/2 Digit Precision Multimeter to test continuity of the circuits before powering to ensure proper will also be utilized. The virtual testing will take place within MultiSim v14.0 for initial proving of W.A.R.P. analog designs in terms of expected outputs.

The team designed the PCB in such a way that used isolated subsystems and thusly broken out pins where applicable. The team was unable to afford one due to budget constraints, but was interested in using a Serial Logic Analyzer to verify the digital waveforms sent and received by the peripherals. This would allow the team to debug any potential communication problems between chips and easily test custom messages being sent through the various protocols utilized by the project. The team was looking at a Saleae Serial Logic Analyzer which costs about $219. Lastly, the team additionally planned on including multiple test points on the PCB, but due to size constraints, other methods described above were implemented instead.

8.2 Hardware Specific Testing

Since the majority of the project involves digital logic, the team has decided at the beginning to only simulate the analog portions of the schematic design.

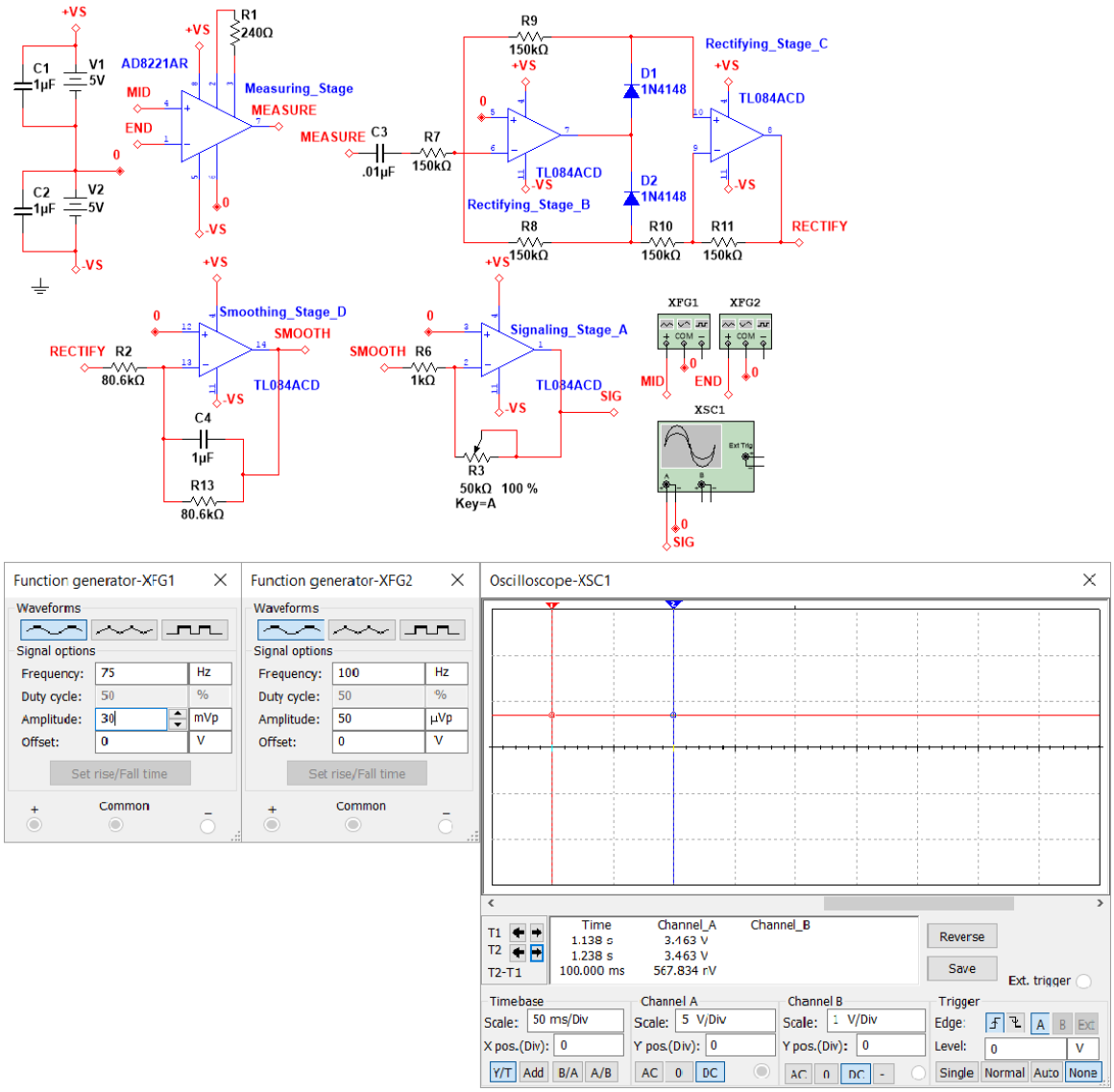


Figure 41: MultiSim Test of EMG Sensor Design

Figure 38 shows the verification of the EMG sensor module within MultiSim in order to assure proper functionality of expected gain from potential readings. In the above figure, the two extremes of human electromyography readings are input from function generators and after going through the instrumentation op-amp in the “Measuring_Stage”. The output is then moved through a general purpose amplifier that rectifies, smoothes and outputs the final signal that is received by other subsystems. A highlight with this particular design is that the trimpot potentiometer across the “Signaling_Stage_A” Op-Amp allows for variable gain before submitting the signal to the rest of the system.

8.3 Software Environments

Several environments exist during a product's life cycle, this allows for testing at different stages and scopes during development. This allows for a rolling deployment providing stages to rollback to in the event of a mad merge being pushed to some environment. Commonly these environments are Development, Testing, Staging, and Production.

8.3.1 Development

The development environment generally on the local machine of a product's developer. At this level, many differing tests can be performed while in this environment. These tests can be as simple as evaluating that a specific association between two database tables can be conditionally invalidated or validated based on some new addition. In many cases involving individuals who develop in conflicting operating systems, it is reasonable to attempt to unify these individual environments under one operating system. This can be done with the use of software such as Otto or Vagrant, both of which can be defined as headless virtual machines running the operating the team plans on deploying to for the production ready application.

8.3.2 Testing

The testing environment is used during the automated testing process, automated testing is a simple testing strategy that can be written to perform unit testing, acceptance testing, testing of mailers, and the testing of third party services. Traditionally, the database transactions and records are kept separate from the database belonging to the development environment and may be destroyed after the tests have completed.

8.3.3 Staging

Prior to deploying the application to a production server after clearing all automated tests, it is likely additional care will be taken to ensure the application behaves as intended in what is effectively an environment exactly the same as production. This will also act as a check to validate the server is configured properly and that all dependencies have been resolved.

8.3.4 Production

The Production environment is simply the application's final stage of deployment; this environment makes the product live to the end users.

8.4 Software Specific Testing

8.4.1 Client Applications

Outside of using the view to ensure that it is rendering correctly, testing can be managed with the use of equality checks with expected values of the state. Doing so, in the architecture chosen by the W.A.R.P., will prove to be a trivial addition as the state is represented as an immutable data structure and its next state can be expected by applying differ actions independently from the rest of the application. This is true for the web, mobile, and desktop applications as the state will be managed in the same way across these platforms.

8.4.2 Server Application

The same style of testing can be conducted for the server application as well, using equality assertion. Elixir includes ExUnit, a framework for creating a test suite, this permits the team to structure automated tests of different types. These tests are to be conducted in the Testing Environment and are to test the following: Models, Channels, and Controllers. The Model testing will ensure the validity of the database types and associations including any created constraints. Channel testing will cover websocket connections are broadcasting the correct information. Controller testing coverage will ensure the routes from the HTTP requests are returning the expected status under differing conditions involving authorization and authentication.

9. Implementation

As most previous sections covered research, design, and prototyping, this section will cover the final steps of converting the final CAD design to a fully functioning printed circuit board. This includes the bill of materials (BOM), renders of the final PCB and general manufacturing information.

9.1 Bill of Materials

The following table shows every single part which is required to assemble a single fully operational W.A.R.P. v1.0 printed circuit board. Within the BOM, the package description, identifier, part name, manufacturer, part number, quantity,

and unit cost when purchasing in quantities of 1 are shown. The team wants to clarify that this is the most expensive the board would be, and that higher quality components were chosen. Although the components for one board cost $65.73, when purchased in bulk, a single board may cost half. Furthermore, when optimizing for cost, lower quality components can be selected without risk of reduced performance. Specifically, the shielded inductors and filtering capacitors are many times over engineered to ensure proper operation. Lastly, these prices are displayed at the time of writing and may increase or decrease at a later date.

Table 9-1: Bill of Materials

Info	Identifier	Part Name	Manufacturer	Part Number	Quantity Needed	Unit Cost	Total Cost
1206	C1, C32, C33	22uF Capacitor (Ceramic)	Murata	GRM31CR61A226ME19L	3	$0.35	$1.05
0805	C12, C15, C19, C23, C24, C34, C35	10uF Capacitor (POL)	AVX Corporation	F920J106MPA	7	$0.31	$2.17
0805	C2	1nF Capacitor (Ceramic), 1000pF	Kemet	C0805C102K5RACTU	1	$0.10	$0.10
0805	C20	1uF Capacitor (Ceramic)	Murata	GRM219R61A105KA01D	1	$0.10	$0.10
0805	C3, C7, C9, C10, C11, C17, C18, C21, C22, C30, C31	100nF / .1uF Capacitor (Ceramic)	Murata	GRM219R71C104KA01D	11	$0.13	$1.43
0805	C4, C5, C6, C13, C36, C37	10uF Capacitor (Ceramic)	Murata	GRM21BR71A106KE51K	6	$0.24	$1.44
0805	C8, C25, C26, C27, C28, C29	10nF / 10,000pF (Ceramic)	Yageo	CC0805KRX7R9BB103	6	$0.10	$0.60
SOD-323F	D1, D2, D3	Switching Diode	Fairchild	1N4148WS	3	$0.15	$0.45
0603	F1	Ferrite Bead	Murata	BLM18HE152SN1D	1	$0.21	$0.21
Jumper	J1 (x5), J5 (x6)	.1" Jumper	3M	969102-0000-DA	11	$0.10	$1.10
Custom	J2	Male JST Connector	JST	S2B-PH-SM4-TB(LF)(SN)	1	$0.58	$0.58
Custom	J3	JTAG Connector	Samtec Inc	FTSH-105-01-F-DV-K	1	$3.02	$3.02

Custom	J6	3.5mm Jack	CUI Inc.	SJ-3524-SMT-TR	1	$1.37	$1.37
Custom	L1	2.2uH Inductor (35mOhm, IDC=4.7A)	Wurth Electronis	74438356022	1	$2.14	$2.14
Custom	L2	4.7uH Inductor (90mOhm, IDC=.93A)	TDK Corp	VLF302515MT-4R7M	1	$1.58	$1.58
0805	R1	953kOhm Resistor	Yageo	RC0805FR-07953KL	1	$0.10	$0.10
0805	R19, R20	1kOhm Resistor	Yageo	RC0805FR-071KL	2	$0.10	$0.20
0805	R2	180kOhm Resistor	Yageo	RC0805FR-07180KL	1	$0.10	$0.10
0805	R21, R22	80.6kOhm Resistor	Yageo	RC0805FR-0780K6L	2	$0.10	$0.20
0805	R23, R24, R25, R26, R28	150kOhm Resistor	Yageo	RC0805FR-07150KL	5	$0.10	$0.50
0805	R27	240 Ohm Resistor	Yageo	RC0805FR-07240RL	1	$0.10	$0.10
0805	R3	100kOhm Resistor	Yageo	RC0805FR-07100KL	1	$0.10	$0.10
0805	R29, R30, R31, R32	0.0Ohm Resistor	Yageo	RC0805JR-070RL	4	$0.01	$0.05
0805	R4	1MOhm Resistor	Yageo	RC0805FR-071ML	1	$0.10	$0.10
0805	R5, R6, R7	2.2kOhm Resistor	Yageo	RC0805FR-072K2L	3	$0.10	$0.30
0805	R8, R17, R18	10kOhm Resistor	Yageo	RC0805FR-0710KL	3	$0.10	$0.30
0805	R9, R16	301Ohm Resistor	Yageo	RC0805FR-07301RL	2	$0.10	$0.20
0805	R10, R13	88.7Ohm Resistor	Yageo	RC0805FR-0788R7L	2	$0.10	$0.20
0805	R11, R14	69.8Ohm Resistor	Yageo	RC0805FR-0769R8L	2	$0.10	$0.20
0805	R12, R15	174Ohm Resistor	Yageo	RC0805FR-07174RL	2	$0.10	$0.20
0805	R33, R34	220Ohm Resistor	Yageo	RC0805FR-07220RL	2	$0.10	$0.20
IC (6-PLCC)	RGB0, RGB1	RGB LED	Broadcom Limited	ASMB-TTB0-0A3A2	2	$1.28	$2.56
Custom	S1	Push-Button	Panasonic	EVQ-PNF04M	1	$0.72	$0.72

IC (16-QFN 3x3)	U1	High Power Voltage Regulator	TI	TPS62130RGTR	1	$2.93	$2.93
IC (14-TSSOP)	U10	Quad Op-Amp	TI	LMV614MTX/NOPB	1	$0.86	$0.86
IC (8-VSSOP)	U11	Voltage Inverter	TI	TL7660CDGKR	1	$1.43	$1.43
IC (8-VSSOP)	U12	Instrumentation Op-Amp	TI	INA826AIDGKR	1	$3.01	$3.01
IC (12-WSON 3x2)	U2	Logic Level Voltage Regulator	TI	#ERROR!	1	$2.60	$2.60
IC (39-SMD)	U3	SaBLE-x (Trace Antenna)	LSR	450-0119	1	$16.52	$16.52
IC (8-SOIC)	U4	1MB Flash Memory (8-SOP 150MIL)	Macronix	MX25R8035FM1IH1	1	$0.63	$0.63
IC (6-SMD)	U5, U7	Digital Relay	Vishay	VO14642AABTR	2	$3.04	$6.08
IC (28-TSSOP)	U13	PWM Driver	NXP Semi	PCA9685PW,118	1	$2.31	$2.31
IC (LGA-14L)	U8	Accelerometer / Gyroscope	ST	LSM6DS3	1	$3.93	$3.93
IC (14-QFN 2x2)	U9	Digital Potentiometer	TI	TPL0102-100RUCR	1	$1.76	$1.76
TOTAL					**104**		**$65.73**

9.2 PCB Manufacturing and Assembly

This section will highlight vendors which were being considered by the W.A.R.P. team. The primary motivations which dictate the criteria for selecting a vendor are mainly 1) Lead Time 2) Total Cost 3) Capabilities. Due to the short time frame of the project, it is incredibly important that manufacturing is as fast as possible and preferably less than a single week. Additionally, it is important that the price isn't too high, to allow the team to order at least two to four batches and remain under the budget. Lastly, since the board must be very small, the manufacturer and assembler must have very tight tolerances to work with small components in the 0201 and 0402 mm size ranges. Other important tolerances are minimum drill size, minimum trace width, and trace clearance. All these factors will be involved in making the final decision of the vendor to be used. Table 8-2 will outline the top choices for vendors:

Table 9-2: Compare and Contrast Manufacturers & Assemblers

	OSH Park	PCBWAY	Quality Manufacturing Services
Website	https://oshpark.com/	http://www.pcbway.com/	http://qmscfl.com/
Max # of Layers Possible	4	10	--
Min Trace Clearance	6 mils	4 mils	--
Min Trace Width	6 mils	4 mils	--
Min Drill Size	13 mils	8 mils	--
Price **(Rush Delivery)**	$5/sq. inch (3 Boards) + $89 Swift Fee (2 Layer) 5 Business Days	$10/sq. inch (5 boards) + $36 Express 24 Hour Manufacturing (2 Layer) 4 - 5 Business Days	Possibly Free Assembly.
Assembly	X	✓	✓

The final decision was to use www.PCBWay.com as a PCB manufacturer due to their clear pricing which can be seen through their online price calculator. In addition, they have a multitude of options for solder mask, trace width, via size, silk screen, PCB thickness and lead time. The team opted for black solder mask and electro less nickel immersion gold (ENIG) finish for the reduced chance of corrosion occurring and the added bonus of using the school colors of black and gold. Standard 1.6mm PCB thickness on a two-layer board with .3mm minimum via sizes were used. Although PCBWay allowed down to 6 mils minimum trace width and 6 mil trace clearance, the team opted to only go down to 8 mil trace width except when working with tight pitch integrated circuits. Similarly, with .3mm (11.81 mil) vias, the team only used a minimum of 13 mil vias to avoid any manufacturing defects. Smaller features are available from the manufacturer, but at an increased cost.

Quality Manufacturing Services Inc. (http://www.qmscfl.com/) was recommended to the team by colleagues and is further recommended by the W.A.R.P. team as a professional assembly service for printed circuit boards. In smaller quantities they normally charge $2 per chip (including ICs, resistors, capacitors, etc.), but offered to waive these charges for the team. Last minute, the team decided to update all passive components from 0402 to 0805 in the case

where QMS would require a fee so they can be easily soldered by hand. QMS ended up assembling all fourteen integrated circuits onto four different PCBs and this allowed the members of the team to solder the remaining 90 components by hand and using a hot air gun and solder paste. The only downside is that QMS required a 2 week lead time, but this was offset by the sheer cost of assembling four boards for $2 per chip. The final assembled product is shown in the figure below.

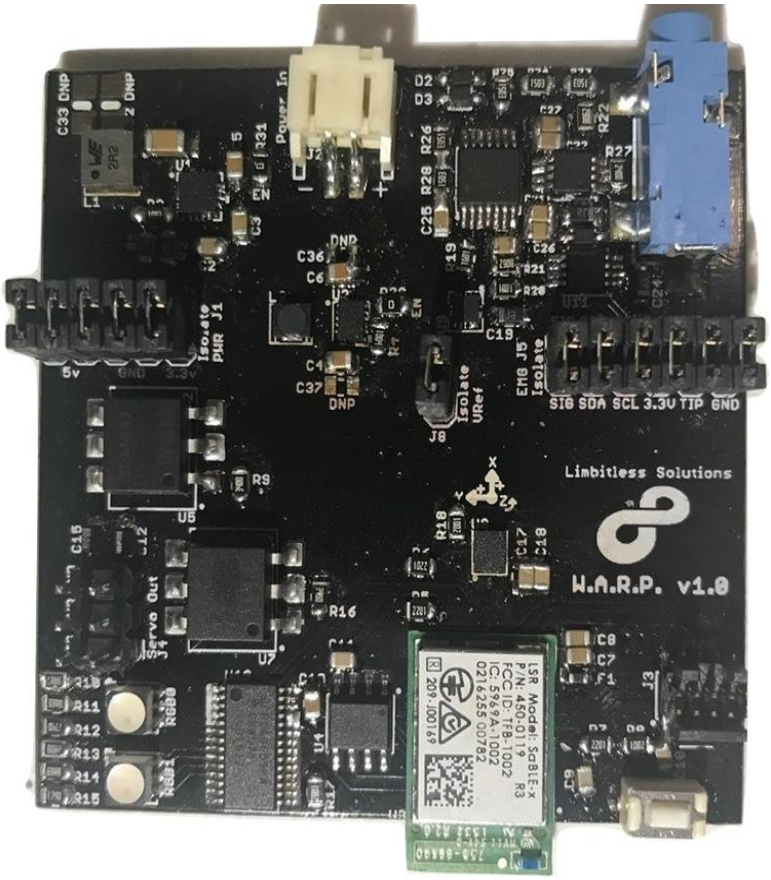


Figure 42: W.A.R.P. PCB v1.0.0

9.3 Embedded Implementation

The Embedded code used to run the CC2640 processor is separated into sections that handle the runtime operations of each major section of the W.A.R.P. module. The team started by using TI example code such as “Project Zero” and “SimplePeripheral” as starting points and past a certain point branched off into making customizations and updates in order to tailor the code for the specific board and project. The code was generally built in a modular fashion by adding multiple tasks or threads into the application code as a method of introducing new functionality. With this method new features can be easily built, enabled, disabled, and minimize their effect on other threads running concurrently. The major sections include the main function loop, the ICall task, the GAP task, and the Idle task.

Sections that control specific components of the W.A.R.P. module include the EMG, IMU, PWM, and LOG functionalities.

9.3.1 Main Function Loop

This section of code in the “warp.c” file pend on messages which can then be extracted, processed, and make API calls through the ICall library and GAP manager. This would then allow simple and abstracted control over the BLE stack by simply queueing up messages to this thread. When main BLE triggers are encountered they are processed through the main function loop and fed into their specific sub-functions.

9.3.2 GAP and ICall Tasks

These two tasks work together to allow BLE implementation into the WARP Embedded application. GAP specifically designated the containers for the information that will be passed over BLE while ICall allows for these containers to be interacted with by the application functions of the W.A.R.P. module.

9.3.3 Idle Task

Controls basic upkeep operations for the TI-RTOS in the event that no other functionality has the ability to run. Originally this thread simply managed data logging through callback functions, but due to multiple issues of other threads pre-empting the BLE task, many of these functions were removed. At the final stages, the idle thread simply executes power management code in order to place the device in low power mode when possible.

9.3.4 EMG Task

This thread mainly handles configuring and polling the ADC and finally transmitting of the signal produced by the EMG sensor over Bluetooth. This thread additionally handles control over the digital potentiometer as this directly affects the EMG sensor through both of its channels which update the voltage reference (vref) and EMG gain. The period of the data readings from the EMG can be configured by modifying the EMG service period characteristic. This thread receives messages from the IMU thread whenever tapping interrupts are triggered as a method of selecting a servo. This thread also communicates with the PWM task as a method of signaling when a servo motor should be triggered. Although not fully implemented, this thread also communicates with the data logging thread as a method of saving and restoring context of the calibration data

and logging EMG data as it comes in. This thread is responsible when properly configured to notify the host application of EMG data in real time.

9.3.5 IMU Task

This thread gives the connecting central device control over the LSM6DS3 inertial measurement which is done through the I2C bus. All registers of the LSM6DS3 can be accessed through the configuration characteristic with the data characteristic providing a consistent readout of the thermometer, gyroscope and accelerometer data. The readout speed of this section is also controllable through the period characteristic. As the other services, this thread can be configured to enabled or disabled and ble broadcasting can be controlled.

9.3.6 PWM Task

This thread acts similarly to the others, but since the PCA9685 PWM driver offloads processing and simply accepts commands and persists delivering pwm signals, this thread simply pends on a mailbox and waits for commands. The majority of the time, this thread is asleep and only active when a pwm device needs to be updated.

9.3.7 LOG Task

The LOG thread, or data logging thread directly interacts with the 1MB external flash memory through the SPI bus. This log acts to not only store and retrieve data logs from the flash, but it also manages reading and writing calibration settings in the flash so this configuration persists through power cycles. The goal of this thread is to also buffer data logs and efficiently read and write data pages in a way that would minimize erase cycles and maximize the external flash memory's lifetime. Ultimately, all the other threads who enable data logging would pass messages into the "logMbxHandle" in order to queue them up and construct data records to be stored in flash. The majority of the code for this feature is written, but not successfully integrated with the remaining code due to RTOS specific issues with memory.

9.3.8 Libraries

When possible, the team opted to used pre-existing libraries to prevent the need to write and test them from scratch. There were a few libraries which were written in order to create a mutex lock on the TI-RTOS drivers. The most notable are the "SensorI2C" library and "Pinterface" libraries which utilize semaphores as mutexes to require a thread to pend

on the lock being released before attempting access. Other libraries, such as the “ExtFlash” were modified to be more accessible. Device specific libraries were also designed for all I2C capable devices as a simpler interface, preventing the user from directly modifying device registers. The team ensured to reduce ambiguity and increase abstraction from hardware in all cases where applicable and where time permitted.

10. Administrative Content

10.1 Milestones

The W.A.R.P. team decided on a schedule which would be strictly adhered to in order to remain on track for project completion. Aligning the milestones with the team's solution for task management will ensure the team remains punctual. Milestones for this project revolve around the course schedule of Senior Design I & II which is instructed by Dr. Lei Wei.

Many of the milestones were scheduled in such a way to provide ample time for external factors which the team had no control over. For instance, when submitting a printed circuit board to be manufactured, it often takes time for the company to fabricate the board, assemble it, and finally ship it back. Following this example, a month of time was allocated for this entire process and another month for assembly of components in the case the board needs to be shipped to another company. Additional time was also factored into this calculation in the event the boards had defects. In the case that the team is on schedule, stretch goals will be attempted in an effort to resolve issues which may arise in the implementation process.

In addition, an early deadline for the finished documentation and product have been set for three weeks and two weeks in advance of their respective due dates to account for unforeseen circumstances during the research and development process. The schedule laid out in Table 10-1 below is intended as a reference and remains as the team's optimistic plan of finishing the tasks within the limited timeframe of the project with Figure 39 being a Gantt representation of the same timeline.

Table 10-1: Project Milestones

Objective	Description	Start Date	End Date	Duration
Senior Design I	Design, Plan, and Prototype	05/16/2016	08/02/2016	12 Weeks
Project Documentation		05/17/2016	07/15/2016	9 Weeks
	Initial Project Idea	05/17/2016	05/20/2016	3 Days
	Initial Proposal - Divide and Conquer	05/24/2016	06/03/2016	2 Weeks
	Revised Proposal	06/03/2016	06/05/2016	2 Days
	Table of Contents	06/04/2016	07/01/2016	4 Weeks

Objective	Description	Start Date	End Date	Duration
	Executive Summary	06/17/2016	06/18/2016	2 Days
	Background	06/18/2016	06/19/2016	2 Days
	Team Organization & Administrative	06/19/2016	06/20/2016	2 Days
	Project Description	06/21/2016	06/24/2016	5 Days
	Rough Draft Document	06/04/2016	07/08/2016	5 Weeks
	Final Document	06/04/2016	07/15/2016	6 Weeks
Research		05/30/2016	06/30/2016	4 Weeks
	Bluetooth Communication	05/30/2016	06/30/2016	4 Weeks
	PCB	05/30/2016	06/30/2016	4 Weeks
	Mobile Application	05/30/2016	06/30/2016	4 Weeks
	Database-Management System	05/30/2016	06/30/2016	4 Weeks
	Accelerometer	05/30/2016	06/30/2016	4 Weeks
	On-board Storage	05/30/2016	06/30/2016	4 Weeks
	In-System Programming	05/30/2016	06/30/2016	4 Weeks
	Physical Case Design	05/30/2016	06/30/2016	4 Weeks
	Power Assembly Design	05/30/2016	06/30/2016	4 Weeks
Order Prototyping Components		07/15/2016	07/22/2016	1 Week
Prototyping		07/22/2016	08/05/2016	2 Weeks
	Breadboard and Testing	07/22/2016	07/29/2016	1 Week
	Proto-Board and Testing	07/29/2016	08/05/2016	1 Week
PCB Acquirement		08/06/2016	08/15/2016	1 Week
	Finalize PCB design	08/06/2016	08/13/2016	1 Week

Objective	Description	Start Date	End Date	Duration
	Choose Parts for PCB	08/09/2016	08/13/2016	4 Days
	Place PCB and Components Order	08/13/2016	08/15/2016	2 Days
Senior Design II	Develop, Implement and Test	08/22/2016	12/02/2016	15 Weeks
Development		08/24/2016	09/23/2016	4 Weeks
	Physical Cuff Housing	08/24/2016	09/23/2016	4 Weeks
	Mobile Application	08/24/2016	09/23/2016	4 Weeks
	Database System	08/24/2016	09/23/2016	4 Weeks
Implementation		09/23/2016	10/07/2016	2 Weeks
	Solder PCB components	09/23/2016	10/07/2016	2 Weeks
	Mobile Application	09/23/2016	10/07/2016	2 Weeks
	Database System	09/23/2016	10/07/2016	2 Weeks
Testing		10/07/2016	10/21/2016	2 Weeks
Integration		10/21/2016	11/04/2016	2 Weeks
[Extra] Cushion Time		11/04/2016	11/18/2016	2 Weeks
Prepare Presentation		11/18/2016	11/25/2016	1 Week

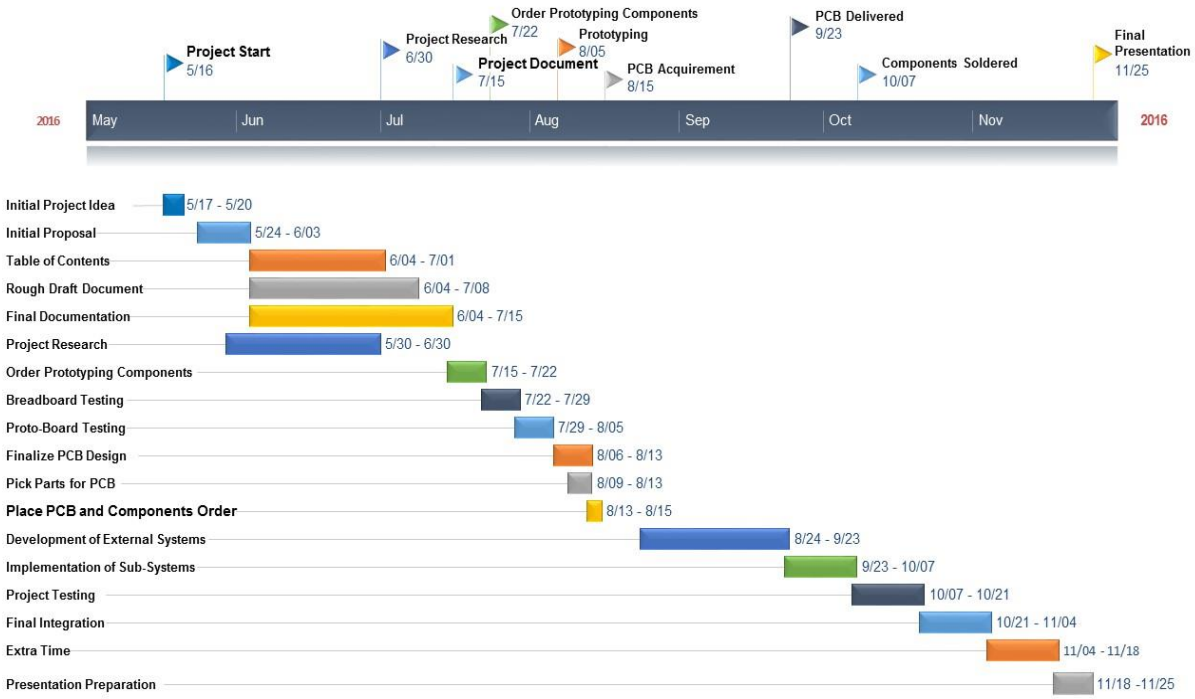


Figure 43: Gantt Chart representation of Timeline

10.2 Budget and Finance Discussion

10.2.1 Estimated Project Budget

Under the project goals reviewed and approved by Limbitless Solutions the team has been allotted up to $1000 in sponsorship funding for research and development in order to produce a functional prototype of W.A.R.P. To meet this monetary constraint, component price approximations have been laid out in the chart below. The majority of the cost associated with the project is involved in sending the board to be manufactured along with the overall cost of prototyping the designs. An estimation of costs is calculated below in order to best utilize the funds provided for the project. Any extra costs will be covered by the team members themselves.

Table 10-2: Estimated Budget

I.D.	Part Name	Estimated Cost
1	PCB Fabrication	$300
2	PCB Assembly	$250
3	Microcontroller	$5
4	Bluetooth Module	$15
5	Power Assembly	$100
6	Human Interface	$50
7	External Memory	$10
8	Accelerometer Module	$5
9	Server	$0
10	Discrete Components	$65
11	Backup Funds	$200
Total Cost		$1000

The team has decided to only use the funds provided by Limbitless when necessary. First and foremost, the team will utilize equipment which is available at no cost to engineering students by the University of Central

Florida. This equipment includes oscilloscopes, multimeters, soldering irons, 3D printers, and laser cutters. Limbitless has additional access to a full machine shop for any mechanical components to be produced. In addition to the available equipment, team members have various privately owned components which will be used when prototyping such as microcontrollers, discrete components, integrated circuits, and sensors.

In order to further reduce costs, free samples, evaluation modules, and server infrastructure will be requested from known contacts within the organization and from providers interested in assisting. With respect to the final cost of the circuit board, it is expected that the cost reflected throughout this project will not be the same for mass production. As such, the goal will be to produce a PCB which will cost below $50 per board in orders of higher quantity.

10.2.2 Itemized Purchases

The team wanted to be overly cautious and test every aspect of the project before manufacturing the board. This caution did serve the team well as the very first version of the printed circuit board worked exactly as designed. The team additionally purchased enough components to assemble 10 fully functioning boards.

Table 10-3: Itemized Purchases

Item	Distributor	Part Number	Quantity	Unit Price	Total Price
TI CC2650 LaunchPad	Texas Instruments	LBLA43A	2	$35.99	$71.98
SaBLE-x Breakout Board (Assembled)	Hardware Breakout LLC	BB-09	1	$44.99	$44.99
SaBLE-x Evaluation Kit	Mouser	450-0150	1	$69.99	$69.99
LSM9DS1 IMU Breakout Board	Sparkfun	SEN-13284	1	$24.95	$24.95
LSM6DS3 IMU Breakout Board	Sparkfun	SEN-13339	1	$19.95	$19.95
10 PCS SOP8, SO8, SOIC8	Amazon	B00JK8EYOG	1	$4.50	$4.50

Item	Distributor	Part Number	Quantity	Unit Price	Total Price
SMD to DIP8 Breakout Board					
5 PCS SOP16, TSSOP16 to DIP16 Breakout Board	Amazon	B01ENR4RWM	1	$4.94	$4.94
SX1509 GPIO Expander	Sparkfun	BOB-13601	1	$12.95	$12.95
MicroB USB Breakout Board	Sparkfun	BOB-12035	1	$1.95	$1.95
SOT-23 Breakout Board	Amazon	B01ENR5V0O	10	$0.69	$10.03
FT232RL USB to Serial Breakout Board	Sparkfun	BOB-12731	1	$14.95	$14.95
P-Channel Mosfet	Fairchild	FDN340P	10	--	--
Advancer Technologies Muscle Sensor v3	Sparkfun	SEN-13027	1	--	--
EMG Sensor Cable	Sparkfun	CAB-12970	1	--	--
EMG Biomedical Sensor Pads	Sparkfun	SEN-12969	1	--	--
XDS200	Texas Instruments	TMDSEMU200-U	1	--	--
15 PCs NPN Transistors	Amazon	2SC2655	1	6.37	6.37
PCB Manufacturing 10 PCs	PCBWay	--	1	96.00	96.00

Item	Distributor	Part Number	Quantity	Unit Price	Total Price
Bulk Order of Components for 10 PCB (BOM)	DigiKey	--	--	442.15	442.15
Secondary Order of components	DigiKey	--	--	53.30	53.30
1 MB Flash	Digikey	1092-1179-ND	--	15.42	15.42
Total Cost					**$894.42**

10.3 Team Organization

10.3.1 Overview

As with any engineered solution, a process must be established and maintained to ensure optimal workflow. Organization and overall understanding of the end goal is the root of any successful team, this is completed through a series of actions managed by a system of order. Clearly defining roles in a team of a smaller size can be difficult, as such each individual may play certain roles in different areas of the project depending on their skillset.

The W.A.R.P. team will be employing various industry standards to optimize workflow, and ensure proper channels of communication and planning as the project progresses. As a team primarily composed of engineers with a strong background in software development, the team decided to utilize the Agile Development Methodology to create a rigid and structured process for reaching goals. More specifically, eXtreme Programming (XP), which is a subset of the agile development process. This process will be used throughout both the hardware and software development life cycles. XP is a methodology designed to allow a team to quickly respond to changes in engineering requirements and promotes a paired approach to solving problems. Furthermore, this approach generally includes consideration of the following roles when making technical decisions.

10.3.1.1 Stakeholder(s)

Group of any users or individuals interfacing with or affected by the production of the project. Limbitless as the primary sponsor, their customers who use the product, and members of the W.A.R.P. team are all stakeholders of this project.

10.3.1.2 Product Owner

Individual or groups representing the stakeholder's interest without working directly on the project. This role is fulfilled by Albert Manero and Dominique Courbin along with the other directors of the organization as they represent the interests of Limbitless Solutions.

10.3.1.3 Team Lead

Due to the nature of the team, major decisions will be made through open discussion and an equally weighted vote in favor of a majority. Each individual member has expertise in a given area, with the responsibility to supply tasks and goals for the team to collaborate on. These members would also provide assistance to move past any difficulties in specific challenges.

10.3.1.4 Team Member

Individual on the team who is solely responsible for actively participating, attending major team meetings, completing individual research in addition to positively contributing to the collective group effort aimed at completing the W.A.R.P. project in a timely manner.

10.3.2 Communication

Effective communication is integral to working with a team of any size, this is especially true in situations involving constraints. Scheduling can often prove to be a difficult task to overcome, especially when time is limited. With this in mind, the need for a reliable channel of communication becomes a necessity. Slack (https://slack.com/) will be used to fulfill this need as it is an application which provides an effortless means of collaboration and communication for all team members via the website, desktop, and mobile applications.

Aside from meetings scheduled and satisfied in W.A.R.P.'s Slack channel, the team meets in person every Tuesday and Thursday of each week at a minimum from noon until four in the afternoon. In person meetings provide a platform for each member to express their thoughts and foster an active discussion concerning the entire project. Furthermore, this time can also be used to complete necessary administrative tasks such as stand-up meetings, a time to discuss team members' commitments, any issues, or

concerns regarding the upcoming tasks. The team intends to meet with Limbitless each week to inform them of progress and ensure the project direction is in-line with organizations expected outcome.

10.3.3 Information Sharing

A medium in which to share information is fundamental when working on a project with frequent changes, preferably one which provides members with real time updates on all shared files. All documentation and reading material created or acquired by the team will be easily accessible for each member via Google Drive, an application which allows for sharing of cloud stored documents, pictures and data. A subsidiary of Google Drive, known as Docs, provides the team with an outlet to create, edit and collaborate on documentation.

10.3.4 Version Control

A common issue in software development is managing a code base that requires the contribution of several individuals simultaneously while also providing that the code is being altered in a manner which ensures that changes are noticed and conflicts can easily be managed. This issue can be solved using version control software such as “Apache Subversion” (SVN) or “Git”. These systems allow each individual to commit changes to a single repository and branch off of it to add new features. All commits require a connection to the centralized repository in SVN, as this is considered a centralized version control system. Unlike with SVN, Git allows for changes to be committed locally and can be pushed to the remote repository at a later time as this is a distributed version control system.

Using Git, individuals can easily track new changes using code diffing, a line-by-line changelog which highlights over new additions and deletions. This does come at a cost, as Git is often considered to have a much steeper learning curve. With Git having a vast amount of tooling, and the team already having experience using it, Git is the version control system of choice to be used for the W.A.R.P. project. Additionally, the team will be using GitHub, a free service which hosts a Git repository online, providing team members access whenever they have an internet connection.

Git and GitHub are not only useful for high level and embedded software, but will also be used to keep track of various hardware designs as they are being created. This will be important if different members are working on separate sub-systems of the printed circuit board and with

keeping a version history of design changes. Additionally, this will allow members to revert back to previous files if a problem occurs in a later iteration of the project. Furthermore, this system will organize the modularity of the design process and provide a remote backups of project assets.

10.3.5 Task Management

In a scenario involving team members, it is important for each member to know what is required of them. Task management involves ranking tasks by priority and reminding team members to complete tasks while informing them of any progress made in order to avoid duplication of work. This process can be put into practice with the use of task management software, an online forum in the form of a general board for posting new tasks in detail, self-assigning tasks, and a visualization of task progression.

Waffle and Pivotal Tracker are both used to manage tasks across the team and provide feedback regarding the team's throughput. Pivotal Tracker is closed source but provides extensive tooling for progress tracking and comprehensive tagging with the allowance of more than one individual per task. There is automatically velocity tracking, calculating the speed of development based on the completion of sized tasks over time, and methods for determining the expected completion date to effectively ensure on time delivery.

Waffle provides a tooling similar to Pivotal Tracker without some of the features mentioned above, yet is open source and is available at no cost. Tasks can be created and assigned a value based on the amount of time to complete and then assigned to individuals. Taking into consideration the above, in addition to the fact that Waffle also easily integrates with GitHub while remaining a free service makes it a more favorable option in comparison to Pivotal Tracker.

During the development of the W.A.R.P. application stack, a new tool was added to GitHub, "GitHub Projects", allowing the team to simply use issues and assigning cards to those within the working project instead of resorting to an external application. While lacking some features of the other applications, this proves to be simple enough for working with small teams.

10.3.6 Design Software

The choice of design software in conjunction with the previously described methodologies and applications is paramount to successfully

manufacturing a working printed circuit board. CadSoft's EAGLE PCB Design Software was selected since it is, by some, considered to be an industry standard. A few team members already have experience using the software, and the "lite" version of the software is available for free. The team may be required to use more advanced features not available in the free version and may, at some point, decide to purchase the professional version to accommodate those requirements.

10.3.7 Consultations

The W.A.R.P. team, although experienced in various areas of software development and electrical engineering, plans on working with experts who can provide guidance. The team will select and work closely with three faculty advisors who have strong backgrounds in Computer Science, Computer Engineering, and Electrical Engineering. Consultations will be requested in the case that the team is in need of technical or managerial direction. As the team's sponsor, Limbitless Solutions will be provided with updates and asked to assist the team through professional contacts when available. As the technical documentation and designs are created, the team may request a review from the above parties before finalizing the designs.

10.3.8 Faculty Advisors

Throughout the design process of Senior Design I and implementation process of Senior Design II, the team will work closely with three advisors. Each advisor is a faculty member at the College of Engineering and Computer Science at UCF and were hand selected by the team members based on positive past experiences as their students. After meeting with each professor individually, the team unanimously agreed upon working with the following professors as the team's advisers:

- Professor Reza Abdolvand
- Professor Ronald DeMara
- Doctor Albert Manero

Each advisor has an area of expertise in the fields of Electrical Engineering, Computer Engineering, and Limbitless Solutions' specific skillset. The team plans on consulting with the advisors regarding design and technical decisions based on their experience working with similar technologies. It is the hopes that with their oversight, W.A.R.P. will have less pitfalls and have a higher probability of success.

10.3.9 Sponsors

As the sponsor of the W.A.R.P. project, Limbitless Solutions provided the basic ideas and requirements for the project, in addition to other resources to help the team succeed. Limbitless provides funding, various contacts in industry, and general technical and team advice. From start to finish, the team has worked primarily with the following directors of Limbitless Solutions to help make this project a reality:

- Dominique Courbin
- Albert Manero

11. Conclusion

In conclusion, this paper briefly describes previous projects which were built before the W.A.R.P. project in order to set up a basis for drawing a difference in design. The mission statement of the project then describes what this team sets out to accomplish through the combination and integration of electronics, embedded hardware & software, mobile software, and server-side software. This paper further touches on the design issues which are involved in creating such a device which is capable of interfacing on a hardware level with multiple LEDs, multiple servo motors, external, and even internal sensors and memory.

The primary goal of this device is to provide the architecture for wireless communication from the transport layer through to the application layer of BLE stack from both the embedded server side, and client side software. Furthermore, the goal is to utilize this communication to demonstrate advanced utilization of this wireless link for use by Limbitless Solutions and their future products.

The Wireless Applications of a Refactored Prosthesis project aimed to advance the technology produced by Limbitless Solutions and the team went above and beyond to add innovative features and functionality which can be used within the prosthetic arms. The printed circuit board ended up regulating a 7.4v nominal voltage to both 3.3v (.3A) and 5v (3A) to power the remainder of the circuit. The EMG sensor was updated to work at lower operating voltages and a digital potentiometer was included to control the EMG gain and generate a programmable reference voltage for the analog comparator. A 6-axis inertial measurement unit was added with the capability of detecting single and double taps as a method of alternating control between multiple servo motors. A PWM driver, and two bright multicolor LEDs were incorporated into the design in order to control the aesthetics of the device and two solid state relays were utilized in order to power down the servo motors when unused. All of these features were then integrated and controlled using the SaBLE-x / CC2640 Cortex-M3/M0 processor. Furthermore, all this data can be configured wirelessly using Bluetooth low energy and data logging using the 1 MB external flash can be achieved.

With all these features in hardware, there is no limit to what can be achieved through software. The embedded software brings all the hardware together and interfaces directly with a custom cross-platform mobile application developed using React Native. This application is used to visualize data broadcasted from the device such as the live EMG graph in addition to calibrating the device using various widgets in the menus.

Although this team has achieved many successes, the WARP team calls upon future developers and engineers to utilize the platform developed with the goal of increasing functionality, reducing cost & size with the hopes of perfecting the product so it can make the world a better place.

Appendices

Appendix A – Abbreviations

Abbreviation	Definition
ABS	Acrylonitrile Butadiene Styrene
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
API	Application Program Interface
AWS	Amazon Web Service
BIM	Boot Image Manager
BLE	Bluetooth Low Energy
BOM	Bill of Materials
CDN	Content Delivery Network
DARPA	The Defense Advanced Research Projects Agency
DC	Direct Current
DIP	Dual-Inline Package
DOM	Document Object Model
EAGLE	Easy Applicable Graphical Layout Editor
EC2	Elastic Compute Cloud
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMC	Electro-Magnetic Compatibility
EMF	Electric and Magnetic Fields
EMG	Electromyography
EN	Harmonized European Standard
ERM	Electromagnetic compatibility and Radio spectrum Matters

FCC	Federal Communications Commission
FDA	Food and Drug Administration
GPIO	General-Purpose Input / Output
HCI	Host Controller Interface
HEC	Harris Engineering Corporation
HOQ	House of Quality
I^2C	Inter-Integrated Circuit Protocol
IC	Integrated Circuit
IP	Internet Protocol
IoT	Internet-of-Things
IMU	Inertial Measurement Unit
ISM	Industrial, Scientific and Medical
ISP	In-System Programmer
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group
JWT	JSON Web Token
LED	Light Emitting Diode
LTE	Long Term Evolution
MCU	Microcontroller Unit
MPL	Modular Prosthetic Limb
MVCC	Multi-Version Concurrency Control
NFC	Near Field Communication
OAD	Over-Air Download
OS	Operating System
PCB	Printed Circuit Board
PLA	Polylactic Acid

PWM	Pulse Width Modulation
RGB	Red, Green, and Blue
RTC	Real Time Clock
RTOS	Real Time Operating System
Rx	Receive
S3	Simple Storage Service
SDK	Software Development Kit
SHF	Super-High Frequency
SMD	Surface Mount Device
SPI	Serial Peripheral Interface
SVN	Apache Subversion
TCP	Transmission Control Protocol
TED	Technology, Engineering, Design
TI	Texas Instruments
T.U.B.A.	The Ultimate Bionic Arm
Tx	Transmit
UART	Universal Asynchronous Receiver / Transmitter
UCF	University of Central Florida
UHF	Ultra-High Frequency
UI	User Interface
USB	Universal Serial Bus
UUID	Universally Unique Identifier
W.A.R.P.	Wireless Applications of a Refactored Prosthesis
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
XP	Extreme Programming

Appendix B - List of Tables

Appendix C - Table of Figures

Appendix D - Datasheets

Component	Part Number
BLE Module / Processor	SaBLE-x 450-0119-R1
BLE Module / Processor	CC2650
BLE Module / Processor	CC2640
Voltage Regulator	LMZ21701SILR
Voltage Regulator	TPS82130
External Flash Memory	MX25R8035F
GPIO Expander	SX1509
P-Channel Mosfet	FDN340P
Switching Diode	1N4148
9-Axis IMU	LSM9DS1
Voltage Inverter	ICL7660
Instrumentation Amplifier	AD8221
General Purpose Quad Op-Amp	TL084
Piranha 5mm RGB LED	YSRGB7A5BSW25

Appendix E – References

- Texas Instruments. (2016, June). SYS/BIOS (TI-RTOS Kernel) User's Guide (v6.46) [Online]. Available: http://www.ti.com/lit/ug/spruex3q/spruex3q.pdf
- Texas Instruments. (2016, June). TI-RTOS 2.20 User's Guide (Rev. M) [Online]. Available: http://www.ti.com/lit/ug/spruhd4m/spruhd4m.pdf
- Texas Instruments. (2016, June). TI-RTOS 2.20 for CC13xx/CC26xx SimpleLink Wireless MCUs Getting Started Guide (Rev. C) [Online]. Available: http://www.ti.com/lit/ug/spruhu7d/spruhu7d.pdf
- Texas Instruments. (2015, May). Intro to the TI-RTOS Kernel Workshop Student Guide (v4.00) [Online]. Available: https://training.ti.com/sites/default/files/docs/TI_RTOS_Kernel_Workshop_Student_Guide_rev4.00_1.pdf
- Texas Instruments. (2016, June). CC13xx, CC26xx SimpleLink Wireless MCU Technical Reference Manual (Rev. F) [Online]. Available: http://www.ti.com.cn/cn/lit/ug/swcu117f/swcu117f.pdf
- Texas Instruments. (2016, June). CC2640 and CC2650 SimpleLink Bluetooth low energy Software Stack 2.2.0 Developer's Guide (Rev. C) [Online]. Available: http://www.ti.com/lit/ug/swru393c/swru393c.pdf
- Texas Instruments. (2015, May). CC2538/CC26xx Serial Bootloader Interface (Rev. A) [Online]. Available: http://www.ti.com/lit/an/swra466a/swra466a.pdf
- Texas Instruments. (2015, August). CC26xx/CC13xx Power Management Software Developer's Reference Guide [Online]. Available: http://www.ti.com.cn/cn/lit/ug/swra486/swra486.pdf
- M. Setton. J. Boe. (2016, March). A Guide to SensorTag Hackathons: Resources [Online]. Available: http://www.ti.com/lit/wp/swry023/swry023.pdf
- LSR. (2015, Feb 5). SaBLE-x Development Board User Guide (Rev. 1.4) [Online]. Available: https://www.lsr.com/downloads/products/330-0168.pdf
- Texas Instruments. (2016, June 27). TI-Simplelink [Repository]. Available: https://github.com/ti-simplelink/
- Texas Instruments. (2016, July). TI Resource Explorer [Repository]. Available: http://dev.ti.com/tirex/#/Device/CC2650

- E2E members. (2016, April 23). BLE Wiki [Online]. Available: http://processors.wiki.ti.com/index.php/Category:BluetoothLE
- Texas Instruments. (2016, July). TI Reference Designs [Online]. Available: http://www.ti.com/general/docs/refdesignsearchresults.tsp
- Texas Instruments. (2016, July). TI Webench [Online]. Available: http://www.ti.com/lsds/ti/analog/webench/overview.page?DCMP=analog_power_mr&HQS=webench-pr
- S. Monk, *Make Your Own PCBs With EAGLE From Schematic Designs to Finished Boards*, New York: McGraw-Hill Education, 2014.
- Cadsoft Computer, *EAGLE Manual Version 7.3*, 4th ed. Florida, 2015.
- A. Weiler and A. Pakosta, *High-Speed Layout Guidelines, 2006.*
- P. Horowitz and W. Hill, *The Art of Electronics*, 3rd ed. New York, Cambridge University Press, 2015.
- J. W. Nilsson and S. A. Riedel, *Electric Circuits*, 10th ed. New Jersey, Pearson, 2015.

Appendix F – Permissions

Re: Request Permission to use Images

Dominique Courbin <dominique@limbitless-solutions.org>

Sat 7/30/2016 4:28 PM

To:Niko Tubach <Ntubach@knights.ucf.edu>;

Cc:Timothy Ashley <tbash@Knights.ucf.edu>; Dmor574 <Dmor574@knights.ucf.edu>;

Depending on the image, you may have to cite the photographer as well, but yeah. No problem!

On Jul 30, 2016 4:15 PM, "Niko Tubach" <Ntubach@knights.ucf.edu> wrote:

> Hey Dominique,
>
> My team and I are writing our final report for Senior Design I and plan on using various images available on the Limbitless website in addition to a picture of TUBA's PCB and the Green PCB currently used. We require written email permission in its usage within our documentation.
>
> All displayed images will be properly cited and all credit will be given back to Limbitless Solutions for their use.
>
> Thank you for your time,
>
> ---
> Niko Tubach
> University of Central Florida
> B.S. Computer Engineering 16'
> W.A.R.P. Team

Request Permission to use Figures

Niko Tubach

Sat 7/30/2016 4:28 PM

Sent Items

To:ti-cares@ti.com <ti-cares@ti.com>;

Cc:Dmor574 <Dmor574@knights.ucf.edu>; Timothy Ashley <tbash@Knights.ucf.edu>;

Hello,

I am currently an Engineering Student at the University of Central Florida working on a Senior Design project. My team and I are writing our final report and plan on using diagrams and figures created by your company involving the CC2640 and we require written email permission in its usage within our documentation.

All displayed images will be properly cited and all credit will be given back to Texas Instruments for their use.

Thank you for your time,

--
Niko Tubach
University of Central Florida
B.S. Computer Engineering 16'
W.A.R.P. Team

Request Permission to use Image

Niko Tubach

Sat 7/30/2016 6:44 PM

Sent Items

To:sales@lsr.com <sales@lsr.com>;

Cc:Dmor574 <Dmor574@knights.ucf.edu>; Timothy Ashley <tbash@Knights.ucf.edu>;

Hello,

I am currently an Engineering Student at the University of Central Florida working on a Senior Design project. My team and I are writing our final report and plan on using a table on general power characteristics of the SaBLE-x and we require written email permission in its usage within our documentation.

All displayed images will be properly cited and all credit will be given back to LSR for their use.

Thank you for your time,

--
Niko Tubach
University of Central Florida
B.S. Computer Engineering 16'
W.A.R.P. Team

X

Appendix G – W.A.R.P. BLE Commands

W.A.R.P. Profile UUID: 712122F7-F6AF-4A4F-9AC2-EB7BF04869D6

Device Information Service UUID: 180A

System ID UUID: 2A23 *R*

Model Number UUID: 2A24 *R*

Set by Software

Serial Number UUID: 2A25 *R*

Set by Software

Firmware Revision UUID: 2A26 *R*

Set by Software

Hardware Revision UUID: 2A27 *R*

Set by Software

Software Revision UUID: 2A28 *R*

Set by Software

Manufacturer Name UUID: 2A29 *R*

Set by Software

IEEE11073 UUID: 2A2A *R*

PnP ID UUID: 2A20 *R*

IMU Service UUID: BB08

IMU Data UUID: BB09 *R/N*

Fourteen Bytes total

Bytes [0] – [1] are temperature readings

Bytes [2] – [7] are gyroscope readings

X axis: [2] – [3]

Y axis: [4] – [5]

Z axis: [6] – [7]

Bytes [8] – [13] are accelerometer readings

X axis: [8] – [9]

Y axis: [4] – [11]

Z axis: [6] – [13]

IMU Conf UUID: BB0A *R/W*

Three Bytes total

Byte 1: Indicates Config command

Byte 2: Holds read/write value of registers to interact with

Byte 3: Holds read/write byte to transmit from/to registers

Write 01 XX XX to start IMU thread and data retrieval (00 in first byte will stop thread)
Write 03 XX XX to start IMU transmit (02 in first byte will stop transmit)
Write 04 XX XX to Disable IMU gyro & accel
Write 05 XX XX to Enable IMU gyro & accel to default values
Write 06 (IMU Register Byte) XX followed by a read to see written register value in Byte 3
Write 07 (IMU Register Byte) (IMU Register Value) to set said IMU register (Byte 2) to said value (Byte 3)
*A full listing of available imu registers and applicable bits can be found in warp_imu.h

IMU Period UUID: BB0B *R/W*

Two Bytes total used together to delay thread in milliseconds

Lower value will update notify faster by lowering delay in thread sleep. Default value = 250

IMU Interrupt UUID: BB0C *R/N*

One Byte total used to notify app of IMU interrupt

Becomes set by interrupts of the IMU currently configured by default IMU values to detect single taps. When interrupt is detected, the chosen servo will be alternated in current codebase.

EMG Service UUID: BB18

EMG Data UUID: BB19 *R/N*

Two Bytes total used to notify app of ADC value

EMG Conf UUID: BB1A *R/W*

Three Bytes total

Byte 1: Indicates Config command
Byte 2: Holds read/write value of registers to interact with
Byte 3: Holds read/write byte to transmit from/to registers
Write 01 XX XX to start EMG thread and data retrieval (00 in first byte will stop thread)
Write 03 XX XX to start EMG transmit (02 in first byte will stop transmit)

EMG Period UUID: BB1B *R/W*

Two Bytes total used together to delay thread in millisecond

Lower value will update notify faster by lowering delay in thread sleep. Default value = 250

EMG Threshold UUID: BB1C *R/W*

Two Bytes total used together to control internal software threshold that dictates when chosen servo will trigger.

EMG Vref UUID: BB1D *R/W*

Single Byte that is used to control the resistance of Wiper A of the potentiometer that is utilized as a voltage reference in the W.A.R.P. design.

EMG Gain UUID: BB1E *R/W*

Single Byte that is used to control the resistance of Wiper B of the potentiometer that is utilized as gain for the final output of the EMG signal. (Higher value = Higher gain)

PWM Service UUID: BB28

PWM Conf UUID: BB29 *R/W*

Three Bytes total

Byte 1: Indicates Config command
Byte 2: Holds read/write value of registers to interact with
Byte 3: Holds read/write byte to transmit from/to registers
Write 01 XX XX to start PWM thread and data retrieval (00 in first byte will stop thread)
Write 03 XX XX to start PWM transmit (02 in first byte will stop transmit)
Write 04 XX XX to Disable PWM
Write 05 XX XX to Enable PWM with default configuration
Write 06 (PWM Register Byte) XX followed by a read to see written register value in Byte 3
Write 07 (PWM Register Byte) (PWM Register Value) to set said PWM register (Byte 2) to said value (Byte 3)
Write 0A 0X (0 is RGB0, 1 is RGB1) 0X (0 – 7 chooses hardcoded color of: Black, White, Red, Blue, Yellow, Green, Orange, Purple)

PWM Setting UUID: BB2A *R/W*
Seven Bytes total used to read/write LED values
0X (0 is RGB0, 3 is RGB1) 0XXX (0 – 4095 for RED) 0XXX (0 – 4095 for GREEN) 0XXX (0 – 4095 for BLUE)

LOG Service UUID: BB38

LOG Data UUID: BB39 *R/N*
Thirty-two Bytes for streaming log data

LOG Conf UUID: BB3A *R/W*
Four Bytes total
- **Byte 1: Configure Command (Special Command for Data Dump)**
- **Byte 2: Page number**
- **Byte 3: Upper Offset value**
- **Byte4: Lower Offset value**

LOG Setting UUID: BB3B *R/W*
Sixteen Bytes total used to read/write values from/to memory

Battery Service UUID: 180F

Battery Level UUID: 2A19 *R*

Generic Config Values

THREAD DISABLE	0x00
THREAD ENABLE	0x01
TRANSMIT DISABLE	0x02
TRANSMIT ENABLE	0x03
DEVICE DISABLE	0x04
DEVICE ENABLE	0x05
DEVICE READ	0x06
DEVICE WRITE	0x07
SPECIAL CFGs	0x08 – 0xFE
ERROR	0xFF

Un-Implemented Services

Connection Control Service
OAD Service